ecma INTERNATIONAL

Standard ECMA-376

1st Edition / December 2006

Office Open XML File Formats

Rue du Rhône 114 CH-1204 Geneva T: +41 22 849 6000 F: +41 22 849 6001

Standard
ECMA-376

1st Edition / December 2006

Office Open XML File Formats

Office Open XML

Part 1: Fundamentals

December 2006

Table of Contents

Foreword

This multi-part Standard deals with Office Open XML Format-related technology, and consists of the following parts:

- **Part 1: "Fundamentals" (this document)**
- Part 2: "Open Packaging Conventions"
- Part 3: "Primer"
- Part 4: "Markup Language Reference"
- Part 5: "Markup Compatibility and Extensibility"

Parts 2 and 4 include a number of annexes that refer to data files provided in electronic form only.

Introduction

This Part is one piece of a Standard that describes a family of XML schemas, collectively called *Office Open XML*, which define the XML vocabularies for word-processing, spreadsheet, and presentation documents, as well as the packaging of documents that conform to these schemas.

The goal is to enable the implementation of the Office Open XML formats by the widest set of tools and platforms, fostering interoperability across office productivity applications and line-of-business systems, as well as to support and strengthen document archival and preservation, all in a way that is fully compatible with the large existing investments in Microsoft Office documents.

The following organizations have participated in the creation of this Standard and their contributions are gratefully acknowledged:

Apple, Barclays Capital, BP, The British Library, Essilor, Intel, Microsoft, NextPage, Novell, Statoil, Toshiba, and the United States Library of Congress

1. Scope

This Standard defines Office Open XML's vocabularies and document representation and packaging. It also specifies requirements for consumers and producers of Office Open XML.

2. Conformance

The text in this Standard is divided into *normative* and *informative* categories. Unless documented otherwise, any feature shall be implemented as specified by the normative text describing that feature in this Standard. Text marked informative (using the mechanisms described in §7) is for information purposes only. Unless stated otherwise, all text is normative.

Use of the word “shall” indicates required behavior.

Any behavior that is not explicitly specified by this Standard is implicitly unspecified (§4).

2.1 Goal

The goal of this clause is to define conformance, and to provide interoperability guidelines in a way that fosters broad and innovative use of the Office Open XML file format, while maximizing interoperability and preserving investment in existing files and applications (§4). By meeting this goal, this Standard benefits the following audiences:

- Developers that design, implement, or maintain Office Open XML applications.
- Developers that interact programmatically with Office Open XML applications.
- Governmental or commercial entities that procure Office Open XML applications.
- Testing organizations that verify conformance of specific Office Open XML applications to this Standard. (Note that this Standard does not include a test suite.)
- Educators and authors who teach about Office Open XML applications.

2.2 Issues

To achieve the above goal, the following issues need to be considered:

1. The application domain encompasses a range of possible consumers (§4) and producers (§4) so broad that defining specific application behaviors would restrict innovation. For example, stipulating visual layout would be inappropriate for a consumer that extracts data for machine consumption, or that renders text in sound. Another example is that restricting capacity or precision runs the risk of diluting the value of future advances in hardware.
2. Commonsense user expectations regarding the interpretation of an Office Open XML package (§4) play such an important role in that package's value that a purely syntactic definition of conformance would fail to effect a useful level of interoperability. For example, such a definition would admit an application that reads a package, and then writes it in a manner that, though syntactically valid, differs arbitrarily from the original.
3. Legitimate operations on a package include deliberate transformations, making blanket change prohibitions inappropriate in the conformance definition. For example, collapsing spreadsheet formulas to their calculated values, or converting complex presentation graphics to static bitmaps,

could be correct for an application whose published purpose is to perform those operations. Again, commonsense user expectation makes the difference.

4. Existing files and applications exercise a broad range of formats and functionality that, if required by the conformance definition, would add an impractical amount of bulk to the This Standard and could inadvertently obligate new applications to implement a prohibitive amount of functionality. This issue is caused by the breadth of currently available functionality and is compounded by the existence of legacy formats.

2.3 What this Standard Specifies

To address the issues listed above, this Standard constrains both syntax and semantics, but it is not intended to predefine application behavior. Therefore, it includes, among others, the following three types of information:

1. Schemas and an associated validation procedure for validating document syntax against those schemas. (The validation procedure includes un-zipping, locating files, processing the extensibility elements and attributes, and XML Schema validation.)
2. Additional syntax constraints in written form, wherever these constraints cannot feasibly be expressed in the schema language.
3. Descriptions of element semantics. The semantics of an element refers to its intended interpretation by a human being.

2.4 Document Conformance

Document conformance is purely syntactic; it involves only Items 1 and 2 in §2.3 above.

- A conforming document shall conform to the schema (Item 1) and any additional syntax constraints (Item 2).
- The document character set shall conform to the Unicode Standard and ISO/IEC 10646-1, with either the UTF-8 or UTF-16 encoding form, as required by the XML 1.0 standard.
- Any XML element or attribute not explicitly included in this Standard shall use the extensibility mechanisms described by Parts 4 and 5 of this Standard.

2.5 Application Conformance

Application conformance is purely syntactic; it also involves only Items 1 and 2 in §2.3 above.

- A conforming consumer shall not reject any conforming documents of the document type (§4) expected by that application.
- A conforming producer shall be able to produce conforming documents.

2.6 Interoperability Guidelines

[*Guidance*: The following interoperability guidelines incorporate semantics (Item 3 in §2.3 above).

For the guidelines to be meaningful, a software application should be accompanied by publicly available documentation that describes what subset of this Standard it supports. The documentation should highlight

any behaviors that would, without that documentation, appear to violate the semantics of document elements. Together, the application and documentation should satisfy the following conditions.

1. The application need not implement operations on all elements defined in this Standard. However, if it does implement an operation on a given element, then that operation should use semantics for that element that are consistent with this Standard.
2. If the application moves, adds, modifies, or removes element instances with the effect of altering document semantics, it should declare the behavior in its documentation.

The following scenarios illustrate these guidelines.

- A presentation editor that interprets the preset shape geometry "`rect`" as an ellipse does not observe the first guideline because it implements "`rect`" but with incorrect semantics.
- A batch spreadsheet processor that saves only computed values even if the originally consumed cells contain formulas, may satisfy the first condition, but does not observe the second because the editability of the formulas is part of the cells' semantics. To observe the second guideline, its documentation should describe the behavior.
- A batch tool that reads a word-processing document and reverses the order of text characters in every paragraph with "Title" style before saving it can be conforming even though this Standard does not anticipate this behavior. This tool's behavior would be to transform the title "Office Open XML" into "LMX nepO eciffO". Its documentation should declare its effect on such paragraphs. *end guidance*]

3. Normative References

The following normative documents contain provisions, which, through reference in this text, constitute provisions of this Standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this Standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 2382.1:1993, *Information technology — Vocabulary — Part 1: Fundamental terms*.

ISO/IEC 10646:2003 (all parts), *Information technology — Universal Multiple-Octet Coded Character Set (UCS)*.

4. Definitions

For the purposes of this Standard, the following definitions apply. Other terms are defined where they appear in *italic* type or on the left side of a syntax rule. Terms explicitly defined in this Standard are not to be presumed to refer implicitly to similar terms defined elsewhere. [*Note*: This part uses OPC-related terms, which are defined in Part 2: "Open Packaging Conventions". *end note*]

application — A consumer or producer.

behavior — External appearance or action.

behavior, implementation-defined — Unspecified behavior where each implementation documents that behavior, thereby promoting predictability and reproducibility within any given implementation. (This term is sometimes called “application-specific behavior”.)

behavior, locale-specific — Behavior that depends on local conventions of nationality, culture, and language.

behavior, unspecified —Behavior where this Standard imposes no requirements. [*Note*: To add an extension, an implementer must use the extensibility mechanisms described by this Standard rather than trying to do so by giving meaning to otherwise unspecified behavior. *end note*]

document type — One of the three types of Office Open XML documents: Wordprocessing, Spreadsheet, and Presentation, defined as follows:

- A document whose package-relationship item contains a relationship to a Main Document part (§11.3.10) is a document of type Wordprocessing.
- A document whose package-relationship item contains a relationship to a Workbook part (§12.3.23) is a document of type Spreadsheet.
- A document whose package-relationship item contains a relationship to a Presentation part (§13.3.6) is a document of type Presentation.

An Office Open XML document can contain one or more embedded Office Open XML packages (§15.2.10) with each embedded package having any of the three document types. However, the presence of these embedded packages does not change the type of the document.

DrawingML — A set of conventions for specifying the location and appearance of drawing elements in an Office Open XML document.

extension — Any XML element or attribute not explicitly included in this Standard, but that uses the extensibility mechanisms described by this Standard.

Office Open XML document — A package containing ZIP items as required by, and satisfying Parts 1 and 4 of, this Standard. A rendition of a data stream formatted using the wordprocessing, spreadsheet, or presentation ML and its related MLs as described in this Standard. Such a document is represented as a package.

package— A ZIP archive that conforms to the Open Packaging Conventions specification defined in Part 2 of this Standard.

package, embedded— A package that has been stored as the target of a valid Embedded Package relationship (§15.2.10) in an Office Open XML document

PresentationML — A set of conventions for representing an Office Open XML document of type Presentation.

relationship —The kind of connection between a source part and a target part in a package. Relationships make the connections between parts directly discoverable without looking at the content in the parts, and without altering the parts themselves. (See also Package Relationships.)

relationships part — A part containing an XML representation of relationships.

relationship, explicit — A relationship in which a resource is referenced from a source part's XML using the Id attribute of a Relationship tag.

relationship, implicit — A relationship that is not explicit.

SpreadsheetML — A set of conventions for representing an Office Open XML document of type Spreadsheet.

WordprocessingML — A set of conventions for representing an Office Open XML document of type Wordprocessing.

5. Notational Conventions

The following typographical conventions are used in this Standard:

1. The first occurrence of a new term is written in italics. [*Example*: … is considered *normative*. *end example*]
2. A term defined as a basic definition is written in bold. [*Example*: **behavior** — External … *end example*]
3. The name of an XML element is written using an Element style. [*Example*: The root element is document. *end example*]
4. The name of an XML element attribute is written using an Attribute style. [*Example*: … an id attribute. *end example*]
5. An XML element attribute value is written using a constant-width style. [*Example*: … value of `CommentReference`. *end example*]
6. An XML element type name is written using a Type style. [*Example*: … as values of the xsd:anyURI data type. *end example*]

6. Acronyms and Abbreviations

This clause is informative

The following acronyms and abbreviations are used throughout this Standard:

IEC — the International Electrotechnical Commission

ISO — the International Organization for Standardization

W3C — World Wide Web Consortium

End of informative text

7. General Description

This Standard is intended for use by implementers, academics, and application programmers. As such, it contains a considerable amount of explanatory material that, strictly speaking, is not necessary in a formal specification.

This Part is divided into the following subdivisions:

1. Front matter (clauses 1–7);
2. Overview (clause 8);
3. Main body (clauses 9–14);
4. Annexes

Examples are provided to illustrate possible forms of the constructions described. References are used to refer to related clauses. Notes are provided to give advice or guidance to implementers or programmers. Rationale provides explanatory material as to why something is or is not in this Standard. Annexes provide additional information or summarize the information contained in this Standard.

Clauses 1–5, 7, and 9–14 form a normative part of this Part; and the Introduction, clauses 6 and 8, as well as the annexes, notes, examples, rationale, guidance, and the index, are informative.

Except for whole clauses or annexes that are identified as being informative, informative text that is contained within normative text is indicated in the following ways:

1. [*Example:* code fragment, possibly with some narrative … *end example*]
2. [*Note:* narrative … *end note*]
3. [*Rationale:* narrative … *end rationale*]
4. [*Guidance*: narrative … *end guidance*]

8. Overview

This clause is informative.

This clause contains an overview of Office Open XML.

8.1 Packages and Parts

An Office Open XML document is represented as a series of related *parts* that are stored in a container called a *package*. Information about the *relationships* between a package and its parts is stored in the package's *package-relationship ZIP item*. Information about the *relationships* between two parts is stored in the *part-relationship ZIP item* for the source part. A package is an ordinary ZIP archive, which contains that package's content-type item, relationship items, and parts. (Packages are discussed further in Part 2.)

A WordprocessingML document contains a part for the body of the text; it might also contain a part for an image referenced by that text, and parts defining document characteristics, styles, and fonts. A SpreadsheetML document contains a separate part for each worksheet; it might also contain parts for images. A PresentationML document contains a separate part for each slide.

8.2 Consumers and Producers

A tool that can read and understand a package is called a *consumer*, while one that can create a package is called a *producer*. An application can be a consumer, a producer, or both. For example, when a word processor creates a new document, it acts as a producer. When it is used to open an existing document for reading or search purposes, it acts as a consumer. When it is used to open an existing document, edit it, and save the result, it acts as both consumer and producer. Similar scenarios exist for spreadsheet and presentation applications.

8.3 WordprocessingML

This subclause introduces the overall form of a WordprocessingML package, and identifies some of its main element types. (See Part 3 for a more detailed introduction.)

A WordprocessingML package has a relationship of type officeDocument, which specifies the location of the main part in the package. For a WordprocessingML document, that part contains the main text of the document.

A WordprocessingML package's main part starts with a word processing root element. That element contains a *body*, which, in turn, contains one or more *paragraphs* (as well as tables, pictures, and the like). A paragraph contains one or more runs, where a *run* is a container for one or more pieces of *text* having the same set of properties. Like many collection element types, each run and paragraph can have associated with it a set of *properties*. For example, a run might have the property bold, which indicates that run's text is to be displayed in a bold typeface.

A WordprocessingML document is organized into *sections*, and the layout of a page on which the text appears within a section is controlled by that section's properties. For example, each section can have its own *headers* and *footers*.

One relationship from the document part specifies the document's styles. A *style* defines a text display format. A style can have properties, which can be applied to individual paragraphs or runs. Styles make runs more compact by reducing the number of repeated definitions and properties, and the amount of work required to make changes to the document's appearance. With styles, the appearance of all the pieces of text that share a common style can be changed in one place, in that style's definition.

A series of paragraphs can have *numbering* applied to them via a numbering definition instance or a numbering style.

Data in a WordprocessingML document can be organized in a *table*, a two-dimensional grid of *cells* organized into *rows* and *columns*. Cells and whole tables can have associated properties. A cell can contain text and paragraphs, for example.

Text within a WordprocessingMLdocument can be determined dynamically via the use of *fields*. Fields consist of *field instructions* (the text that dictates the field's dynamic behavior) and the *field result* (the text resulting from the dynamic calculation of the field instructions. For example, page numbers are represented as fields. A *hyperlink* consists of two pieces of information: the hyperlink itself—the text the user will click—and the target for the link. Potential targets include external files, e-mail addresses, web sites, and bookmarks within the document itself.

A WordprocessingML document can also contain *custom markup*, user-defined semantics applied to arbitrary document content.

A WordprocessingML document is not stored as one large body in a single part; instead, the elements that implement certain groupings of functionality are stored in separate parts. For example, all footnotes in a document are stored in one footnote part, while each section can have up to three different header parts and three different footer parts, to support headers and footers on odd-numbered pages, even-numbered pages, and the first page.

8.4 SpreadsheetML

This subclause introduces the overall form of a SpreadsheetML package, and identifies some of its main element types. (See Part 3 for a more detailed introduction.)

A SpreadsheetML package has a relationship of type officeDocument, which specifies the location of the main part in the package. For a SpreadsheetML document, that part contains the workbook definition.

A SpreadsheetML package's main part starts with a spreadsheet root element. That element is a *workbook*, which refers to one or more *worksheets*, which, in turn, contain the data. A worksheet is a two-dimensional grid of *cells* that are organized into *rows* and *columns*.

The cell is the primary place in which data is stored and operated on. A cell can have a number of characteristics, such as numeric, text, date, or time *formatting*; *alignment*; *font*; *color*; and a *border*. Each cell is identified by a *cell reference*, a combination of its column and row headings.

Each horizontal set of cells in a worksheet is called a *row*, and each row has a heading numbered sequentially, starting at 1. Each vertical set of cells in a worksheet is called a *column*, and each column has an alphabetic heading named sequentially from A–Z, then AA–AZ, BA–BZ, and so on.

Instead of data, a cell can contain a *formula*, which is a recipe for calculating a value. Some formulas—called *functions*—are predefined, while others are user-defined. Examples of predefined formulas are `AVERAGE`, `MAX`, `MIN`, and `SUM`. A function takes one or more arguments on which it operates, producing a result. For example, in the formula `SUM(B1:B4)`, there is one argument, `B1:B4`, which is the range of cells B1–B4, inclusive.

Other features that a SpreadsheetML document can contain include the following: *comments*, *hyperlinks*, *images*, and sorted and filtered *tables*.

A SpreadsheetML document is not stored as one large body in a single part; instead, the elements that implement certain groupings of functionality are stored in separate parts. For example, all the data for a worksheet is stored in that worksheet's part, all string literals from all worksheets are stored in a single shared string part, and each worksheet having comments has its own comments part.

8.5 PresentationML

This subclause introduces the overall form of a PresentationML package, and identifies some of its main element types. (See Part 3 for a more detailed introduction.)

A PresentationML package has a relationship of type officeDocument, which specifies the location of the main part in the package. For a PresentationML document, that part contains the presentation definition.

A PresentationML package's main part starts with a presentation root element. That element contains a *presentation*, which, in turn, refers to a *slide* list, a *slide master* list, a *notes master* list, and a *handout master* list. The slide list refers to all of the slides in the presentation; the slide master list refers to all of the slide masters used in the presentation; the notes master contains information about the formatting of notes pages; and the handout master describes how a handout looks.

A *handout* is a printed set of slides that can be handed out to an *audience* for future reference.

As well as text and graphics, each slide can contain *comments* and *notes*, can have a *layout*, and can be part of one or more *custom presentations*. (A comment is an annotation intended for the person maintaining the presentation slide deck. A note is a reminder or piece of text intended for the presenter or the audience.)

Other features that a PresentationML document can contain include the following: *animation*, *audio*, *video*, and *transitions* between slides.

A PresentationML document is not stored as one large body in a single part; instead, the elements that implement certain groupings of functionality are stored in separate parts. For example, all comments in a document are stored in one comment part while each slide has its own part.

8.6 Supporting MLs

This subclause introduces the set of markup languages used across package types. (See Part 3 for a more detailed introduction.)

The three markup languages described above define the structure of a package that is either a document (WordprocessingML), a spreadsheet (SpreadsheetML), or a presentation (PresentationML). However, there is also a set of shared markup languages used for common elements such as charts, diagrams, and drawing objects. These MLs are discussed below.

8.6.1 DrawingML

DrawingML specifies the location and appearance of drawing elements in a package. For example, these elements could be, but are not limited to, shapes, pictures, and tables. The root element of a DrawingML XML fragment specifies the presence of a drawing at this location in the document.

A *shape* is a geometric object such as a circle, square, or rectangle; a *picture* is an image presented inside the document; and a *table* is a two-dimensional grid of *cells* organized into *rows* and *columns*. Cells and whole tables can have associated properties. A cell can contain text, for example.

DrawingML also specifies the location and appearance of charts in a package. The root element of a chart part is chart, and specifies the appearance of the chart at this location in the document.

In addition, DrawingML specifies package-wide appearance characteristics, such as the package's theme. The *theme* of a document specifies the *color scheme*, *fonts*, and *effects*, which can be referenced by parts of the document—such as text, drawings, charts, and diagrams—in order to create a consistent visual presentation.

A *chart* is a presentation of data in a graphical fashion, such as a pie chart, bar chart, line chart, in order to make trends and exceptions in the data more visually apparent.

DrawingML also specifies the location and appearance of diagrams in a document. Together, the following four parts define a diagram:

- The *data* part (§14.2.4) specifies individual items of information presented in the diagram. Typically, each piece is a simple line of text, but depending on the diagram, an item of data might also be an image.
- The *layout* part (§14.2.5) specifies how the data and shapes are laid out to create the resulting diagram.
- The *colors* part (§14.2.3) specifies the color which is applied to each individual shape in the diagram.
- The *styles* part (§14.2.6) defines how each individual shape in the diagram maps to the document's theme.

8.6.2 VML

VML specifies the appearance and content of certain shapes in a document. This is used for shapes such as text boxes, as well as shapes which must be stored to maintain compatibility with earlier versions of consumer/producer applications.

[*Note*: The VML format is a legacy format originally introduced with Office 2000 and is included and fully defined in this Standard for backwards compatibility reasons. The DrawingML format is a newer and richer format created with the goal of eventually replacing any uses of VML in the Office Open XML formats. VML should be considered a deprecated format included in Office Open XML for legacy reasons only and new applications that need a file format for drawings are strongly encouraged to use preferentially DrawingML. *end note*]

A shape definition is typically specified using two elements: shapeData, which stores information about the shape, and shape, which stores the shape definition and appearance directly.

8.6.3 Custom XML Data Properties

Custom XML Data properties allow the ability to store arbitrary XML in a package, along with schema information used by that XML.

8.6.4 File Properties

The *core file properties* of a package enable users to discover, get, and set common sets of properties from within that package, regardless of whether it's a WordprocessingML, SpreadsheetML, or PresentationML package, or another use of OPC. Such properties include creator name, creation date, title, and description.

Extended file properties are specific to Office Open XML packages. For example, for a WordprocessingML package, these properties include the number of characters, words, lines, paragraphs, and pages in the document. For a SpreadsheetML package, these properties include worksheet titles. For a PresentationML package, these properties include presentation format, the number of slides, the number of notes, and whether or not any slides are hidden.

Custom file properties are defined by the user. Examples include the name of the client for whom the document was prepared, a date/time on which some event happened, a document number, or some Boolean status flag. Each custom file property has a value, and that value has a type.

8.6.5 Math

Math specifies the structure and appearance of equations in a document; it is specified with a root element of math.

8.6.6 Bibliography

Bibliography specifies the structure for all references stored within a document, for use in citations or a bibliography.

End of informative text.

9. Packages

An Office Open XML document is stored as a package, whose format is defined by Part 2: "Open Packaging Conventions". This subclause contains information regarding Office Open XML's use of OPC.

Throughout this Standard, the Open Packaging Conventions are referred to by their abbreviated name, OPC.

9.1 Constraints on Office Open XML's Use of OPC

While the OPC specification is designed for the representation of Office Open XML documents, it could also support a much broader range of applications. As a result, the use of some OPC features is restricted within Office Open XML documents. These additional requirements are discussed in the following subordinate subclauses. Any requirement not mentioned here is inherited from the OPC specification.

9.1.1 Part Names

An Office Open XML part name shall contain only ASCII characters, in non-escaped or escaped form. The following ASCII characters are permitted in non-escaped form: "!", "$", "%", "&", "'", "(", ")", "*", "+", ",", "-", ".", the decimal digits "0"–"9", ":", ";", "=", "@", the Latin alphabetic characters "A"–"Z" and "a"–"z", "_", and "~". All other ASCII characters are permitted only when escaped as an encoded triplet of the form "%HH", where H is a hexadecimal digit.]

9.1.2 Part Addressing

Parts in an Office Open XML package targeted by relationships are addressed in relationship markup through part names. External document resources targeted by a relationship can be addressed using both relative and absolute references.

9.1.3 Fragments

Fragment identifiers are supported as part of all Office Open XML external relationship targets and some Office Open XML internal relationship targets.

9.1.4 Physical Packages

Each Office Open XML document is implemented as a ZIP archive.

9.1.5 Interleaving

Office Open XML document parts shall be arranged contiguously in a package as defined by simple ordering.

Parts within an Office Open XML package shall not be interleaved. All parts shall be stored as complete ZIP items, and the interleaving functionality defined in Part 2 of this Standard shall not be used. [*Note*: Part 2 of this Standard specifies a method for interleaving parts, which is a very useful capability for stream processing. In order to simplify initial implementations of the Standard, interleaving is not used in this current version of

the Office Open XML formats but it may be used in further versions of the standard or by other formats that leverage OPC. *end note*]

9.1.6 Unknown Parts

With the exception of relationship parts, all other parts in an Office Open XML document that are not the target of a valid relationship are considered *unknown parts*. Unknown parts shall be ignored on document consumption and can, but need not, be discarded on production.

9.1.7 Trash Items

Trash items represent parts that have been discarded or are no longer in use. Trash items shall not conform to OPC part naming guidelines as defined in Part 2 and shall not be associated with a content type. All trash items shall follow the naming scheme: `[trash]/hhhh.dat` where h represents a hexadecimal value.

[*Example*: A package has two parts that must be updated in-place but both parts have grown beyond their growth hints. The newer updated parts are added as new ZIP items while the original parts are renamed to:

```
[trash]/0000.dat
[trash]/0001.dat
```

end example]

9.1.8 Invalid Parts

ZIP archive items that do not conform to OPC part naming guidelines or are not associated with a content type shall not be allowed in an Office Open XML document, with the exception of items specifically defined by Part 2: “Open Packaging Conventions” and trash items.

9.1.9 Unknown Relationships

All relationships not defined within this Standard are considered *unknown relationships*. Unknown relationships are valid within an Office Open XML document provided that they conform to relationship markup guidelines as defined by the OPC specification. Specifically:

- Conforming consumers shall not fail to load a document containing unknown relationships.
- Conforming producers are encouraged to roundtrip and preserve unknown relationships and their target parts.

9.2 Relationships in Office Open XML

In OPC, relationships describe references from parts to other internal resources in the package or to external resources. They represent the type of connection between a source part and a target resource, and make the connection directly discoverable without looking at the part contents, so they are quick to resolve.

The same ZIP item can be the target of multiple relationships. [*Note*: Having multiple paths to a target can make access to that target more convenient. *end note*]

Office Open XML imposes constraints on relationships, described in subsequent clauses of this part. Relationships in Office Open XML are either explicit or implicit.

For an explicit relationship, a resource is referenced from a source part's XML using the Id attribute of a Relationship tag. [*Example*: A document part can have a relationship to a hyperlink only if that hyperlink's Relationship element's Id attribute value is referenced explicitly by the document part's XML. *end example*] [*Note*: Because this mechanism is used generically across multiple tag types, explicit relationships can be extracted from an Office Open XML document without prior knowledge of tag semantics. *end note*]. Certain relationships shall be *explicit*.. All other relationships are *implicit* [*Note*: The syntax for specifying an implicit relationship varies among tag types. *end note*]. Relationships that are required or permitted, and restrictions on those relationships are described in §10–15 of this Part.

[*Example*: Consider a WordprocessingML document that contains the following footnote sentence fragment, "… produced by Ecma[1] (http://www.ecma-international.org/).", which contains a footnote and a hyperlink to a web site. The relationship from a source to a footnote is implicit while that to a hyperlink is explicit.

The Main Document part's relationship file contains the following:

```
<Relationships …>
  <Relationship Id="rId5" Type="…/footnotes"
    Target="footnotes.xml"/>
  <Relationship Id="rId7" Type="…/hyperlink"
    Target="http://www.ecma-international.org/" TargetMode="External"/>
</Relationships>
```

All footnotes for a WordprocessingML document are contained in the same Footnotes part. Let's look at how the Main Document refers to the footnote. At the point at which the footnote reference is inserted, the following XML is present:

```
<w:r>
      <w:footnoteReference w:id="2"/>
</w:r>
```

The w:id="2" refers to the footnote with id=2 in the Footnotes part, the relevant piece of which is:

```
<w:footnote w:id="2">
  …
  Ecma is an international standards development organization (SDO).
  …
</w:footnote>
```

In the case of the hyperlink, the main document part makes an explicit reference to this relationship when it refers to the hyperlink, by using the following:

```
<w:hyperlink r:id="rId7" w:history="1">
  …
</w:hyperlink>
```

The important distinction here is that there is no explicit reference to a relationship ID designating the Footnotes part. The reference to the footnote with `id=2` is “understood” to be in the Footnotes part that must always exist if there are any footnotes in the document. *end example*]

[*Example*: The following figure shows how the source, relationship item, and the target relate to each other for implicit and explicit relationships, respectively. The target does not have to be a file, however.

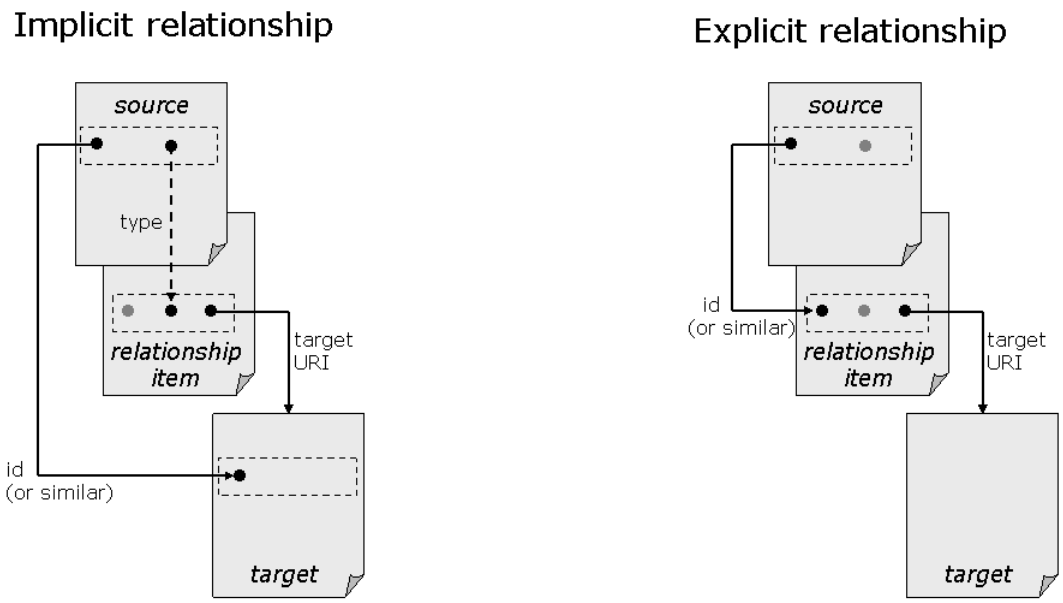
The dots correspond to attributes of relevant elements. Where one attribute refers to a piece in another part, this is indicated by arrows. Solid arrows indicate that the value of the source directly corresponds to the value at the target (for instance, id=rId4 in the source part corresponds to id=rId4 in the relationship item).

Dotted arrows indicate that the value of the source only implicit corresponds to the value of the target (for instance, "footnoteReference" in the source indicates the type "footnotes" in the relationship item).

The main difference between the two types of relationship is that for implicit relationships, the id of the reference refers to an element with the same id in the target part, whereas for explicit relationships, the id refers to a relationship with the same id in the relationship item.

Both relationship types use the target URI of the relationship in the relationship item to locate the target.

For explicit relationships, the id in the source XML links to a relationship item with a direct explicit reference to the target. For implicit relationships, the relationship item is implied by the containing tag (e.g., footnote) and the id in the source XML is used to locate the correct element within the implied target.

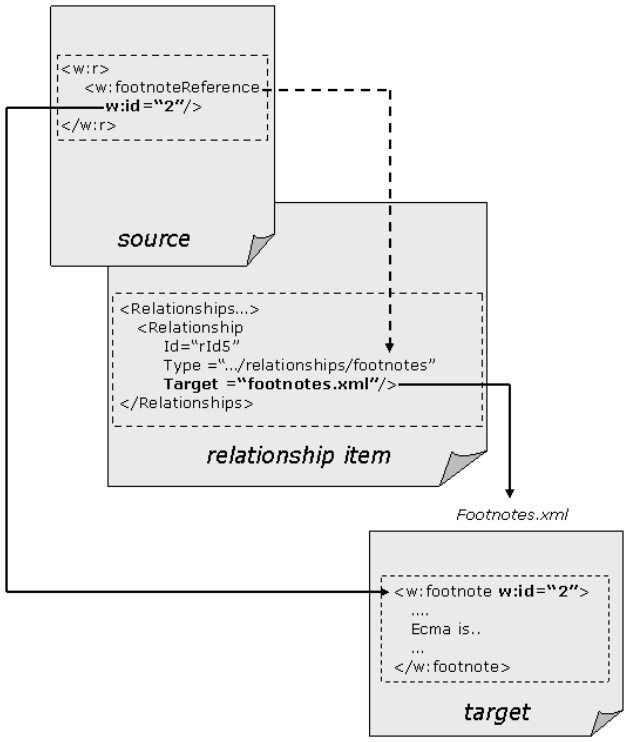


1

2 *end example*]

3 [*Example*: The following figure shows the implicit relationship for the footnote example described earlier.

Example Implicit relationship

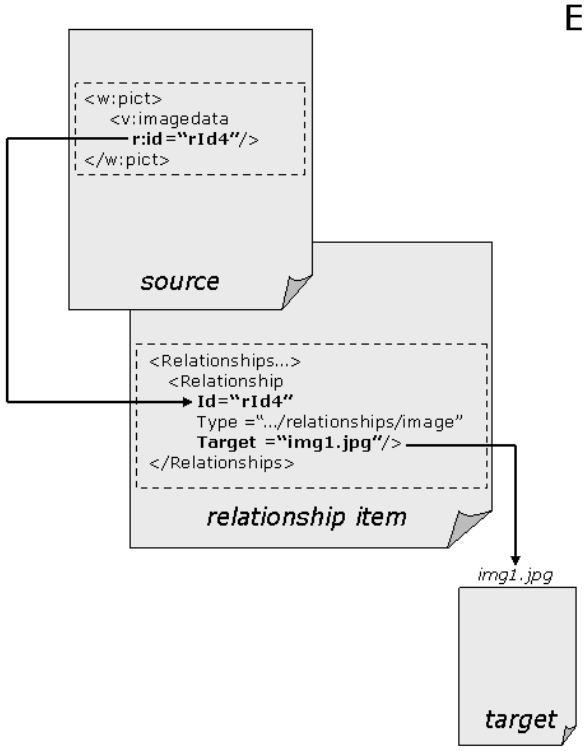


1

2 *end example*]

3 [*Example*: The following figure shows an explicit relationship.

Example Explicit relationship



1

2 *end example*]

10. Markup Compatibility and Extensibility

Office Open XML documents are designed to allow for innovation by extending their capabilities via a scheme defined by Part 5: "Markup Compatibility and Extensibility". This subclause contains information regarding Office Open XML's use of the Markup Compatibility constructs.

10.1 Constraints on Office Open XML's Use of Markup Compatibility and Extensibility

While the Markup Compatibility and Extensibility specification is designed for and used by Office Open XML documents, it could also be used to support a much broader range of applications. As a result, the use of some Markup Compatibility and Extensibility features is restricted within Office Open XML documents. These additional requirements are discussed in the following subordinate subclauses. Any requirement not mentioned here is inherited from the Markup Compatibility and Extensibility specification.

10.1.1 PreserveElements and PreserveAttributes

The PreserveElements and PreserveAttributes elements, as defined in Part 5, allow a markup language to specify the conditions under which extensions should be round-tripped, even when their contents are edited. Within the context of the markup languages explicitly defined by this Standard, no such conditions are specified, and therefore applications are not obliged to support these hints at any point in an Office Open XML document. Instead, the well-defined extensibility constructs defined below should be used.

All other constructs defined in Part 5 shall be supported.

10.1.2 Office Open XML Native Extensibility Constructs

Clause 12 of Part 5 specifies the ability for a markup language to define additional constructs for extensibility of a specific markup language. Within the context of Office Open XML documents, the extLst element(s) defined in individual markup languages shall allow the round-tripping of all unknown content regardless of the state of the PreserveElements and PreserveAttributes elements. See Part 4 for additional information on the valid use of the extLst construct.

11. WordprocessingML

This clause contains specifications for relationship items and parts that are specific to WordprocessingML. Parts that can occur in a WordprocessingML document, but are not WordprocessingML-specific, are specified in §15.2. Unless stated explicitly, all references to relationship items, content-type items, and parts in this clause refer to WordprocessingML ZIP items.

11.1 Glossary of WordprocessingML-Specific Terms

The following terms are used in the context of a WordprocessingML document:

comment — A note that an author or reviewer attaches to a piece of text in a document. Although a consumer may chose to display comments, they are not considered part of the body of the document. A comment includes the text of the note, the comment author's name and initials, and date of creation, among other things.

document setting — A reusable element in a template. [*Note*: Such elements include boilerplate text, cover pages, equations, footers, headers, tables, text boxes, and watermarks. *end note*]

glossary document — An additional WordprocessingML document story used to store reusable fragments of rich WordprocessingML content. It is called the glossary document as this story contains one or more fragments that can be indexed and extracted by name, like items in a glossary.

master document — A document that is the parent of one or more subdocuments. [*Note*: A master document can be used to manage a multipart document, such as a book having several chapters. In such as case, the master document might contain the cover page, front matter, table of contents, and cross-reference index, while each chapter and appendix resides in its own subdocument. *end note*]

section — A portion of a document in which certain page formatting options can be set. [*Note*: A new section is created to change such properties as line numbering, number of columns, or headers and footers. *end note*]

subdocument — A piece of a master document. [*Note*: A chapter or appendix might be a subdocument in a book. *end note*]

supplementary document storage location — A part within a WordprocessingML document in which fragments of WordprocessingML content can be stored separate from the printed page. See also **glossary document**

template — A document that is a pattern for creating other documents. A template can contain text, formatting, and graphics, among other things, such that documents based on it automatically have access to these elements.

11.2 Package Structure

A WordprocessingML package shall contain a package-relationship item and a content-type item. The package-relationship item shall contain implicit relationships with targets of the following type:

- One Main Document part (§11.3.10)

The package-relationship item is permitted to contain implicit relationships with targets of the following type:

- Digital Signature Origin (§15.2.6)
- File Property parts (§15.2.11) (Application-Defined File Properties, Core File Properties, and Custom File Properties), as appropriate.
- Thumbnail (§15.2.14).

The required and optional relationships between parts are defined in §11.3 and its subordinate clauses.

[*Example*: The following package represents the minimal conformant WordprocessingML package as defined by this Standard:

First, the content type for relationship parts and the Main Document part (the only required part) must be defined (physically located at /[Content_Types].xml in the package):

```
<Types xmlns="…">
  <Default Extension="rels"
    ContentType="application/vnd.openxmlformats-
      package.relationships+xml"/>
  <Override PartName="/document.xml"
    ContentType="application/vnd.openxmlformats-
      officedocument.wordprocessingml.document.main+xml"/>
</Types>
```

Next, the single required relationship (the package-level relationship to the Main Document part) must be defined (physically located at /_rels/.rels in the package):

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://schemas.openxmlformats.org/officeDocument/2006/
      relationships/officeDocument"
    Target="document.xml"/>
</Relationships>
```

Finally, the minimum content for the Main Document part must be defined (physically located at /document.xml in the package):

```
<w:document xmlns:w="…">
  <w:body>
    <w:p/>
  </w:body>
</w:document>
```

end example]

[*Example*: Consider a WordprocessingML document that is an early draft of this Standard. Here's an example of the hierarchical folder structure that might be used for the ZIP items in the package for that document. As shown, one part, Main Document (stored in the ZIP item /word/document.xml), has its own relationship item:

```
/[Content_Types].xml        Content-type item
/_rels/.rels                      Package-relationship item
/docProps/app.xml                 Application-Defined File Properties part
/docProps/core.xml                Core File Properties part
/word/document.xml                Main Document part
/word/_rels/document.xml.rels     Part-relationship item
/word/comments.xml                Comment part
/word/endnotes.xml                Endnotes part
/word/fontTable.xml               Font Table part
/word/footer1.xml                 Footer parts
/word/footer2.xml
/word/footer3.xml
/word/footer4.xml
/word/footnotes.xml               Footnotes part
/word/header1.xml                 Header parts
/word/header2.xml
/word/header3.xml
/word/header4.xml
/word/header5.xml
/word/header6.xml
/word/numbering.xml               Numbering Definitions part
/word/settings.xml                Document Settings part
/word/styles.xml                  Style Definitions part
/word/theme/theme1.xml            Theme part
```

The package-relationship item contains the following:

```
<Relationships xmlns="…">
  <Relationship Id="rId3"
    Type="http://…/extended-properties" Target="docProps/app.xml"/>
  <Relationship Id="rId2"
    Type="http://…/core-properties" Target="docProps/core.xml"/>
```

```
    <Relationship Id="rId1"
      Type="http://…/officeDocument" Target="word/document.xml"/>
  </Relationships>
```

end example]

11.3 Part Summary

The subclauses subordinate to this one describe in detail each of the part types specific to WordprocessingML. [*Note*: For convenience, information from those subclauses is summarized in the following table:

Part	Relationship Target of	Root Element	Ref.
Alternative Format Import	Comments, Endnotes, Footer, Footnotes, Header, or Main Document	Not applicable	§11.3.1
Comments	Glossary Document or Main Document	comments	§11.3.2
Document Settings	Glossary Document or Main Document	settings	§11.3.3
Endnotes	Glossary Document or Main Document	endnotes	§11.3.4
Font Table	Glossary Document or Main Document	fonts	§11.3.5
Footer	Glossary Document or Main Document	ftr	§11.3.6
Footnotes	Glossary Document or Main Document	footnotes	§11.3.7
Glossary Document	Main Document	glossaryDocument	§11.3.8
Header	Glossary Document or Main Document	hdr	§11.3.9
Main Document	WordprocessingML package	document	§11.3.10
Numbering Definitions	Glossary Document or Main Document	numbering	§11.3.11
Style Definitions	Glossary Document or Main Document	styles	§11.3.12
Web Settings	Glossary Document or Main Document	webSettings	§11.3.13

end note]

11.3.1 Alternative Format Import Part

Content Type:	Any content, support for which is application-defined. [*Note*: Some examples of formats which might be supported include: • Text = application/txt • RTF = application/rtf • HTML = application/html • XML = application/xml *end note*]
Root Namespace:	not applicable
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/aFChunk

An alternative format import part allows content specified in an alternate format (HTML, MHTML, RTF, earlier versions of WordprocessingML, or plain text) to be embedded directly in a WordprocessingML document in order to allow that content to be migrated to the WordprocessingML format.

Any document part that permits a p element can also contain an altChunk element, whose id attribute refers to a relationship. That relationship shall target a part within the package, which contains the content to be imported into this WordprocessingML document.

A package is permitted to contain zero or more Alternative Format Import parts, each of which shall have a corresponding alternate format file that is the target of an explicit relationship from a Comments (§11.3.2), Endnotes (§11.3.4), Footer (§11.3.6), Footnotes (§11.3.7), Header (§11.3.9), or Main Document (§11.3.10) part. This relationship shall be explicitly referenced using its relationship ID in the source part using the appropriate XML syntax (i.e.; in the id attribute on the altChunk element), and the presence of this relationship without such a reference shall be considered invalid.

A WordprocessingML consumer shall treat the contents of such legacy text files as if they were formatted using equivalent WordprocessingML, and if that consumer is also a WordprocessingML producer, it shall emit the legacy text in WordprocessingML format.

This Standard does not specify how one might create a WordprocessingML package that contains Alternative Format Import relationships and altChunk elements.

However, a conforming producer shall not create a WordprocessingML package that contains Alternative Format Import relationships and elements. [*Note*: The Alternative Format Import machinery provides a one-time conversion facility. A producer could have an extension that allows it to generate a package containing these relationships and elements, yet when run in conforming mode, does not do so. *end note*]

[*Example*: The following Main Document part-relationship item contains a relationship to an Alternative Format Import part:

```
<Relationships xmlns="…">
  <Relationship Id="rId5"
    Type="http://…/aFChunk" Target="Demo.html"
    TargetMode="Internal"/>
</Relationships>
```

The Main Document part contains the following XML fragment:

```
<w:body>
  …
  <w:p/>
  <w:altChunk r:id="rId5"/>
  <w:p/>
  …
</w:body>
```

which results in the entire contents of Demo.html being converted and brought into the document at that point (assuming that the content type of Demo.html is supported by the application consuming this WordprocessingML file). *end example*]

An Alternative Format Import part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

An Alternative Format Import part shall not have any explicit or implicit relationships to parts defined by this Standard.

11.3.2 Comments Part

Content Type:	application/vnd.openxmlformats-officedocument.wordprocessingml.comments+xml
Root Namespace:	http://schemas.openxmlformats.org/wordprocessingml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/comments

An instance of this part type contains the information about each comment in the document.

A package shall contain no more than two Comments parts. If it exists, one instance of that part shall be the target of an implicit relationship from the Main Document (§11.3.10) part, and the other shall be the target of an implicit relationship from the Glossary Document (§11.3.8) part.

[*Example*: The following Main Document part-relationship item contains a relationship to the Contents part, which is stored as the ZIP item comment.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId93"
    Type="http://…/comments" Target="comments.xml"/>
</Relationships>
```

end example]

The root element for a Comment part shall be comments.

[*Example*:

```
<w:comments … >
  <w:comment>
    …
  </w:comment>
  …
</w:comments>
```

end example]

The XML markup for a comment in a Main Document part uses the commentReference element.

[*Example*: Consider the case in which the Main Document part contains the text "… in the Standard.", and there is an comment inserted immediately after the period:

```
<w:p …>
  …
  <w:r>
    <w:t>... in the Standard.</w:t>
  </w:r>
  <w:r>
    <w:commentReference w:id="1"/>
  </w:r>
</w:p>
```

end example]

Each comment has a corresponding comment element in the Comments part, which contains the text of the comment.

[*Example*: The text of the comment is "This is my comment.":

```
<w:comments xmlns:w="…"
  <w:comment w:id="1">
    <w:p>
      <w:r>
        <w:t>This is my comment.</w:t>
    </w:p>
  </w:comment>
</w:comments>
```

end example]

A Comments part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Comments part is permitted to contain explicit relationships to the following parts defined by this Standard:

- Alternative Format Import (§11.3.1)
- Chart (§14.2.1)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.8)
- Embedded Object (§15.2.9)
- Embedded Package (§15.2.10)
- Hyperlinks (§15.3)
- Images (§15.2.13)
- Video (§15.2.16)

A Comments part shall not have any implicit or explicit relationships to any other part defined by this Standard.

11.3.3 Document Settings Part

Content Type:	application/vnd.openxmlformats-officedocument.wordprocessingml.settings+xml
Root Namespace:	http://schemas.openxmlformats.org/wordprocessingml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/settings

An instance of this part type contains all the document's properties.

A package shall contain no more than two Document Settings parts. If it exists, one instance of that part shall be the target of an implicit relationship from the Main Document (§11.3.10) part, and the other shall be the target of an implicit relationship from the Glossary Document (§11.3.8) part.

[*Example*: The following Main Document part-relationship item contains a relationship to a Document Settings part, which is stored in the ZIP item documentProperties1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
    Type="http://…/settings" Target="settings.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be settings.

[*Example*:

```
<w:settings … >
  …
  <w:defaultTabStop w:val="360"/>
  <w:footnotePr>
    …
  </w:footnotePr>
  <w:endnotePr>
  …
  </w:endnotePr>
  <w:rsids>
  …
  </w:rsids>
  …
</w:settings>
```

end example]

A Document Settings part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Document Settings part is permitted to contain explicit relationships to the following parts defined by this Standard:

- Document Template (§11.4)
- Mail Merge Data Source (§11.7)
- Mail Merge Header Data Source (§11.8)
- XSL Transformation (§11.9)

A Document Settings part shall not have any implicit or explicit relationships to any other part defined by this Standard.

11.3.4 Endnotes Part

Content Type:	application/vnd.openxmlformats-officedocument.wordprocessingml.endnotes+xml
Root Namespace:	http://schemas.openxmlformats.org/wordprocessingml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/endnotes

An instance of this part type contains all the endnotes for the document.

A package shall contain no more than two Endnotes parts. If it exists, one instance of that part shall be the target of an implicit relationship from the Main Document (§11.3.10) part, and the other shall be the target of an implicit relationship from the Glossary Document (§11.3.8) part.

[*Example*: The following Main Document part-relationship item contains a relationship to the Endnotes part, which is stored as the ZIP item endnotes.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId6"
    Type="http://…/endnotes" Target="endnotes.xml"/>
</Relationships>
```

end example]

The root element for an Endnotes part shall be endnotes.

[*Example*:

```
<w:endnotes xmlns:w="…" …>
  <w:endnote …>
    …
  </w:endnote>
  <w:endnote …>
    …
  </w:endnote>
</w:endnotes>
```

end example]

The XML markup for an endnote in a Main Document part uses the endnoteReference element.

[*Example*: Consider the case in which the Main Document part contains the text "… in the Standard.", and there is an endnote inserted immediately after the period:

```
<w:p …>
  …
  <w:r>
    <w:t>... in the Standard.</w:t>
  </w:r>
  <w:r>
    <w:rPr>
      <w:rStyle w:val="EndnoteReference"/>
    </w:rPr>
    <w:endnoteReference w:id="5"/>
  </w:r>
</w:p>
```

end example]

Each endnote has a corresponding endnote element in the Endnotes part, which contains the text of the endnote, and the endnoteRef element.

[*Example*: The text of the endnote is "This can be downloaded from http://www.aabbcc.com/index.html." where "http://www.aabbcc.com/index.html" is marked as a hyperlink:

```
<w:endnotes xmlns:w="…">
  <w:endnote w:id="5">
    <w:p>
      <w:r>
        <w:rPr>
          <w:rStyle w:val="EndnoteReference"/>
        </w:rPr>
        <w:endnoteRef/>
      </w:r>
      <w:r>
        <w:t xml:space="preserve"> This can be downloaded from </w:t>
      </w:r>
      <w:hyperlink r:id="rId2">
        <w:r>
          <w:rPr>
            <w:rStyle w:val="Hyperlink"/>
          </w:rPr>
          <w:t>http://www.aabbcc.com/index.html</w:t>
        </w:r>
      </w:hyperlink>
```

```
          <w:r>
            <w:t>.</w:t>
        </w:p>
      </w:endnote>
    </w:endnotes>
```

end example]

An Endnotes part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

An Endnotes part is permitted to contain explicit relationships to the following parts defined by this Standard:

- Alternative Format Import (§11.3.1)
- Chart (§14.2.1)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.8)
- Embedded Object (§15.2.9)
- Embedded Package (§15.2.10)
- Hyperlinks (§15.3)
- Images (§15.2.13)
- Video (§15.2.16)

An Endnotes part shall not have any implicit or explicit relationships to any other part defined by this Standard.

11.3.5 Font Table Part

Content Type:	application/vnd.openxmlformats-officedocument.wordprocessingml.fontTable+xml
Root Namespace:	http://schemas.openxmlformats.org/wordprocessingml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/fontTable

An instance of this part type contains information about each of the fonts used by content in the document. When a consumer reads a WordprocessingML document, it shall use this information to determine which fonts to use to display the document when the specified fonts are not available on the consumer's system.

A package shall contain no more than two Font Table parts. If it exists, one instance of that part shall be the target of an implicit relationship in the part-relationship item for the Main Document (§11.3.10) part, and the other instance shall be the target of an implicit relationship from the Glossary Document (§11.3.8) part.

[*Example*: The following Main Document part-relationship item contains a relationship to the Font Table part, which is stored as the ZIP item fontTable.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/fontTable" Target="fontTable.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be fonts.

[*Example*:

```
<w:fonts … >
  <w:font w:name="Calibri">
    <w:panose1 w:val="020F0502020204030204"/>
    <w:charset w:val="00"/>
    <w:family w:val="swiss"/>
    <w:pitch w:val="variable"/>
    <w:sig w:usb0="A00002EF" w:usb1="4000207B" w:usb2="00000000"
      w:usb3="00000000" w:csb0="0000009F" w:csb1="00000000"/>
  </w:font>
</w:fonts>
```

end example]

A Font Table part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Font Table part is permitted to contain explicit relationships to the following parts defined by this Standard:

- Fonts (§15.2.12)

A Font Table part shall not have any implicit or explicit relationships to any other part defined by this Standard.

11.3.6 Footer Part

Content Type:	application/vnd.openxmlformats-officedocument.wordprocessingml.footer+xml
Root Namespace:	http://schemas.openxmlformats.org/wordprocessingml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/footer

An instance of this part type contains the information about a footer displayed for one or more sections.

A package is permitted to contain zero or one Footer part for each kind of footer (first page, odd page, or even page) in each section of the document. Each Footer part shall be the target of an explicit relationship in the part-relationship item for the Main Document (§11.3.10) part, or the Glossary Document (§11.3.8) part.

[*Example*: The Main Document part-relationship item contains one relationship, for the odd footer part, which is stored as the ZIP item footer3.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId91"
    Type="http://…/footer" Target="footer3.xml"/>
</Relationships>
```

end example]

The root element for a Footer part type shall be ftr.

[*Example*:

```
<w:ftr xmlns:w="…" …>
  …
</w:ftr>
```

end example]

The XML markup for a footer in a section of a Main Document part involves the footerReference element in that section's sectPr element which explicitly references the relationship for the header.

[*Example*: Consider the case in which a section in the Main Document part contains odd and even headers, and an odd footer:

```
<w:document xmlns:w="…">
  …
  <w:sectPr>
    <w:footerReference w:val="rId89" w:type="default"/>
    <w:footerReference w:val="rId90" w:type="even"/>
    <w:footerReference w:val="rId91" w:type="first"/>
    <w:type w:val="oddPage"/>
    <w:pgSz w:w="11909" w:h="16834" w:code="9"/>
    <w:pgMar w:top="1440" w:right="1152" w:bottom="1440"
      w:left="1152" w:header="720" w:footer="720" w:gutter="0"/>
    <w:lnNumType w:countBy="1"/>
    <w:pgNumType w:numFmt="lowerRoman"/>
    <w:cols w:space="720"/>
  </w:sectPr>
</w:document>
```

end example]

Each footer has a corresponding ftr element in a Footer part, which contains the text of the footer.

[*Example*: Here is the odd footer corresponding to the example above. It has the page number centered and displayed using lowercase Roman numerals (as set by the pgNumType element above):

```
<w:ftr xmlns:w="…">
  <w:p>
    <w:pPr>
      <w:pStyle w:val="Centered"/>
    </w:pPr>
    <w:fldSimple w:instr="PAGE">
      <w:r>
        <w:rPr>
          <w:noProof/>
        </w:rPr>
        <w:t>i</w:t>
      </w:r>
    </w:fldSimple>
  </w:p>
</w:ftr>
```

end example]

A Footer part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Footer part is permitted to have explicit relationships to the following parts defined by this Standard:

- Alternative Format Import (§11.3.1)
- Chart (§14.2.1)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.8)
- Embedded Object (§15.2.9)
- Embedded Package (§15.2.10)
- Hyperlinks (§15.3)
- Images (§15.2.13)
- Video (§15.2.16)

A Footer part shall not have any implicit or explicit relationships to any other part defined by this Standard.

11.3.7 Footnotes Part

Content Type:	application/vnd.openxmlformats-officedocument.wordprocessingml.footnotes+xml
Root Namespace:	http://schemas.openxmlformats.org/wordprocessingml/2006/main

Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/footnotes

An instance of this part type contains all the footnotes for the document.

A package shall contain no more than two Footnotes parts. If it exists, one instance of that part shall be the target of an implicit relationship from the Main Document (§11.3.10) part, and the other shall be the target of an implicit relationship from the Glossary Document (§11.3.8) part.

[*Example*: The Main Document part-relationship item contains a relationship to the Footnotes part, which is stored as the ZIP item footnotes.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId5"
    Type="http://…/footnotes" Target="footnotes.xml"/>
</Relationships>
```

end example]

The root element for a Footnotes part shall be footnotes.

[*Example*:

```
<w:footnotes xmlns:w="…" …>
  <w:footnote …>
    …
  </w:footnote>
  <w:footnote …>
    …
  </w:footnote>
</w:footnotes>
```

end example]

The XML markup for a footnote in a Main Document part involves the footnoteReference element.

[*Example*: Consider the case in which the Main Document part contains the text "… in the Standard.", and there is a footnote inserted immediately after the period:

```
<w:p …>
  …
  <w:r>
    <w:t>... in the Standard.</w:t>
  </w:r>
  <w:r>
    <w:rPr>
      <w:rStyle w:val="FootnoteReference"/>
    </w:rPr>
    <w:footnoteReference w:id="5"/>
  </w:r>
</w:p>
```

end example]

Each footnote has a corresponding footnote element in the Footnotes part, which contains the text of the footnote and the footnoteRef element.

[*Example*: The text of the footnote is "This can be downloaded from http://www.aabbcc.com/index.html." where "http://www.aabbcc.com/index.html" is marked as a hyperlink:

```
<w:footnotes xmlns:w="…"
  <w:footnote w:id="5">
    <w:p>
      <w:r>
        <w:rPr>
          <w:rStyle w:val="FootnoteReference"/>
        </w:rPr>
        <w:footnoteRef/>
      </w:r>
      <w:r>
        <w:t xml:space="preserve">This can be downloaded from </w:t>
      </w:r>
      <w:hyperlink r:id="rId2" w:history="1">
        <w:r>
          <w:rPr>
            <w:rStyle w:val="Hyperlink"/>
          </w:rPr>
          <w:t>http://www.aabbcc.com/index.html</w:t>
        </w:r>
      </w:hyperlink>
```

```
          <w:r>
            <w:t>.</w:t>
          </w:r>
        </w:p>
      </w:footnote>
    </w:footnotes>
```

end example]

A Footnotes part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Footnotes part is permitted to have explicit relationships to the following parts defined by this Standard:

- Alternative Format Import (§11.3.1)
- Chart (§14.2.1)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.8)
- Embedded Object (§15.2.9)
- Embedded Package (§15.2.10)
- Hyperlinks (§15.3)
- Images (§15.2.13)
- Video (§15.2.16)

A Footer part shall not have any implicit or explicit relationships to any other part defined by this Standard.

11.3.8 Glossary Document Part

Content Type:	application/vnd.openxmlformats-officedocument.wordprocessingml.document.glossary+xml
Root Namespace:	http://schemas.openxmlformats.org/wordprocessingml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/glossaryDocument

An instance of this part type is a supplementary document storage location which stores the definition and content for content that shall be carried with the document for future insertion and/or use, but which shall not be visible within the contents of the main document story. [*Example*: A legal contract template might include one or more optional clauses that shall not appear in the document until those clauses are inserted explicitly via a user action. To store these optional clauses until they are inserted, their contents are placed in the glossary document part. *end example*]

[*Note*: This part is intended for storage of optional "document fragments" which are often used to perform document assembly. The use of the word *glossary* is a reference to the fact that each of these entries was

historically referenced by its first word in legacy word processing applications, like definitions of terms in a traditional glossary. *end note*]

The root element for a part of this content type shall be glossaryDocument.

[*Example*: The following part contains two building blocks. The first block is named "rainbow colors", belongs to a category called "Misc", belongs to a gallery called "docParts", and contains the text "The colors … and violet." The details of the second block have been omitted:

```
<w:glossaryDocument xmlns:w="…" >
  <w:docParts>
    <w:docPart>
      <w:docPartPr>
        <w:name w:val="rainbow colors"/>
        <w:style w:val="Normal"/>
        <w:category>
          <w:name w:val="Misc"/>
          <w:gallery w:val="docParts"/>
        </w:category>
      </w:docPartPr>
      <w:docPartBody>
        <w:p>
          <w:r>
            <w:t>The colors of the rainbow are red, orange, yellow,
              green, blue, indigo, and violet.</w:t>
          </w:r>
        </w:p>
      </w:docPartBody>
    </w:docPart>
    <w:docPart>
      …
    </w:docPart>
  </w:docParts>
</w:glossaryDocument>
```

end example]

A package shall contain at most one Glossary Document part, and that part shall be the target of an implicit relationship from the Main Document (§11.3.10) part.

[*Example*: The following Main Document part-relationship item contains a relationship to a Glossary Document part, which is stored in the ZIP item `glossary/document.xml`:

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
    Type="http://…/glossaryDocument" Target="glossary/document.xml"/>
</Relationships>
```

end example]

A Glossary Document part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Glossary Document part is permitted to have implicit relationships to the following parts defined by this Standard:

- Comments (§11.3.2)
- Document Settings (§11.3.3)
- Endnotes (§11.3.4)
- Font Table (§11.3.5)
- Footnotes (§11.3.7)
- Numbering Definitions (§11.3.11)
- Style Definitions (§11.3.11)

A Glossary Document part is permitted to have explicit relationships to the following parts defined by this Standard:

- Alternative Format Import (§11.3.1)
- Chart (§14.2.1)
- Diagrams: Diagram Colors (§14.2.3), Diagram Data (§14.2.4), Diagram Layout Definition (§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.8)
- Embedded Object (§15.2.9)
- Embedded Package (§15.2.10)
- Footer (§11.3.6)
- Header (§11.3.9)
- Hyperlinks (§15.3)
- Images (§15.2.13)
- Printer Settings (§15.2.14)
- Video (§15.2.16)

A Glossary Document part shall not have implicit or explicit relationships to any other part defined by this Standard.

11.3.9 Header Part

Content Type:	application/vnd.openxmlformats-officedocument.wordprocessingml.header+xml

Root Namespace:	http://schemas.openxmlformats.org/wordprocessingml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/header

An instance of this part type contains the information about a header displayed for one or more sections.

A package shall contain zero or one Header part for each kind of header (first page, odd page, or even page) in each section of the document. Each Header part shall be the target of an explicit relationship from the Main Document (§11.3.10) part or the Glossary Document (§11.3.8) part.

[*Example*: The Main Document part-relationship item contains two relationships: one for the even header part (which is stored as the ZIP item header2.xml) and one for the odd header part (which is stored as the ZIP item header3.xml):

```
<Relationships xmlns="…">
  <Relationship Id="rId89" Type="http://…/header" Target="header2.xml"/>
  <Relationship Id="rId90" Type="http://…/header" Target="header3.xml"/>
</Relationships>
```

end example]

The root element for a Header part shall be hdr.

[*Example*:

```
<w:hdr xmlns:w="…" …>
  …
</w:hdr>
```

end example]

The XML markup for a header in a section of a Main Document part involves the headerReference element in that section's sectPr element.

[*Example*: Consider the case in which a section in the Main Document part contains odd and even headers, and an odd footer:

```
<w:body>
  …
  <w:sectPr w:rsidR="00363F31" w:rsidSect="008D4B40">
    <w:headerReference w:val="rId89" w:type="default"/>
    <w:headerReference w:val="rId90" w:type="even"/>
    <w:headerReference w:val="rId91" w:type="first"/>
    <w:type w:val="oddPage"/>
    <w:pgSz w:w="11909" w:h="16834" w:code="9"/>
    <w:pgMar w:top="1440" w:right="1152" w:bottom="1440"
      w:left="1152" w:header="720" w:footer="720" w:gutter="0"/>
    <w:lnNumType w:countBy="1"/>
    <w:pgNumType w:fmt="lowerRoman"/>
    <w:cols w:space="720"/>
  </w:sectPr>
</w:body>
```

end example]

Each header has a corresponding hdr element in a Header part, which contains the text of the header.

[*Example*: Here is the even header corresponding to the examples above:

```
<w:hdr xmlns:w="…">
  <w:p>
    <w:pPr>
      <w:pStyle w:val="Header"/>
    </w:pPr>
    <w:r>
      <w:t>My Test Document</w:t>
    </w:r>
  </w:p>
</w:hdr>
```

Here is the odd header corresponding to the examples above:

```
<w:hdr xmlns:w="…">
  <w:p>
    <w:pPr>
      <w:pStyle w:val="Header"/>
    </w:pPr>
    <w:r>
      <w:tab/>
      <w:t>Table of Contents</w:t>
    </w:r>
  </w:p>
</w:hdr>
```

end example]

A Header part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Header part is permitted to have explicit relationships to the following parts defined by this Standard:

- Alternative Format Import (§11.3.1)
- Chart (§14.2.1)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.8)
- Embedded Object (§15.2.9)
- Embedded Package (§15.2.10)
- Hyperlinks (§15.3).
- Images (§15.2.13)
- Video (§15.2.16)

A Header part shall not have any implicit or explicit relationships to other parts defined by this Standard.

11.3.10 Main Document Part

Content Type(s):	application/vnd.openxmlformats-officedocument.wordprocessingml.main+xml application/vnd.openxmlformats-officedocument.wordprocessingml.template.main+xml
Root Namespace:	http://schemas.openxmlformats.org/wordprocessingml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument

An instance of this part type contains the body of the document.

A package shall contain a Main Document part (§11.3.10) part. The Main Document part shall be the target of a relationship in the package-relationship item.

The root element for a part of this content type shall be document.

[*Example*: Given the following package-relationship item excerpt:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/officeDocument" Target="word/document.xml"/>
</Relationships>
```

/word/document.xml" contains the following:

```
<w:document …>
  <w:body>
    …
  </w:body>
</w:document>
```

end example]

A Main Document part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Main Document part is permitted to have implicit relationships to the following parts defined by this Standard:

- Additional Characteristics (§15.2.1)
- Bibliography (§15.2.3)
- Comments (§11.3.2)
- Custom XML Data Storage (§15.2.4)
- Document Settings (§11.3.3)
- Endnotes (§11.3.4)
- Font Table (§11.3.5)
- Footnotes (§11.3.7)
- Glossary Document (§11.3.8)
- Numbering Definitions (§11.3.11)
- Style Definitions (§11.3.12)
- Theme (§14.2.7)
- Thumbnail (§15.2.14)

A Main Document part is permitted to contain explicit relationships to the following parts defined by this Standard:

- Alternative Format Import (§11.3.1)
- Chart (§14.2.1)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.8)
- Embedded Object (§15.2.9)
- Embedded Package (§15.2.10)
- Footer (§11.3.6)
- Header (§11.3.9)
- Hyperlinks (§15.3)
- Images (§15.2.13)
- Printer Settings (§15.2.14)
- Subdocument (§11.6)
- Video (§15.2.16)

A Main Document shall not have implicit or explicit relationships to any other part defined by this Standard.

11.3.11 Numbering Definitions Part

Content Type:	application/vnd.openxmlformats-officedocument.wordprocessingml.numbering+xml
Root Namespace:	http://schemas.openxmlformats.org/wordprocessingml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/numbering

An instance of this part type contains a definition for the structure of each unique numbering definition in this document.

[*Example:* If a set of paragraphs are added to a document which have a circle bullet at the first level, a square bullet at the second level, and a checkmark bullet at the third level, such as the following:

- First level
  - Second level
    - ✓ Third level

The numbering definition part will contain the definition for each of these levels (their bullet style, indent, etc.) even if the second and third levels are not actually used in the document *end example*]

A package shall contain no more than two Numbering Definitions parts. If they exist, one instance of that part shall be the target of an implicit relationship from the Main Document (§11.3.10) part, and the other shall be the target of an implicit relationship from the Glossary Document (§11.3.8) part.

[*Example*:

```
<Relationships xmlns="…">
  <Relationship Id="rId2"
    Type="http://…/numbering" Target="numbering.xml"/>
</Relationships>
```

end example]

The XML markup for a list usage involves a reference to a numbering definition via the child elements of the numPr element.

[*Example*: Here we have a paragraph set using the style `Text`, followed by a list of things which have the paragraph style `ListBullet`, followed by another paragraph set using the style `Text`:

```
<w:p>
  <w:pPr>
    <w:pStyle w:val="Text"/>
  </w:pPr>
  <w:r>
    <w:t>The kinds of fruit needed are:</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:pStyle w:val="ListBullet"/>
    <w:numPr>
      <w:ilvl w:val="0" />
      <w:numId w:val="5" />
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>Apples</w:t>
  </w:r>
</w:p>
```

```
    <w:p>
      <w:pPr>
        <w:pStyle w:val="ListBullet"/>
        <w:numPr>
          <w:ilvl w:val="0" />
          <w:numId w:val="5" />
        </w:numPr>
      </w:pPr>
      <w:r>
        <w:t>Oranges</w:t>
      </w:r>
    </w:p>
    <w:p>
      <w:pPr>
        <w:pStyle w:val="Text"/>
      </w:pPr>
      <w:r>
        <w:t>Other items may be needed too.</w:t>
      </w:r>
    </w:p>
```

end example]

The root element for a Numbering Definition part shall be numbering, with each numbering definition being defined by an abstractNum element.

[*Example*:

```
    <w:numbering xmlns:w="…">
      <w:abstractNum w:numId="11">
        <w:nsid w:val="394E2425"/>
        <w:multiLevelType w:val="hybridMultilevel"/>
        <w:tmpl w:val="F628E89A"/>
        <w:lvl w:ilvl="0" w:tplc="151C4798">
          <w:start w:val="1"/>
          <w:numFmt w:val="bullet"/>
          <w:pStyle w:val="ListBullet"/>
          <w:lvlText w:val="…"/>
          <w:lvlJc w:val="left"/>
          <w:pPr>
            <w:tabs>
              <w:tab w:val="list" w:pos="720"/>
            </w:tabs>
            <w:ind w:left="720" w:hanging="360"/>
          </w:pPr>
```

```
          <w:rPr>
            <w:rFonts w:ascii="Symbol" w:hAnsi="Symbol" w:hint="default"/>
          </w:rPr>
        </w:lvl>
        …
      </w:abstractNum>
    </w:numbering>
```

end example]

A Numbering Definitions part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Numbering Definitions part is permitted to contain explicit relationships to the following parts defined by this Standard:

- Images (§15.2.13)

A Numbering Definitions part shall not have any implicit or explicit relationships to any other part defined by this Standard.

11.3.12 Style Definitions Part

Content Type:	application/vnd.openxmlformats-officedocument.wordprocessingml.styles+xml
Root Namespace:	http://schemas.openxmlformats.org/wordprocessingml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/styles

An instance of this part type contains the definition for a set of styles used by this document.

A package shall contain at most two Style Definitions parts. One instance of that part shall be the target of an implicit relationship from the Main Document (§11.3.10) part, and the other shall be the target of an implicit relationship in from the Glossary Document (§11.3.8) part.

[*Example*:

```
  <Relationships xmlns="…">
    <Relationship Id="rId3"
      Type="http://…/styles" Target="styles.xml"/>
  </Relationships>
```

end example]

The root element for a Styles Definition part shall be styles, which is a container for one or more style elements.

[*Example*: Here is the style `ListBullet` (which is used in a Main Document Part in §11.3.10):

```
<w:styles xmlns:wx="…" xmlns:w="…" … xml:space="preserve">
  <w:style w:type="paragraph" w:styleId="ListBullet">
    <w:name w:val="List Bullet"/>
    <w:basedOn w:val="Text"/>
    <w:autoRedefine/>
    <w:rsid w:val="00081289"/>
    <w:pPr>
      <w:pStyle w:val="ListBullet"/>
      <w:numPr>
        <w:numId w:val="1"/>
      </w:numPr>
      <w:tabs>
        <w:tab w:val="clear" w:pos="360"/>
      </w:tabs>
      <w:ind w:left="648"/>
    </w:pPr>
  </w:style>
</w:styles>
```

end example]

A Style Definitions part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Style Definitions part shall not have implicit or explicit relationships to any part defined by this Standard.

11.3.13 Web Settings Part

Content Type:	application/vnd.openxmlformats-officedocument.wordprocessingml.webSettings+xml
Root Namespace:	http://schemas.openxmlformats.org/wordprocessingml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/webSettings

An instance of this part type contains the definition for web-specific settings used by this document.

A package shall contain at most two Web Settings parts. One instance of that part shall be the target of an implicit relationship from the Main Document (§11.3.10) part, and the other shall be the target of an implicit relationship from the Glossary Document (§11.3.8) part.

[*Example*:

```
<Relationships xmlns="…">
  <Relationship Id="rId3"
    Type="http://…/webSettings" Target="webSettings.xml"/>
</Relationships>
```

end example]

The root element for a Web Settings part shall be webSettings.

[*Example*:

```
<w:webSettings …>
  <w:frameset>
    …
    <w:frame>
      <w:sz w:val="216" />
      <w:name w:val="Frame2" />
      <w:sourceFileName r:id="rId1" />
    </w:frame>
    <w:frame>
      <w:name w:val="Frame1" />
      <w:sourceFileName r:id="rId2" />
    </w:frame>
  </w:frameset>
</w:webSettings>
```

end example]

A Web Settings part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Web Settings part is permitted to contain explicit relationships to the following parts defined by this Standard:

- Frameset (§11.5)

A Web Settings part shall not contain implicit or explicit relationships to any other part defined by this Standard.

11.4 Document Template

Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/attachedTemplate

A *document template* can be represented by an instance of a WordprocessingML package, and contains styles, numbering definitions, and so on that are made available when documents based on that template are edited.

A WordprocessingML document can refer to another document as its document template, by having a Document Settings part (§11.3.3) that contains an explicit relationship to the file location of the necessary document template using the id attribute on the attachedTemplate element.

[*Example*: Consider a document specifying a document template located at `c:\template.docx`:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/attachedTemplate" Target="file:///c:\template.docx"
    TargetMode="External"/>
</Relationships>
```

The document's Document Settings part contains an attachedTemplate element that explicitly references this relationship:

```
<w:settings … >
  <w:attachedTemplate r:id="rId1"/>
</w:settings>
```

end example]

11.5 Framesets

Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/frame

A *frameset* is a WordprocessingML document which specifies the location and placement of other WordprocessingML documents (which, when used in this context, are referred to as *frames*). A frameset shall be represented by an instance of a WordprocessingML document with a Web Settings part (§11.3.13) whose relationship item targets each of that frameset's frames.

[*Example*: Consider a frameset document having two frames. The frameset's Web Settings part-relationships item contains the following, in which frame1.docx and frame2.docx are packages containing the corresponding frames:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/frame" Target="frame1.docx" TargetMode="External"/>
  <Relationship Id="rId2"
    Type="http://…/frame" Target="frame2.docx" TargetMode="External"/>
</Relationships>
```

The frameset document's Web Settings part contains a frameset element that references its frames:

```
<w:webSettings …>
  <w:frameset>
    …
    <w:frame>
      <w:sz w:val="216" />
      <w:name w:val="Frame2" />
      <w:sourceFileName r:id="rId1" />
    </w:frame>
    <w:frame>
      <w:name w:val="Frame1" />
      <w:sourceFileName r:id="rId2" />
    </w:frame>
  </w:frameset>
</w:webSettings>
```

end example]

A frame shall be represented by an instance of a WordprocessingML package.

A frame shall be located external to the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `External`).

11.6 Master Documents and Subdocuments

Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/subDocument

A master document shall be represented by an instance of a WordprocessingML document whose Main Document (§11.3.10) part targets each of that master document's subdocuments.

[*Rationale*: Sometimes, it is convenient to deal with a document as a collection of pieces, especially when those pieces might be edited by different authors in a collaborative group. Perhaps it simply makes sense to think about a book as a collection of chapters rather than as one big document. The breaking-up of a document into such pieces can be achieved by having a master document with one or more subdocuments. *end rationale*]

[*Example*: Consider a master document, whose three subdocuments are called Start, Middle, and End, respectively. Master's Main Document part has a corresponding relationships part that contains the following, in which Start.docx, Middle.docx, and End.docx are packages containing the corresponding subdocuments:

```
<Relationships xmlns="…">
  <Relationship Id="rId5"
    Type="http://…/subDocument"
    Target="Start.docx" TargetMode="External"/>
```

```
    <Relationship Id="rId6"
      Type="http://…/SubDocument"
      Target="Middle.docx" TargetMode="External"/>
    <Relationship Id="rId7"
      Type="http://…/SubDocument"
      Target="End.docx" TargetMode="External"/>
  </Relationships>
```

The master document's Main Document part contains subDoc elements that reference its subdocuments:

```
  <w:document xmlns:r="…" xmlns:wx="…" …>
    <w:body>
      <w:p …>
        <w:pPr>
          …
        </w:pPr>
      </w:p>
      <w:subDoc r:id="rId5"/>
      …
      <w:subDoc r:id="rId6"/>
      …
      <w:subDoc r:id="rId7"/>
      …
    </w:body>
  </w:document>
```

end example]

A subdocument shall be represented by an instance of a WordprocessingML package.

A subdocument shall be located external to the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `External`).

11.7 Mail Merge Data Source

Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/mailMergeSource

A document that stores information about a mail merge operation is permitted to contain a Document Settings part (§11.3.3) whose relationship item targets the file location of the necessary data source using this relationship.

[*Example*: Consider a document specifying a mail merge whose data source is located at `http://www.openxmlformats.org/data.txt`:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/mailMergeSource"
    Target="http://www.openxmlformats.org/data.txt"
    TargetMode="External"/>
</Relationships>
```

The document's Document Settings part contains a dataSource element that explicitly references this relationship:

```
<w:settings …>
  <w:mailMerge>
    …
    <w:dataSource r:id="rId1" />
    …
  </w:mailMerge>
</w:settings>
```

end example]

A mail merge data source shall be located external to the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `External`).

11.8 Mail Merge Header Data Source

Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/mailMergeHeaderSource

A document that stores information about a mail merge operation is permitted to contain a Document Settings part (§11.3.3) whose relationship item targets the file location of the necessary header data source using this relationship.

[*Example*: Consider a document specifying a mail merge whose header data source is located at `http://www.openxmlformats.org/header.txt`:

```
<Relationships xmlns="…">
  <Relationship Id="rId2"
    Type="http://…/mailMergeHeaderSource"
    Target=http://www.openxmlformats.org/header.txt
    TargetMode="External"/>
</Relationships>
```

The document's Document Settings part contains a headerSource element that explicitly references this relationship:

```
<w:settings …>
  <w:mailMerge>
    …
    <w:headerSource r:id="rId2" />
    …
  </w:mailMerge>
</w:settings>
```

end example]

A mail merge header data source shall be located external to the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `External`).

11.9 XSL Transformation

Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/transform

A document can store information about an XSL Transformation which might be applied on save, by containing a Document Settings part (§11.3.3) whose part relationship item contains an explicit relationship to the file location of the XSL Transformation using this relationship.

[*Example*: Consider a document specifying an XSL Transformation located at `http://www.openxmlformats.org/test.xsl`:

```
<Relationships xmlns="…">
  <Relationship Id="rId8" Type="http://…/transform"
    Target="http://www.openxmlformats.org/test.xsl"
    TargetMode="External"/>
</Relationships>
```

The document's Document Settings part contains a saveThroughXslt element that explicitly references this relationship:

```
<w:settings …>
  …
  <w:saveThroughXslt r:id="rId8" />
  …
</w:settings>
```

end example]

An XSL transformation shall be located external to the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `External`).

12. SpreadsheetML

This clause contains specifications for relationship items and parts that are specific to SpreadsheetML. Parts than can occur in a SpreadsheetML document, but are not SpreadsheetML-specific, are specified in §15.2. Unless stated explicitly, all references to relationship items, content-type items, and parts in this clause refer to SpreadsheetML ZIP items.

12.1 Glossary of SpreadsheetML-Specific Terms

The following terms are used in the context of a SpreadsheetML document:

cell — The location at the intersection of a row and column, in which numeric or textual data or a formula is stored. A cell can have a number of characteristics, such as numeric or text formatting, alignment, font, color, and border.

cell reference — An individual cell's designation using a combination of its column and row headings, as in A13, H19, and BX1200. A **relative cell reference** in a formula automatically changes when the formula is copied down a column or across a row. An **absolute cell reference** is fixed. Absolute references don't change when a formula is copied from one cell to another. A **mixed cell reference** has either an absolute column and a relative row, or an absolute row and a relative column.

chart — A graphical representation of data, as in a bar, column, line, pie chart, for example.

column — Any vertical set of cells in a worksheet. Each column has an alphabetic heading. Columns are named sequentially, going from A–Z, then AA–AZ, BA–BZ, and so on.

comment — A reminder or annotation that can be attached to a cell.

connection — The means by which external data—that is, data stored outside of a workbook (in a database or on a Web server, for example)—can be imported into a worksheet.

formula — A recipe for calculating a value. Some formulas are predefined; others are user-defined.

function — A predefined formula, such as AVERAGE, MAX, MIN, and SUM. A function takes one or more arguments on which it operates, producing a result. [*Note*: In the formula =SUM(B1:B4), there is one argument, B1:B4, which is the range of cells B1–B4, inclusive. *end note*]

pivot table — A kind of table that is used to manage and analyze related data that is stored elsewhere.

row — Any horizontal set of cells in a worksheet. Each row has a numeric heading. Rows are numbered sequentially, starting at 1.

table — A rectangular-shaped set of related rows and columns that can be sorted, filtered, and totaled as a group. Rows in a table can be hidden by applying **autofilters** to one or more columns.

workbook — A collection of worksheets.

worksheet — A two-dimensional grid of cells that are organized into rows and columns.

12.2 Package Structure

A SpreadsheetML package shall contain a package-relationship item and a content-type item. The package-relationship item shall contain implicit relationships with targets of the following type:

- One Workbook part (12.3.23).

The package-relationship item is permitted to contain implicit relationships with targets of the following type:

- Digital Signature Origin (§15.2.6)
- File Property parts (§15.2.11) (Application-Defined File Properties, Core File Properties, and Custom File Properties), as appropriate.
- Thumbnail (§15.2.14).

The required and optional relationships between parts are defined in §12.3 and its subordinate clauses.

[*Example*: The following package represents the minimal conformant SpreadsheetML package as defined by this Standard:

First, the content types for relationship parts, the Workbook part, and at least one Sheet part must be defined (physically located at /[Content_Types].xml in the package):

```
<Types xmlns="…">
  <Default Extension="rels"
   ContentType="application/vnd.openxmlformats-package.relationships+xml"
   />
  <Override PartName="/workbook.xml"
   ContentType="application/vnd.openxmlformats-officedocument.
   spreadsheetml.sheet.main+xml" />
  <Override PartName="/sheet1.xml"
   ContentType="application/vnd.openxmlformats-
   officedocument.spreadsheetml.worksheet+xml" />
</Types>
```

Next, the required package-level relationship to the Workbook part must be defined (physically located at /_rels/.rels in the package):

```
<Relationships xmlns="…">
```

```
    <Relationship Id="rId1"
     Type=http://schemas.openxmlformats.org/officeDocument/2006/
     relationships/officeDocument"
     Target="workbook.xml" />
  </Relationships>
```

Next, the minimum content for the Workbook part must be defined (physically located at /workbook.xml in the package):

```
  <workbook xmlns="…" xmlns:r="…">
    <sheets>
      <sheet name="1" sheetId="1" r:id="rId1" />
    </sheets>
  </workbook>
```

Next, the required workbook-level relationship to the single Sheet part must be defined, (physically located at /_rels/workbook.xml.rels in the package):

```
  <Relationships xmlns="…">
    <Relationship Id="rId1"
     Type="http://schemas.openxmlformats.org/officeDocument/2006/
     relationships/worksheet" Target="sheet1.xml" />
  </Relationships>
```

Finally, the minimum content for a single Sheet part must be defined (physically located at /sheet1.xml in the package):

```
  <worksheet xmlns="…" xmlns:r="…">
    <sheetData />
  </worksheet>
```

end example]

[*Example*: Consider a SpreadsheetML document that contains a workbook having three worksheets. Here's an example of the hierarchical folder structure that might be used for the ZIP items in the package for that document. As shown, one part, Workbook (stored in the ZIP item /xl/workbook.xml), has its own relationship item:

```
  /_rels/.rels                    Package-relationship item
  /[Content_Types].xml            Content-type item
  /docProps/app.xml               Application-Defined File Properties part
  /docProps/core.xml              Core File Properties part
  /xl/workbook.xml                Workbook part
  /xl/_rels/workbook.xml.rels     Part-relationship item
```

```
/xl/calcChain.xml                Calculation Chain part
/xl/sharedStrings.xml            Shared String Table part
/xl/styles.xml                   Styles part
/xl/volatileDependencies.xml     Volatile Dependencies part
/xl/theme/theme1.xml             Theme part
/xl/worksheets/sheet1.xml        Worksheet parts
/xl/worksheets/sheet2.xml
/xl/worksheets/sheet3.xml
```

The package-relationship item contains the following:

```
<Relationships xmlns="…">
  <Relationship Id="rId3"
    Type="http://…/extended-properties" Target="docProps/app.xml"/>
  <Relationship Id="rId2"
    Type="http://…/core-properties" Target="docProps/core.xml"/>
  <Relationship Id="rId1"
    Type="http://…/officeDocument" Target="xl/workbook.xml"/>
</Relationships>
```

end example]

12.3 Part Summary

The subclauses subordinate to this one describe in detail each of the part types specific to SpreadsheetML. [*Note*: For convenience, information from those subclauses is summarized in the following table:

Part	Relationship Target of	Root Element	Ref.
Calculation Chain	Workbook	calcChain	§12.3.1
Chartsheet	Workbook	chartsheet	§12.3.2
Comments	Chartsheet, Dialogsheet, Worksheet	comments	§12.3.3
Connections	Workbook	connections	§12.3.4
Custom Property	Workbook	Not applicable	§12.3.5
Custom XML Mappings	Workbook	mapInfo	§12.3.6
Dialogsheet	Workbook	dialogSheet	§12.3.7
Drawings	Chartsheet, Worksheet	wsDr	§12.3.8
External Workbook References	Workbook	externalReference	§12.3.9
Metadata	Workbook	metadata	§12.3.10
Pivot Table	Worksheet	pivotTableDefinition	§12.3.11

Part	Relationship Target of	Root Element	Ref.
Pivot Table Cache Definition	Pivot Table, Workbook	pivotCacheDefinition	§12.3.12
Pivot Table Cache Records	Pivot Table Cache Definition	pivotCacheRecords	§12.3.13
Query Table	Worksheet	queryTable	§12.3.14
Shared String Table	Workbook	sst	§12.3.15
Shared Workbook Revision Headers	Workbook	headers	§12.3.16
Shared Workbook Revision Log	Shared Workbook Revision Headers	revisions	§12.3.17
Shared Workbook User Data	Workbook	users	§12.3.18
Single Cell Table Definitions	Dialogsheet, Worksheet	singleCells	§12.3.19
Styles	Workbook	styleSheet	§12.3.20
Table Definition	Dialogsheet, Worksheet	table	§12.3.21
Volatile Dependencies	Workbook	volTypes	§12.3.22
Workbook	SpreadsheetML package	workbook	§12.3.23
Worksheet	Workbook	worksheet	§12.3.24

end note]

12.3.1 Calculation Chain Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.calcChain+xml
Root Namespace:	http://schemas.openxmlformats.org/spreadsheetml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/calcChain

An instance of this part type contains an ordered set of references to all cells in all worksheets in the workbook whose value is calculated from any formula. The ordering allows inter-related cell formulas to be calculated in the correct order when a worksheet is loaded for use.

A package shall contain no more than one Calculation Chain part. If it exists, that part shall be the target of an implicit relationship from the Workbook part (§12.3.23).

[*Example*: The following Workbook part-relationship item contains a relationship to the Calculation Chain part, which is stored in the ZIP item calcChain.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId7"
    Type="http://…/calcChain" Target="calcChain.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be calcChain.

[*Example*: Cells D8, E8, and F8 each contain a value that is the result of calculations that shall be performed in the order E8, D8, F8:

```
<calcChain xmlns="…">
  <c r="E8" i="1"/>
  <c r="D8"/>
  <c r="F8" s="1"/>
</calcChain>
```

end example]

A Calculation Chain part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Calculation Chain part shall not have implicit or explicit relationships to any part defined by this Standard.

12.3.2 Chartsheet Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.chartsheet+xml
Root Namespace:	http://schemas.openxmlformats.org/spreadsheetml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/chartsheet

An instance of this part type represents a chart that is stored in its own sheet.

A package is permitted to contain zero or more Chartsheet parts. Each such part shall be the target of an explicit relationship from the Workbook part (§12.3.23).

[*Example*: The following Workbook part-relationship item contains three relationships to Chartsheet parts, which are stored in the ZIP items chartsheets/sheet*N*.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId2"
    Type="http://…/chartsheet" Target="chartsheets/sheet1.xml"/>
  <Relationship Id="rId5"
    Type="http://…/chartsheet" Target="chartsheets/sheet2.xml"/>
  <Relationship Id="rId6"
    Type="http://…/chartsheet" Target="chartsheets/sheet3.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be chartsheet.

[*Example*: sheet1.xml refers to a drawing that is the target of a relationship in the Chartsheet part's relationship item:

```
<chartsheet xmlns:r="…" …>
  <sheetViews>
    <sheetView scale="64"/>
  </sheetViews>
  <drawing r:id="rId1"/>
</chartsheet>
```

end example]

A Chartsheet part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Chartsheet part is permitted to have implicit relationships to the following parts defined by this Standard:

- Printer Settings (§15.2.14)

A Chartsheet part is permitted to have explicit relationships to the following parts defined by this Standard:

- Drawings (§12.3.8)

A Chartsheet part shall not have implicit or explicit relationships to any other part defined by this Standard.

12.3.3 Comments Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.comments+xml
Root Namespace:	http://schemas.openxmlformats.org/spreadsheetml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/comments

An instance of this part type contains all the comments for a given worksheet, as well as the names of the authors of those comments.

A package shall contain exactly one Comments part for each worksheet that contains one or more comments. If a Comments part exists, it shall be the target of an implicit relationship from the Workbook part (§12.3.23).

[*Example*: The following Worksheet part-relationship item contains a relationship to the Comments part, which is stored in the ZIP item comments2.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId2"
    Type="http://…/comments" Target="../comments2.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be comments.

[*Example*: This Comments part results from a workbook that has one or more comments from each of two people: James Jones and Mary Smith:

```
<comments xmlns:st="…" >
  <authors>
    <author>James Jones</author>
    <author>Mary Smith</author>
  </authors>
  <commentList>
    <comment r="C7" authorId="0">
      <text>
        <st:r>
          <st:rPr>
            …
          </st:rPr>
          <st:t>James Jones:</st:t>
        </st:r>
        <st:r>
          <st:rPr>
            …
          </st:rPr>
          <st:t>Check that this date is correct.</st:t>
        </st:r>
      </text>
    </comment>
```

```
      <comment r="E7" authorId="1">
        …
      </comment>
    </commentList>
  </comments>
```

end example]

A Comments part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Comments part shall not implicit or explicit relationships to any part defined by this Standard.

12.3.4 Connections Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.connections+xml
Root Namespace:	http://schemas.openxmlformats.org/spreadsheetml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/connections

An instance of this part type describes all of the connections currently established for a workbook.

A package shall contain no more than one Connections part, and that part shall be the target of an implicit relationship from the Workbook part (§12.3.23).

[*Example*: The following Workbook part-relationship item contains a relationship to the Connections part, which is stored in the ZIP item connections.xml:

```
  <Relationships xmlns="…">
    <Relationship Id="rId5"
      Type="http://…/connections" Target="connections.xml"/>
  </Relationships>
```

end example]

The root element for a part of this content type shall be connections.

[*Example*: A workbook has three connections, two from one worksheet, and one from another. connections.xml defines these three connections:

```
<connections …>
  <connection id="1" odcFile="…" keepAlive="1" name="…" type="5"
    refreshedVersion="2" background="1" saveData="1">
    <dbPr connection="Provider=MSDASQL.1;Persist Security Info=True;Data
Source=dBASE Files;Extended Properties=&quot;DSN=dBASE Files;DBQ=E:\MY
DOCUMENTS;DefaultDir=E:\MY
DOCUMENTS;DriverId=533;MaxBufferSize=2048;PageTimeout=5;&quot;;Initial
Catalog=E:\MY DOCUMENTS" command="`E:\MY DOCUMENTS`\`ADDRESS`"
commandType="3"/>
  </connection>
  <connection id="2" …>
    <dbPr … />
  </connection>
  <connection id="3" …>
    <dbPr … />
  </connection>
</connections>
```

end example]

A Connections part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Connections part shall not have implicit or explicit relationships to any part defined by this Standard..

12.3.5 Custom Property Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.customProperty
Root Namespace:	Not applicable
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/customProperty

This binary part supports the storage of arbitrary user-defined data.

A package shall contain at most one Custom Property part, and that part shall be the target of an implicit relationship from the Workbook (§12.3.23) part.

[*Example*: The following Workbook part-relationship item contains a relationship to the Custom Property part, which is stored in the ZIP item CustomProperty.bin:

```
<Relationships xmlns="…">
  <Relationship Id="rId7"
    Type="http://…/customProperty" Target="CustomProperty.bin"/>
</Relationships>
```

end example]

A Custom Property part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Custom Property part shall not have implicit or explicit relationships to any part defined by this Standard.

12.3.6 Custom XML Mappings Part

Content Type:	application/xml
Root Namespace:	http://schemas.openxmlformats.org/spreadsheetml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/xmlMaps

An instance of this part type contains a schema for an XML file, and information on the behavior that is used when allowing this custom XML schema to be mapped into the spreadsheet.

A package shall contain no more than one Custom XML Mappings part, and that part shall be the target of an implicit relationship from the Workbook part (§12.3.23). The Worksheet part into which this data is imported shall also have a relationship file that targets one or more Table Definition (§12.3.21) parts and/or one or more Single Cell Table Defintions (§12.3.19) parts.

[*Example*: The following Workbook part-relationship item contains a relationship to the Custom XML Mappings part, which is stored in the ZIP item xmlMaps.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId9"
    Type="http://…/xmlMaps" Target="xmlMaps.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be mapInfo.

[*Example*: xmlMaps.xml contains the following:

```
<mapInfo SelectionNamespaces="">
  <Schema ID="Schema1">
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element nillable="true" name="names">
           <xsd:complexType>
          <xsd:sequence minOccurs="0">
            <xsd:element minOccurs="0" maxOccurs="unbounded"
              nillable="true" name="name" form="unqualified">
```

```
                    <xsd:complexType>
                      <xsd:sequence minOccurs="0">
                        <xsd:element minOccurs="0" nillable="true"
                          type="xsd:string" name="firstname"
                          form="unqualified"/>
                        <xsd:element minOccurs="0" nillable="true"
                          type="xsd:string" name="initial"
                          form="unqualified"/>
                        <xsd:element minOccurs="0" nillable="true"
                          type="xsd:string" name="lastName"
                          form="unqualified"/>
                      </xsd:sequence>
                    </xsd:complexType>
                  </xsd:element>
                </xsd:sequence>
              </xsd:complexType>
            </xsd:element>
          </xsd:schema>
        </Schema>
        <Map ID="1" Name="names_Map" RootElement="names" SchemaID="Schema1"
          ShowImportExportValidationErrors="false" AutoFit="true"
          Append="false"
          PreserveSortAFLayout="true" PreserveFormat="true">
          <DataBinding FileBinding="Test.xml" DataBindingLoadMode="1"/>
        </Map>
      </mapInfo>
```

end example]

A Custom XML Mappings part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Custom XML Mappings part shall not have implicit or explicit relationships to any other part defined by this Standard.

12.3.7 Dialogsheet Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.dialogsheet+xml
Root Namespace:	http://schemas.openxmlformats.org/spreadsheetml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/dialogsheet

An instance of this part type contains information about a legacy custom dialog box for a user form.

A package is permitted to contain one or more Dialogsheet parts, and each such part shall be the target of an explicit relationship from the Workbook part (§12.3.23).

[*Example*: The following Workbook part-relationship item contains relationships to a Dialogsheet part, which is stored in the ZIP item dialogsheets/sheet1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId2"
    Type="http://…/dialogsheet" Target="dialogsheets/sheet1.xml"/>
</Relationships>
```

The Workbook part contains the following:

```
<workbook xmlns:r="…" …>
  …
  <sheets>
    …
    <sheet name="Dialog1" tabId="4" type="dialog" r:id="rId2"/>
  </sheets>
  …
</workbook>
```

end example]

The root element for a part of this content type shall be dialogsheet.

[*Example*: sheet1.xml contains the following:

```
<dialogsheet xmlns:r="…" …>
  <sheetPr>
    <pageSetUpPr/>
  </sheetPr>
  <sheetViews>
    …
  </sheetViews>
    …
  <legacyDrawing r:id="rId1"/>
</dialogsheet>
```

end example]

A Dialogsheet part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Dialogsheet part is permitted to have implicit relationships to the following parts defined by this Standard:

- Printer Settings (§15.2.14)

A Dialogsheet part is permitted to have explicit relationships to the following parts defined by this Standard:

- Embedded Control Persistence (§15.2.8)
- Drawings (§12.3.8)
- Embedded Object (§15.2.9)

A Dialogsheet part shall not have implicit or explicit relationships to any other part defined by this Standard.

12.3.8 Drawings Part

Content Type:	application/vnd.openxmlformats-officedocument.drawing+xml
Root Namespace:	http://schemas.openxmlformats.org/drawingml/2006/spreadsheetDrawing
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/drawing

An instance of this part type contains the presentation and layout information for one or more drawing elements that are present on this worksheet.

A package is permitted to contain one or more Drawings parts, and each such part shall be the target of an explicit relationship from a Worksheet part (§12.3.24), or a Chartsheet part (§12.3.2). There shall be only one Drawings part per worksheet or chartsheet.

[*Example*: The following Chartsheet part-relationship item contains a relationship to a Drawings part, which is stored in the ZIP item ../drawings/drawing1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http:// …/drawing" Target="../drawings/drawing1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be wsDr.

[*Example*: drawing1.xml refers to a chart that is the target of a relationship in the Drawing part's relationship item:

```
<xdr:wsDr xmlns:xdr="…" xmlns:a="…">
  <xdr:absoluteAnchor>
    <xdr:pos x="1518046" y="-1443632"/>
    <xdr:extents cx="8587382" cy="5848945"/>
    <xdr:graphicFrame macro="">
      <xdr:nvGraphicFramePr>
        <xdr:cNvPr id="24" name="Chart 24" descr=""/>
        <xdr:cNvGraphicFramePr/>
      </xdr:nvGraphicFramePr>
      <xdr:xfrm>
        <a:off x="0" y="0"/>
        <a:ext cx="0" cy="0"/>
      </xdr:xfrm>
      <a:graphic>
        <a:graphicData uri="http://…/chart">
          <a:chart relId="rId1"/>
        </a:graphicData>
      </a:graphic>
    </xdr:graphicFrame>
    <xdr:clientData/>
  </xdr:absoluteAnchor>
</xdr:wsDr>
```

end example]

A Drawings part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Drawings part is permitted to have explicit relationships to the following parts defined by this Standard:

- Chart (§14.2.1)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Hyperlinks (§15.3)
- Images (§15.2.13)

A Drawings part shall not have any implicit or explicit relationships to any other part defined by this Standard.

12.3.9 External Workbook References Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.externalLink+xml
Root Namespace:	http://schemas.openxmlformats.org/spreadsheetml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/externalLink

An instance of this part specifies information about data referenced in other SpreadsheetML packages.

[*Example*: Consider two workbooks, wb1 and wb2, stored in packages called wb1.xlsx and wb2.xlsx, respectively. The value of a cell on a worksheet in wb1 can be computed using the value of one or more cells in wb2. This is done by having wb1 contain an external reference to wb2. *end example*]

A package is permitted to contain one or more External Workbook References parts, and those parts shall be the target of an explicit relationship in the Workbook part (§12.3.23).

[*Example*: A Workbook part for wb1 contains the following, which indicates that somewhere in its three worksheets, an external reference is made to a target specified in relationship id rId4 of the part's relationship item:

```
<workbook xmlns:r="…"/>
  …
  <sheets>
    <sheet name="Sheet1" tabId="1" r:id="rId1"/>
    <sheet name="Sheet2" tabId="2" r:id="rId2"/>
    <sheet name="Sheet3" tabId="3" r:id="rId3"/>
  </sheets>
  …
  <externalReferences>
    <externalReference r:id="rId4"/>
  </externalReferences>
  …
</workbook>
```

That part's relationship item contains the following:

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
    Type="http://…/externalLink"
    Target="externalReferences/externalReference1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be externalLink.

[*Example*: externalReference1.xml contains:

```
<externalLink xmlns:r="…" … r:id="rId1">
  <externalBook>
    <sheetNames>
      <sheetName val="Sheet1"/>
      <sheetName val="Sheet2"/>
      <sheetName val="Sheet3"/>
    </sheetNames>
    <sheetDataSet>
      <sheetData sheetId="0">
        <row r="7">
          <cell r="C8">
            <v>0</v>
          </cell>
        </row>
      </sheetData>
      <sheetData sheetId="1"/>
      <sheetData sheetId="2"/>
    </sheetDataSet>
  </externalBook>
</externalLink>
```

This part's relationship item contains the following:

```
<Relationships …>
  <Relationship Id="rId1"
    Type="…/externalReference"
    Target="wb2.xlsx" TargetMode="External"/>
</Relationships>
```

where wb2.xlsx is the workbook in which one or more cells' values are used in calculating the values of a cell in workbook wb1. *end example*]

An External Workbook References part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

An External Workbook References part shall specify an explicit relationship to one or more this StandardExternal Workbooks (§12.4).

An External Workbook References part shall not have any implicit or explicit relationships to other parts defined by this Standard.

12.3.10 Metadata Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.sheetMetadata+xml

Root Namespace:	http://schemas.openxmlformats.org/spreadsheetml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/sheetMetadata

An instance of this part type contains information relating to a cell whose value is related to one or more other cells via OnLine Analytical Processing (OLAP) technology.

A package shall contain no more than one Cell Metadata part, and that part shall be the target of an implicit relationship from the Workbook part (§12.3.23).

[*Example*: The following Workbook part-relationship item contains a relationship to the Metadata part, which is stored in the ZIP item metadata.xml. Cell B3 contains the formula `CUBEMEMBER ("externalData", "[Account].[All Account]")`:

```
<Relationships xmlns="…">
  <Relationship Id="rId10"
    Type="http://…/sheetMetadata" Target="metadata.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be `metadata`.

[*Example*: metadata.xml contains the following:

```
<metadata …>
  <metadataTypes count="1">
    <metadataType name="XLMDX" minSupportedVersion="120000" copy="1"
      pasteAll="1" pasteValues="1" merge="1" splitFirst="1" rowColShift="1"
      clearFormats="1" clearComments="1" assign="1" coerce="1"/>
  </metadataTypes>
  <metadataStrings count="2">
    <s v="externalData"/>
    <s v="[Account].[All Account]"/>
  </metadataStrings>
  <mdxMetadata count="1">
    <m n="0" f="m">
      <t c="1">
        <n v="1"/>
      </t>
    </m>
  </mdxMetadata>
```

```
    <valueMetadata count="1">
      <b>
        <r t="1" v="0"/>
      </b>
    </valueMetadata>
  </metadata>
```

The corresponding Connections part contains the following:

```
  <connections …>
    <connection id="1" odcFile="…" keepAlive="1" name="externalData"
      description="…" type="5" refreshedVersion="3" background="1">
      <dbPr connection="Provider=MSOLAP.2;…" command="Budget" commandType="1"/>
      <olapPr sendLocale="1" rowDrillCount="1000" serverFill="1"
        serverNumberFormat="1" serverFont="1" serverFontColor="1"/>
    </connection>
  </connections>
```

The corresponding Volatile Dependencies part contains the following:

```
  <volTypes …">
    <volType type="cubeFunctions">
      <main first="externalData">
        <tp t="e">
          <v>#N/A</v>
          <stp>1</stp>
          <r r="B3" s="1"/>
        </tp>
      </main>
    </volType>
  </volTypes>
```

The corresponding Pivot Table Cache Definition part contains the following:

```
  <pivotCacheDefinition … saveData="0" refreshedBy="…"
    refreshedDate="2005-11-28T16:55:44" backgroundQuery="1" createdVersion="3"
    refreshedVersion="3" recordCount="0">
    <cacheSource type="external" connectionID="1"/>
    <cacheFields count="0"/>
    <cacheHierarchies count="6">
      …
    </cacheHierarchies>
```

```
    <kpis count="0"/>
    <tupleCache>
      <queryCache count="3">
        <query mdx="[product].[category]"/>
        <query mdx=""/>
        <query mdx="[Account].[All Account]">
          <tpls c="1">
            <tpl hier="0" item="4294967295"/>
          </tpls>
        </query>
      </queryCache>
    </tupleCache>
  </pivotCacheDefinition>
```

end example]

A Metadata part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Metadata part shall not have implicit or explicit relationships to any part defined by this Standard.

12.3.11 Pivot Table Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.pivotTable+xml
Root Namespace:	http://schemas.openxmlformats.org/spreadsheetml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/pivotTable

An instance of this part type contains a pivot table definition.

A package shall contain exactly one Pivot Table part per pivot table, and each such part shall be the target of an implicit relationship in the relationship part for the Worksheet part (§12.3.24) that corresponds to the worksheet containing the pivot table.

[*Example*: The following Worksheet part-relationship item contains a relationship to two Pivot Table parts, which are stored in the ZIP items ../pivotTables/pivotTable*N*.xml:

```
  <Relationships xmlns="…">
    <Relationship Id="rId1"
      Type="http://…/pivotTable" Target="../pivotTables/pivotTable1.xml"/>
    <Relationship Id="rId2"
      Type="http://…/pivotTable" Target="../pivotTables/pivotTable2.xml"/>
  </Relationships>
```

end example]

The root element for a part of this content type shall be pivotTableDefinition.

[*Example*: pivotTable1.xml contains the following:

```
<pivotTableDefinition … cache="4" applyNumberFormats="0" applyBorderFormats="0"
  applyFontFormats="0" applyPatternFormats="0" applyAlignmentFormats="0"
  applyWidthHeightFormats="1" dataCaption="Data" updatedVersion="3"
  minRefreshableVersion="3" useAutoFormatting="1" itemPrintTitles="1"
  createdVersion="3" indent="0" outline="1" outlineData="1">
  <location ref="H4:H5" firstHeaderRow="1" firstDataRow="1" firstDataCol="0"/>
  <pivotFields count="1">
    <pivotField dataField="1" numFmtId="0" outline="1"
      subtotalTop="1" showAll="0" measureFilter="0" sortType="manual"/>
  </pivotFields>
  <rowItems count="1">
    <i t="data"/>
  </rowItems>
  <colItems count="1">
    <i t="data"/>
  </colItems>
  <dataFields count="1">
    <dataField name="Sum of 1000" fld="0" subtotal="average"
      baseField="0" baseItem="0" numFmtId="0"/>
  </dataFields>
  <tableStyle name="TableStyle2" showRowHeaders="1" showColHeaders="1"
    showRowStripes="1" showColStripes="1"/>
</pivotTableDefinition>
```

end example]

A Pivot Table part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Pivot Table part is permitted to have implicit relationships to the following parts defined by this Standard:

- Pivot Table Cache Definition (§12.3.12).

A Pivot Table part shall not have any implicit or explicit relationships to other parts defined by this Standard.

12.3.12 Pivot Table Cache Definition Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.pivotCacheDefinition+xml
Root Namespace:	http://schemas.openxmlformats.org/spreadsheetml/2006/main

Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/pivotCacheDefinition

An instance of this part type contains a cache definition for a pivot table.

A package shall contain exactly one Pivot Table Cache Definition part per pivot table, and each such part shall be the target of an implicit relationship from a Pivot Table (§12.3.11) part as well as an explicit relationship from a Workbook (§12.3.23) part.

[*Example*: The following Pivot Table part-relationship item contains a relationship to the Pivot Table Cache Definition part, which is stored in the ZIP item ../pivotCache/pivotCacheDefinition2.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type=http://…/pivotCacheDefinition
    Target="../pivotCache/pivotCacheDefinition2.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be pivotCacheDefinition.

[*Example*: pivotCacheDefinition2.xml contains the following:

```
<pivotCacheDefinition … r:id="rId1" refreshedBy="John Jones"
  refreshedDate="2005-11-18T16:47:49" createdVersion="3"
  refreshedVersion="3" recordCount="11">
  <cacheSource type="worksheet">
    <worksheet range="C4:C15" sheet="Sheet1"/>
  </cacheSource>
  <cacheFields count="1">
    <cacheField name="1000">
      <sharedItems containsSemiMixedTypes="0" containsString="0"
        containsNumber="1" containsInteger="1" minValue="234
        maxValue="2543"/>
    </cacheField>
  </cacheFields>
</pivotCacheDefinition>
```

end example]

A Pivot Table Cache Definition part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Pivot Table Cache Definition part is permitted to have an explicit relationship to the following part defined by this Standard:

- Pivot Table Cache Records (§12.3.13).

A Pivot Table Cache Definition part shall not have any implicit or explicit relationships to other parts defined by this Standard.

12.3.13 Pivot Table Cache Records Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.pivotCacheRecords+xml
Root Namespace:	http://schemas.openxmlformats.org/spreadsheetml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/spreadsheetml/pivotCacheRecords

An instance of this part type contains the cache records for a pivot table.

A package shall contain exactly one Pivot Table Cache Records part per pivot table, and each such part shall be the target of an explicit relationship in the Pivot Table Cache Definition (§12.3.12) part for the corresponding pivot table.

[*Example*: The following Pivot Table Cache Definition part-relationship item contains a relationship to the Pivot Table Cache Records part, which is stored in the ZIP item pivotCacheRecords2.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/pivotCacheRecords" Target="pivotCacheRecords2.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be pivotCacheRecords.

[*Example*: pivotCacheRecords2.xml contains the following:

```
<pivotCacheRecords … count="11">
  <r>
    <n v="1234"/>
  </r>
    …
  <r>
    <n v="876"/>
  </r>
</pivotCacheRecords>
```

end example]

A Pivot Table Cache Records part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Pivot Table Cache Records part shall not have implicit or explicit relationships to any other part defined by this Standard.

12.3.14 Query Table Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.queryTable+xml
Root Namespace:	http://schemas.openxmlformats.org/spreadsheetml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/queryTable

An instance of this part type contains information that describes how the source table is connected to an external data source, and defines the properties that is used when this table is refreshed from that source.

A package is permitted to contain one Query Table part per table, and each of those parts shall be the target of an implicit relationship from the corresponding Table Definitions (§12.3.21) part.

[*Example*: The following Table part-relationship item contains a relationship to the Query Table part corresponding to the connections details for that table. These parts are stored in the ZIP items ../queryTables/queryTable*n*.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/queryTable"
    Target="../queryTables/queryTable1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be queryTable.

[*Example*: queryTable2.xml deals with a connection to a database file having the seven fields shown:

```
<queryTable … name="+Connect to New Data Source_1"
  growShrinkType="insertDelete" connectionId="2" autoFormatId="16"
  applyNumberFormats="0" applyBorderFormats="0" applyFontFormats="1"
  applyPatternFormats="1" applyAlignmentFormats="0"
  applyWidthHeightFormats="0">
```

```
    <queryTableRefresh nextId="8">
      <queryTableFields count="7">
        <queryTableField id="1" name="ACCOUNT"/>
        <queryTableField id="2" name="CHECKNUM"/>
        <queryTableField id="3" name="DATE"/>
        <queryTableField id="4" name="AMOUNT"/>
        <queryTableField id="5" name="PAYEE"/>
        <queryTableField id="6" name="CHARGECODE"/>
        <queryTableField id="7" name="DESCRIPT"/>
      </queryTableFields>
    </queryTableRefresh>
  </queryTable>
```

end example]

A Query Table part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Query Table part shall not have implicit or explicit relationships to any other part defined by this Standard.

12.3.15 Shared String Table Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.sharedStrings+xml
Root Namespace:	http://schemas.openxmlformats.org/spreadsheetml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/sharedStrings

An instance of this part type contains one occurrence of each unique string that occurs on all worksheets in a workbook.

A package shall contain exactly one Shared String Table part, and that part shall be the target of an implicit relationship from the Workbook part (§12.3.23).

[*Example*: The following Workbook part-relationship item contains a relationship to the Shared String Table part, which is stored in the ZIP item sharedStrings.xml:

```
  <Relationships xmlns="…">
    <Relationship Id="rId6"
      Type="http://…/sharedStrings" Target="sharedStrings.xml"/>
  </Relationships>
```

end example]

The root element for a part of this content type shall be sst.

[*Example*: Here are three of the six strings used in the worksheets:

```
<sst xmlns:st="…" … totalCount="6" uniqueCount="6">
  <sstItem>
    <t>Expenses Log</t>
  </sstItem>
  <sstItem>
    <t>Period Start</t>
  </sstItem>
  <sstItem>
    <t>Period End</t>
  </sstItem>
  …
</sst>
```

end example]

A Shared String Table part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Shared String Table part shall not have implicit or explicit relationships to any other part defined by this Standard.

12.3.16 Shared Workbook Revision Headers Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.revisionHeaders+xml
Root Namespace:	http://schemas.openxmlformats.org/spreadsheetml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/revisionHeaders

An instance of this part type contains information about each of the editing sessions performed on the parent workbook at the worksheet level (worksheets added and rearranged in each session).

A package shall contain at most one Shared Workbook Revision Headers part. If it exists, that part shall be the target of an implicit relationship from the Workbook (§12.3.23) part.

[*Example*: The following Workbook part-relationship item contains a relationship to the Shared Workbook Revision Headers part, which is stored in the ZIP item handout revisions/revisionHeaders.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId9"
    Type="http://…/revisionHeaders"
    Target="revisions/revisionHeaders.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be headers.

[*Example*: revisionHeaders.xml contains the following:

```
<headers xmlns:r="…" guid="{233BEE23-EB5C-4542-905D-0230EFFED88B}"
  diskRevisions="1" revisionId="4" version="3">
  <header guid="…" dateTime="…" maxSheetId="4" userName="…" r:id="rId1">
    <sheetIdMap count="3">
      …
    </sheetIdMap>
  </header>
  …
  <header guid="…" dateTime="…" maxSheetId="4" userName="…" r:id="rId3">
    <sheetIdMap count="3">
      …
    </sheetIdMap>
  </header>
</headers>
```

end example]

A Shared Workbook Revision Headers part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Shared Workbook Revision Headers part is permitted to have explicit relationships to the following parts defined by this Standard:

- Shared Workbook Revision Log (§12.3.17)

A Shared Workbook Revision Headers part shall not have any implicit or explicit relationships to other parts defined by this Standard.

12.3.17 Shared Workbook Revision Log Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.revisionLog+xml
Root Namespace:	http://schemas.openxmlformats.org/spreadsheetml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/revisionLog

An instance of this part type contains information about edits performed on individual cells in the parent workbook's worksheets in each editing session.

A package shall contain one Shared Workbook Revision Log part for each session's set of changes, and those parts shall be the target of an explicit relationship from the Shared Workbook Revision Headers (§12.3.16) part.

[*Example*: The following Shared Workbook Revision Headers part-relationship item contains a number of relationships to Shared Workbook Revision Log parts, which are stored in the ZIP item revisionLog*N*.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/revisionLog" Target="revisionLog1.xml"/>
  …
  <Relationship Id="rId6"
    Type="http://…/revisionLog" Target="revisionLog6.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be revisions.

[*Example*: revisionLog2.xml contains the following:

```
<revisions xmlns:xs="…" …>
  <rfmt sheetId="1" sqref="B4:B15">
    <dxf>
      <xs:fill>
        <xs:pattern patternType="solid">
          <xs:fgColor type="icv" val="64"/>
          <xs:bgColor type="rgb" val="4278252287"/>
        </xs:pattern>
      </xs:fill>
    </dxf>
  </rfmt>
  <rcv guid="{CBCE5672-5A4D-48C9-A120-F72804F8CF64}" action="delete"/>
  <rcv guid="{CBCE5672-5A4D-48C9-A120-F72804F8CF64}" action="add"/>
</revisions>
```

end example]

A Shared Workbook Revision Log part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Shared Workbook Revision Log part shall not have implicit or explicit relationships to any other part defined by this Standard.

12.3.18 Shared Workbook User Data Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.userNames+xml
Root Namespace:	http://schemas.openxmlformats.org/spreadsheetml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/usernames

An instance of this part type contains a list of all the users that are sharing the parent workbook.

A package shall contain at most one Shared Workbook User Data part, and that part shall be the target of an implicit relationship in the Workbook (§12.3.23) part.

[*Example*: The following Workbook part-relationship item contains a relationship to the Shared Workbook User Data part, which is stored in the ZIP item revisions/userNames.xml:

```
<Relationships xmlns="…">
  Relationship Id="rId8"
    Type="http://…/usernames" Target="revisions/userNames.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be users.

[*Example*: userNames.xml shows that there are two users sharing this workbook:

```
<users … count="2">
  <usrinfo guid="{B5A024F7-40BE-4A48-9B6D-B1655241C84D}"
    name="Mary Jones" id="-264292310" dateTime="2005-11-18T18:53:16"/>
  <usrinfo …/>
</users>
```

end example]

A Shared Workbook User Data part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Shared Workbook User Data part shall not have implicit or explicit relationships to any other part defined by this Standard.

12.3.19 Single Cell Table Definitions Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.tableSingleCells+xml
Root Namespace:	http://schemas.openxmlformats.org/spreadsheetml/2006/main

Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/tableSingleCells

An instance of this part type contains information on how to map non-repeating elements from a custom XML file into cells in a worksheet. [*Note*: Repeating custom XML elements are mapped using a Table (§12.3.21). *end note*]

A package shall contain at most one Single Cell Table Definitions part per worksheet, and that part shall be the target of an implicit relationship from a Worksheet (§12.3.24) part. A Single Cell Table Definitions part can describe one or more single cell table definitions for any given worksheet.

[*Example*: The following Worksheet part-relationship item contains a relationship to the Single Cell Table Definitions part, which is stored in the ZIP item ../tables/tableSingleCells1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/tableSingleCells"
    Target="../tables/tableSingleCells1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be singleCells.

[*Example*: A worksheet contains two single cell table definitions; e.g., ../tables/tableSingleCells1.xml contains the following, where the elements id and count are nested inside element names:

```
<singleCells …>
  <singleCell id="1" name="Table1" displayName="Table1" ref="B4">
    <cellPr id="1" uniqueName="id">
      <xmlPr mapId="1" xpath="/names/id" xmlDataType="string"/>
    </cellPr>
  </singleCell>
  <singleCell id="2" name="Table2" displayName="Table2" ref="B7">
    <cellPr id="1" uniqueName="count">
      <xmlPr mapId="1" xpath="/names/count" xmlDataType="integer"/>
    </cellPr>
  </singleCell>
</singleCells>
```

end example]

A Single Cell Table Definitions part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Single Cell Table Definitions part shall not have implicit or explicit relationships to any other part defined by this Standard.

12.3.20 Styles Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.styles+xml
Root Namespace:	http://schemas.openxmlformats.org/spreadsheetml/2006/mains
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/styles

An instance of this part type contains all the characteristics for all the cells in the workbook. Such information includes numeric and text formatting, alignment, font, color, and border.

A package shall contain no more than one Styles part, and that part shall be the target of an implicit relationship from the Workbook (§12.3.23) part.

[*Example*: The following Workbook part-relationship item contains a relationship to the Styles part, which is stored in the ZIP item styles.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId5"
    Type="http://…/styles" Target="styles.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be styleSheet.

[*Example*:

```
<styleSheet xmlns="…">
  <numFmts count="5">
    <numFmt numFmtId="164" formatCode="&quot;$&quot;#,##0.00"/>
    <numFmt numFmtId="165"
      formatCode="&quot;Yes&quot;;&quot;Yes&quot;;&quot;No&quot;"/>
    <numFmt numFmtId="166"
      formatCode="&quot;True&quot;;&quot;True&quot;;&quot;False&quot;"/>
    <numFmt numFmtId="167"
      formatCode="&quot;On&quot;;&quot;On&quot;;&quot;Off&quot;"/>
    <numFmt numFmtId="168"
      formatCode="[$€-2]\ #,##0.00_);[Red]\([$€-2]\ #,##0.00\)"/>
  </numFmts>
```

```
    <fonts count="5">
      …
    </fonts>
    <fills count="4">
      …
    </fills>
    <borders count="1">
      …
    </borders>
    …
    <colors>
      …
    </colors>
  </styleSheet>
```

end example]

A Styles part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Styles part shall not have implicit or explicit relationships to any other part defined by this Standard.

12.3.21 Table Definition Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.table+xml
Root Namespace:	http://schemas.openxmlformats.org/spreadsheetml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/table

An instance of this part type contains a description of a single table and its autofilter information. (The data for the table is stored in the corresponding Worksheet part.)

A package shall contain one Table Definition part per table, and each such part shall be the target of an implicit relationship from the Worksheet (§12.3.24) part that corresponds to the worksheet containing that table.

[*Example*: The following Worksheet part-relationship item contains relationships to two Table Definition parts, which are stored in the ZIP items ../tables/table*N*.xml:

```
  <Relationships xmlns="…">
    <Relationship Id="rId2"
      Type="http://…/table" Target="../tables/table1.xml"/>
    <Relationship Id="rId3"
      Type="http://…/table" Target="../tables/table2.xml"/>
  </Relationships>
```

end example]

The root element for a part of this content type shall be table.

[*Example*: table2.xml describes a table that spans a 2-column range of cells, F2:G19:

```
<table xmlns:af="…" … id="2" name="Table2" displayName="Table2" ref="F2:G19"
  totalsRowShown="0" headerRowDxfId="7">
  <autoFilter ref="F2:G19"/>
  <tableColumns count="2">
    <tableColumn id="1" name="Salesman" dataDxfId="9" totalsRowDxfId="6"/>
    <tableColumn id="2" name="Units" dataDxfId="8" totalsRowDxfId="5"/>
  </tableColumns>
  <tableStyle name="TableStyle2" showFirstColumn="0" showLastColumn="0"
    showRowStripes="1" showColumnStripes="1"/>
</table>
```

When the filter "Salesman equal to Smith" is applied, the autoFilter element in table2.xml is extended, as follows:

```
<autoFilter ref="F2:G19">
  <af:filterColumn colId="0">
    <af:filters>
      <af:filter val="Smith"/>
    </af:filters>
  </af:filterColumn>
</autoFilter>
```

end example]

A Table Definition part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be Internal).

A Table Definition part is permitted to explicit relationships to the following parts defined by this Standard:

- Query Table (§12.3.14)

A Table Definition part shall not have any implicit or explicit relationships to any other part defined by this Standard.

12.3.22 Volatile Dependencies Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.volatileDependencies+xml
Root Namespace:	http://schemas.openxmlformats.org/spreadsheetml/2006/main

Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/volatileDependencies

An instance of this part type contains information involving real-time data formulas in a workbook.

A package shall contain exactly one Volatile Dependencies part, and that part shall be the target of an implicit relationship from the Workbook (§12.3.23) part.

[*Example*: The following Workbook part-relationship item contains a relationship to the Volatile Dependencies part, which is stored in the ZIP item volatileDependencies.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId8"
    Type="http://…/volatileDependencies"
    Target="volatileDependencies.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be volTypes.

[*Example*:

```
<volTypes xmlns="…"/>
```

end example]

A Volatile Dependencies part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Volatile Dependencies part shall not have implicit or explicit relationships to any other part defined by this Standard.

12.3.23 Workbook Part

Content Type(s):	application/vnd.openxmlformats-officedocument.spreadsheetml.main+xml application/vnd.openxmlformats-officedocument.spreadsheetml.template.main+xml
Root Namespace:	http://schemas.openxmlformats.org/spreadsheetml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument

An instance of this part type contains workbook data and references to all of its worksheets.

A package shall contain exactly one Workbook part, and that part shall be the target of a relationship in the package-relationship item.

[*Example*: The following SpreadsheetML package-relationship item contains a relationship to the Workbook part, which is stored in the ZIP item xl/workbook.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/officeDocument" Target="xl/workbook.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be workbook.

[*Example*: This workbook has three worksheets, named January, February, and March:

```
<workbook xmlns="…" xmlns:r="…">
  <fileVersion lastEdited="4" lowestEdited="4" rupBuild="3417"/>
  <bookViews>
    <workbookView xWindow="240" yWindow="15" windowWidth="8505"
      windowHeight="6240"/>
  </bookViews>
  <sheets>
    <sheet name="January" tabId="1" r:id="rId1"/>
    <sheet name="February" tabId="2" r:id="rId2"/>
    <sheet name="March" tabId="3" r:id="rId3"/>
  </sheets>
  <workbookPr showObjects="all"/>
  <webPublishing codePage="1252"/>
  <calcPr calcId="122211" fullCalcOnLoad="1"/>
</workbook>
```

end example]

A Workbook part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Workbook part is permitted to have implicit relationships to the following parts defined by this Standard:

- Additional Characteristics (§15.2.1)
- Bibliography (§15.2.3)
- Calculation Chain (§12.3.1)
- Cell Metadata (§12.3.10)
- Connections (§12.3.4)
- Custom XML Mappings (§12.3.6)

- Custom XML Data Storage (§15.2.4)
- Shared String Table (§12.3.15)
- Shared Workbook Revision Headers (§12.3.16)
- Shared Workbook User Data (§12.3.18)
- Styles (§12.3.20)
- Theme (§14.2.7)
- Thumbnail (§15.2.14)
- Volatile Dependencies (§12.3.22)

A Workbook part is permitted to have explicit relationships to the following parts defined by this Standard:

- Chartsheet (§12.3.2)
- Dialogsheet (§12.3.7)
- External Workbook References (§12.3.8)
- Pivot Table Cache Definition (§12.3.12)
- Worksheet (§12.3.24)

A Workbook part shall not have implicit or explicit relationships to any other part defined by this Standard.

12.3.24 Worksheet Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.worksheet+xml
Root Namespace:	http://schemas.openxmlformats.org/spreadsheetml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/worksheet

An instance of this part type contains all the data, formulas, and characteristics associated with a given worksheet.

A package shall contain exactly one Worksheet part per worksheet, and those parts shall be the target of an explicit relationship from the Workbook (§12.3.23) part. Specifically, the id attribute on the sheet element shall reference the desired worksheet part.

[*Example*: The following Workbook part-relationship item contains three relationships to Worksheet parts, which are stored in the ZIP items worksheets/sheet*N*.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/worksheet" Target="worksheets/sheet1.xml"/>
  <Relationship Id="rId2"
    Type="http://…/worksheet" Target="worksheets/sheet2.xml"/>
  <Relationship Id="rId3"
    Type="http://…/worksheet" Target="worksheets/sheet3.xml"/>
 </Relationships>
```

end example]

The root element for a part of this content type shall be worksheet.

[*Example*: This worksheet, has cells in the range B1 to F8. Row 8 contains three cells whose values are calculated using the following formulas: D8=SUM(D5:D7), E8=SUM(E5:E7), and F8= D8-E8:

```
<worksheet xmlns="…" …>
  <sheetPr/>
  <dimension range="B1:F8"/>
  …
  <sheetData>
    <row r="1" spans="2:6" ht="360">
      <c r="B1" s="1" t="s">
        <v>0</v>
      </c>
    </row>
    …
    <row r="8" spans="2:6" ht="360">
      <c r="D8" s="5">
        <f>SUM(D5:D7)</f>
        <v>2280.5299999999997</v>
      </c>
      <c r="E8" s="5">
        <f>SUM(E5:E7)</f>
        <v>1251.31</v>
      </c>
      <c r="F8" s="6">
        <f>D8-E8</f>
        <v>1029.2199999999998</v>
      </c>
    </row>
  </sheetData>
  …
</worksheet>
```

end example]

A Worksheet part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Worksheet part is permitted to contain implicit relationships to the following parts defined by this Standard:

- Comments (§12.3.3)
- Pivot Table Definitions (§12.3.11)
- Printer Settings (§15.2.14)
- Single Cell Table Definitions (§12.3.19)
- Table Definition (§12.3.21)

A Worksheet part is permitted to contain explicit relationships to the following parts defined by this Standard:

- Drawings (§12.3.8)
- Embedded Control Persistence (§15.2.8)
- Embedded Object (§15.2.9)
- Embedded Package (§15.2.10)
- Hyperlinks (§15.3)
- Images (§15.2.13)
- VML Drawing (§15.2.17)

A Worksheet part shall not have implicit or explicit relationships to any other part defined by this Standard.

12.4 External Workbooks

Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/externalLinkPath

An *external workbook* is a SpreadsheetML package whose contents are being referenced by the current SpreadsheetML package. When a package refers to external workbooks, it shall store the location of those workbooks using this relationship.

A package is permitted to contain one or more External Workbook relationships, and those relationships shall be an explicit relationship from the External Workbook References (§12.3.9) part.

[*Example*: An External Workbook References part, which references the package c:\sourceData.xlsx would have an External Workbook relationship, which points at that file:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/externalLinkPath"
    Target="c:\sourceData.xlsx" TargetMode="External"/>
</Relationships>
```

end example]

A external workbook shall be located external to the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `External`).

13. PresentationML

This clause contains specifications for relationship items and parts that are specific to PresentationML. Parts than can occur in a PresentationML document, but are not PresentationML-specific, are specified in §15.2. Unless stated explicitly, all references to relationship items, content-type items, and parts in this clause refer to PresentationML ZIP items.

13.1 Glossary of PresentationML-Specific Terms

The following terms are used in the context of a PresentationML document:

comment — A note that an author or reviewer attaches to a piece of text in a document. Although a consumer may chose to display comments, they are not considered part of the body of the document. A comment includes the text of the note, the comment author's name and initials, and date of creation, among other things.

handout — A printed set of slides that can be handed out to an audience for future reference.

note — A slide annotation, reminder, or piece of text intended for the presenter or the audience.

presentation — A collection of slides intended to be viewed by an audience.

slide — A frame containing one or more pieces of text and/or images.

slide layout — The organization of elements on a slide.

13.2 Package Structure

A PresentationML package shall contain a package-relationship item and a content-type item. The package-relationship item shall contain implicit relationships with targets of the following type:

- One Presentation part (§13.3.6).

The package-relationship item is permitted to contain implicit relationships with targets of the following type:

- Digital Signature Origin (§15.2.6)
- File Property parts (§15.2.11) (Application-Defined File Properties, Core File Properties, and Custom File Properties), as appropriate.
- Thumbnail (§15.2.14).

The required and optional relationships between parts are defined in §13.3 and its subordinate clauses.

[*Example*: The following package represents the minimal conformant PresentationML package as defined by this Standard:

First, the content type for relationship parts and the Main Presentation part (the only required part) must be defined (physically located at /[Content_Types].xml in the package):

```
<Types xmlns="…">
  <Default Extension="rels"
    ContentType="application/vnd.openxmlformats-
      package.relationships+xml"/>
  <Override PartName="/presentation.xml"
    ContentType="application/vnd.openxmlformats-
      officedocument.presentationml.presentation.main+xml"/>
</Types>
```

Next, the single required relationship (the package-level relationship to the Main Presentation part) must be defined (physically located at /_rels/.rels in the package):

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://schemas.openxmlformats.org/officeDocument/2006/
      relationships/officeDocument"
    Target="presentation.xml"/>
</Relationships>
```

Finally, the minimum content for the Main Presentation part must be defined (physically located at /presentation.xml in the package):

```
<p:presentation xmlns:p="…">
  <p:notesSz cx="913607" cy="913607"/>
</p:presentation>
```

end example]

[*Example*: Consider a simple PresentationML document containing two slides, which both use an image as a template. Here's an example of the hierarchical folder structure that might be used for the ZIP items in the package for that document. As shown, a number of parts have their own relationship items:

```
/_rels/.rels                          Package-relationship item
/[Content_Types].xml                  Content-type item
/docProps/app.xml                     Application-Defined File
Properties part
/docProps/core.xml                    Core File Properties part
/docProps/custom.xml                  Custom File Properties part
/docProps/thumbnail.wmf               Package thumbnail image
/ppt/presentation.xml                 Presentation part
/ppt/_rels/presentation.xml.rels      Part-relationship item
```

```
/ppt/presProps.xml                              Presentation Properties part
/ppt/tableStyles.xml                            Table Styles part
/ppt/viewProps.xml                              View Properties part
/ppt/handoutMasters/handoutMaster1.xml          Handout Master part
/ppt/handoutMasters/_rels/handoutMaster1.xml.rels
                                                Part-relationship item
/ppt/media/image1.jpeg                          Slide template image
/ppt/notesMasters/notesMaster1.xml              Notes Master part
/ppt/notesMasters/_rels/notesMaster1.xml.rels
                                                Part-relationship item
/ppt/notesSlides/notesSlide1.xml                Notes Slide parts
/ppt/notesSlides/notesSlide2.xml
/ppt/notesSlides/_rels/notesSlide1.xml.rels
                                                Part-relationship items
/ppt/notesSlides/_rels/notesSlide2.xml.rels
/ppt/slideLayouts/slideLayout1.xml              Slide Layout parts 1–6
/ppt/slideLayouts/slideLayout2.xml
/ppt/slideLayouts/slideLayout3.xml
/ppt/slideLayouts/slideLayout4.xml
/ppt/slideLayouts/slideLayout5.xml
/ppt/slideLayouts/slideLayout6.xml
/ppt/slideLayouts/_rels/slideLayout1.xml.rels
                                                Part-relationship items
/ppt/slideLayouts/_rels/slideLayout2.xml.rels
/ppt/slideLayouts/_rels/slideLayout3.xml.rels
/ppt/slideLayouts/_rels/slideLayout4.xml.rels
/ppt/slideLayouts/_rels/slideLayout5.xml.rels
/ppt/slideLayouts/_rels/slideLayout6.xml.rels
/ppt/slideMasters/slideMaster1.xml              Slide Master part
/ppt/slideMasters/_rels/slideMaster1.xml.rels
                                                Part-relationship item
/ppt/slides/slide1.xml                          Slide parts
/ppt/slides/slide2.xml
/ppt/slides/_rels/slide1.xml.rels               Part-relationship items
/ppt/slides/_rels/slide2.xml.rels
/ppt/theme/theme1.xml                           Theme parts
/ppt/theme/theme2.xml
/ppt/theme/theme3.xml
```

```
/ppt/theme/themeOverride1.xml                    Theme Override parts
/ppt/theme/themeOverride2.xml
/ppt/theme/themeOverride3.xml
/ppt/theme/themeOverride4.xml
/ppt/theme/themeOverride5.xml
/ppt/theme/themeOverride6.xml
/ppt/theme/themeOverride7.xml
/ppt/theme/themeOverride8.xml
/ppt/theme/themeOverride9.xml
/ppt/theme/themeOverride10.xml
```

The package-relationship item contains the following:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/officeDocument" Target="ppt/presentation.xml"/>
  <Relationship Id="rId3"
    Type="http://…/core-properties" Target="docProps/core.xml"/>
  <Relationship Id="rId2"
    Type="http://…/thumbnail" Target="docProps/thumbnail.wmf"/>
  <Relationship Id="rId4"
    Type="http://…/extended-properties" Target="docProps/app.xml"/>
</Relationships>
```

end example]

13.3 Part Summary

The subclauses subordinate to this one describe in detail each of the part types specific to PresentationML.

[*Note*: For convenience, information from those subclauses is summarized in the following table:

Part	Relationship Target of	Root Element	Ref.
Comment Authors	Presentation	cmAuthorLst	§13.3.1
Comments	Slide	cmLst	§13.3.2
Handout Master	Presentation	handoutMaster	§13.3.3
Notes Master	Notes Slide, Presentation	notesMaster	§13.3.4
Notes Slide	Slide	notes	§13.3.5
Presentation	PresentationML package	presentation	§13.3.6
Presentation Properties	Presentation	presentationPr	§13.3.7
Slide	Presentation	sld	§13.3.8

Part	Relationship Target of	Root Element	Ref.
Slide Layout	Slide Master, Notes Slide, Presentation, Slide, Slide Master	sldLayout	§13.3.9
Slide Master	Presentation, Slide Layout	sldMaster	§13.3.10
Slide Synchronization Data	Slide	sldSyncPr	§13.3.11
User-Defined Tags	Presentation, Slide	tagLst	§13.3.12
View Properties	Presentation	viewPr	§13.3.13

end note]

13.3.1 Comment Authors Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.commentAuthors+xml
Root Namespace:	http://schemas.openxmlformats.org/presentationml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/commentAuthors

An instance of this part type contains information about each author who has added a comment to the document. That information includes the author's name, initials, a unique author-ID, a last-comment-index-used count, and a display color. (The color can be used when displaying comments to distinguish comments from different authors.)

A package shall contain at most one Comment Authors part. If it exists, that part shall be the target of an implicit relationship from the Presentation (§13.3.6) part.

[*Example*: The following Presentation part relationship item contains a relationship to the Comment Authors part, which is stored in the ZIP item commentAuthors.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId8"
    Type="http://…/commentAuthors" Target="commentAuthors.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be cmAuthorLst.

[*Example*: Two people have authored comments in this document: Mary Smith and Peter Jones. Her initials are "mas", her author-ID is 0, and her comments' display color index is 0. Since Mary's last-comment-index-used value is 3, the next comment-index to be used for her will be 4. His initials are "pjj", his author-ID is 1, and his

comments' display color index is 1. Since Peter's last-comment-index-used value is 1, the next comment-index to be used for him will be 2:

```
<p:cmAuthorLst xmlns:p="…" …>
  <p:cmAuthor id="0" name="Mary Smith" initials="mas" lastIdx="3"
    clrIdx="0"/>
  <p:cmAuthor id="1" name="Peter Jones" initials="pjj" lastIdx="1"
    clrIdx="1"/>
</p:cmAuthorLst>
```

end example]

A Comment Authors part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Comment Authors part shall not have implicit or explicit relationships to any other part defined by this Standard.

13.3.2 Comments Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.comments+xml
Root Namespace:	http://schemas.openxmlformats.org/presentationml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/comments

An instance of this part type contains the comments for a single slide. Each comment is tied to its author via an author-ID. Each comment's index number and author-ID combination are unique.

A package shall contain one Comments part for each slide containing one or more comments, and each of those parts shall be the target of an implicit relationship from its corresponding Slide (§13.3.8) part.

[*Example*: The following Slide part-relationship item contains a relationship to a Comments part, which is stored in the ZIP item ../comments/comment2.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
    Type="http://…/comments"
    Target="../comments/comment2.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be cmLst .

[*Example*: The Comments part contains three comments, two created by one author, and one created by another, all at the dates and times shown. The index numbers are assigned on a per-author basis, starting at 1 for an author's first comment:

```
<p:cmLst xmlns:p="…" …>
  <p:cm authorId="0" dt="2005-11-13T17:00:22.071" idx="1">
    <p:pos x="4486" y="1342"/>
    <p:text>Comment text goes here.</p:text>
  </p:cm>
  <p:cm authorId="0" dt="2005-11-13T17:00:34.849" idx="2">
    <p:pos x="3607" y="1867"/>
    <p:text>Another comment's text goes here.</p:text>
  </p:cm>
  <p:cm authorId="1" dt="2005-11-15T00:06:46.919" idx="1">
    <p:pos x="1493" y="2927"/>
    <p:text>comment …</p:text>
  </p:cm>
</p:cmLst>
```

end example]

A Comments part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Comments part shall not have implicit or explicit relationships to any other part defined by this Standard.

13.3.3 Handout Master Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.handoutMaster+xml
Root Namespace:	http://schemas.openxmlformats.org/presentationml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/handoutMaster

An instance of this part type contains the look, position, and size of the slides, notes, header and footer text, date, or page number on the presentation's handout.

A package shall contain at most one Handout Master part, and it shall be the target of an explicit relationship from the Presentation (§13.3.6) part.

[*Example*: The following Presentation part-relationship item contains a relationship to the Handout Master part, which is stored in the ZIP item handoutMasters/handoutMaster1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId5"
    Type="http://…/handoutMaster"
    Target="handoutMasters/handoutMaster1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be handoutMaster.

[*Example*:

```
<p:handoutMaster xmlns:p="…">
  <p:cSld name="">
    …
  </p:cSld>
  <p:clrMap … />
</p:handoutMaster>
```

end example]

A Handout Master part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Handout Master part is permitted to have implicit relationships to the following parts defined by this Standard:

- Additional Characteristics (§15.2.1)
- Bibliography (§15.2.3)
- Custom XML Data Storage (§15.2.4)
- Theme (§14.2.7)
- Thumbnail (§15.2.14)

A Handout Master part is permitted to have explicit relationships to the following parts defined by this Standard:

- Audio (§15.2.2)
- Chart (§14.2.1)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.8)
- Embedded Object (§15.2.9)
- Embedded Package (§15.2.10)
- Hyperlink (§15.3)
- Image (§15.2.13)

- Video (§15.2.14)
- VML Drawing (§15.2.17)

A Handout Master part shall not have implicit or explicit relationships to any other part defined by this Standard.

13.3.4 Notes Master Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.notesMaster+xml
Root Namespace:	http://schemas.openxmlformats.org/presentationml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/notesMaster

An instance of this part type contains information about the content and formatting of all notes pages.

A package shall contain at most one Notes Master part, and that part shall be the target of an implicit relationship from the Notes Slide (§13.3.5) part, as well as an explicit relationship from the Presentation (§13.3.6) part.

[*Example*: The following Presentation part-relationship item contains a relationship to the Notes Master part, which is stored in the ZIP item notesMasters/notesMaster1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
    Type="http://…/notesMaster" Target="notesMasters/notesMaster1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be notesMaster.

[*Example*:

```
<p:notesMaster xmlns:p="…">
  <p:cSld name="">
    …
  </p:cSld>
  <p:clrMap … />
</p:notesMaster>
```

end example]

A Notes Master part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Notes Master part is permitted to have implicit relationships to the following parts defined by this Standard:

- Additional Characteristics (§15.2.1)
- Bibliography (§15.2.3)
- Custom XML Data Storage (§15.2.4)
- Theme (§14.2.7)
- Thumbnail (§15.2.14)

A Notes Master part is permitted to have explicit relationships to the following parts defined by this Standard:

- Audio (§15.2.2)
- Chart (§14.2.1)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.8)
- Embedded Object (§15.2.9)
- Embedded Package (§15.2.10)
- Hyperlink (§15.3).
- Image (§15.2.13)
- Video (§15.2.14)
- VML Drawing (§15.2.17)

The Notes Master part shall not have implicit or explicit relationships to any other part defined by this Standard.

13.3.5 Notes Slide Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.notesSlide+xml
Root Namespace:	http://schemas.openxmlformats.org/presentationml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/notesSlide

An instance of this part type contains the notes for a single slide.

A package shall contain one Notes Slide part for each slide that contains notes. If they exist, those parts shall each be the target of an implicit relationship from the Slide (§13.3.8) part.

[*Example*: The following Slide part-relationship item contains a relationship to a Notes Slide part, which is stored in the ZIP item ../notesSlides/notesSlide1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId3"
    Type="http://…/notesSlide" Target="../notesSlides/notesSlide1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be notes.

[*Example*:

```
<p:notes xmlns:p="…">
  <p:cSld name="">
    …
  </p:cSld>
  <p:clrMapOvr>
    <a:masterClrMapping/>
  </p:clrMapOvr>
</p:notes>
```

end example]

A Notes Slide part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Notes Slide part is permitted to have implicit relationships to the following parts defined by this Standard:

- Additional Characteristics (§15.2.1)
- Bibliography (§15.2.3)
- Custom XML Data Storage (§15.2.4)
- Notes Master (§13.3.4)
- Theme Override(§14.2.8)
- Thumbnail (§15.2.14)

A Notes Slide part is permitted to have explicit relationships to the following parts defined by this Standard:

- Audio (§15.2.2)
- Chart (§14.2.1)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.8)
- Embedded Object (§15.2.9)
- Embedded Package (§15.2.10)
- Hyperlink (§15.3).
- Image (§15.2.13)

- Video (§15.2.14)
- VML Drawing (§15.2.17)

The Notes Slide part shall not have implicit or explicit relationships to any other part defined by this Standard.

13.3.6 Presentation Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.presentation.main+xml application/vnd.openxmlformats-officedocument.presentationml.slideshow.main+xml application/vnd.openxmlformats-officedocument.presentationml.template.main+xml
Root Namespace:	http://schemas.openxmlformats.org/presentationml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument

An instance of this part type contains the definition for a slide presentation.

A package shall contain exactly one Presentation part, and that part shall be the target of a relationship in the package-relationship item.

[*Example*: The following PresentationML's package-relationship item contains a relationship to the Presentation part, which is stored in the ZIP item ppt/presentation.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/officeDocument" Target="ppt/presentation.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be presentation.

[*Example*: This presentation contains two slides:

```
<p:presentation xmlns:p="…" … >
  <p:sldMasterIdLst>
    <p:sldMasterId
      xmlns:rel="http://…/relationships" rel:id="rId1"/>
  </p:sldMasterIdLst>
  <p:notesMasterIdLst>
    <p:notesMasterId
      xmlns:rel="http://…/relationships" rel:id="rId4"/>
  </p:notesMasterIdLst>
```

```
    <p:handoutMasterIdLst>
      <p:handoutMasterId
         xmlns:rel="http://…/relationships" rel:id="rId5"/>
    </p:handoutMasterIdLst>
    <p:sldIdLst>
      <p:sldId id="267"
        xmlns:rel="http://…/relationships" rel:id="rId2"/>
      <p:sldId id="256"
        xmlns:rel="http://…/relationships" rel:id="rId3"/>
    </p:sldIdLst>
    <p:sldSz cx="9144000" cy="6858000"/>
    <p:notesSz cx="6858000" cy="9144000"/>
  </p:presentation>
```

end example]

A Presentation part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Presentation part is permitted to have implicit relationships to the following parts defined by this Standard:

- Additional Characteristics (§15.2.1)
- Comments (§13.3.2)
- Bibliography (§15.2.3)
- Custom XML Data Storage (§15.2.4)
- Font (§15.2.12)
- Presentation Properties (§13.3.7)
- Table Styles (§14.2.9)
- Theme (§14.2.7)
- Thumbnail (§15.2.14)
- View Properties (§13.3.13).

A Presentation part is permitted to have explicit relationships to the following parts defined by this Standard:

- Notes Master (§13.3.4)
- Handout Master (§13.3.3)
- Slide (§13.3.8)
- Slide Layout (§13.3.9)
- Slide Master (§13.3.10)
- User Defined Tags (§13.3.12)

13.3.7 Presentation Properties Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.presentationProperties+xml
Root Namespace:	http://schemas.openxmlformats.org/presentationml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/presProps

An instance of this part type contains all the presentation's properties.

A package shall contain exactly one Presentation Properties part, and that part shall be the target of an implicit relationship from the Presentation (§13.3.6) part.

[*Example*: The following Presentation part-relationship item contains a relationship to the Presentation Properties part, which is stored in the ZIP item presProps.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId6"
    Type="http://…/presProps" Target="presProps.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be presentationPr.

[*Example*:

```
<p:presentationPr xmlns:p="…" …>
  <p:clrMru>
    …
  </p:clrMru>
  …
</p:presentationPr>
```

end example]

A Presentation Properties part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Presentation Properties part shall not have implicit or explicit relationships to any other part defined by this Standard.

13.3.8 Slide Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.slide+xml

Root Namespace:	http://schemas.openxmlformats.org/presentationml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/slide

A Slide part contains the contents of a single slide.

A package shall contain one Slide part per slide, and each of those parts shall be the target of an explicit relationship from the Presentation (§13.3.6) part.

[*Example*: Consider a PresentationML document having two slides. The corresponding Presentation part-relationship item contains two relationships to Slide parts, which are stored in the ZIP items slides/slide1.xml and slides/slide2.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId2"
    Type="http://…/slide" Target="slides/slide1.xml"/>
  <Relationship Id="rId3"
    Type="http://…/slide" Target="slides/slide2.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be sld.

[*Example*: slides/slide1.xml contains:

```
<p:sld xmlns:p="…">
  <p:cSld name="">
    …
  </p:cSld>
  <p:clrMapOvr>
    …
  </p:clrMapOvr>
  <p:timing>
    <p:tnLst>
      <p:par>
        <p:cTn id="1" dur="indefinite" restart="never" nodeType="tmRoot"/>
      </p:par>
    </p:tnLst>
  </p:timing>
</p:sld>
```

end example]

A Slide part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Slide part is permitted to have implicit relationships to the following parts defined by this Standard:

- Additional Characteristics (§15.2.1)
- Bibliography (§15.2.3)
- Comments (§12.3.3)
- Custom XML Data Storage (§15.2.4)
- Notes Slide (§13.3.5)
- Theme Override (§14.2.8)
- Thumbnail (§15.2.14)
- Slide Layout (§13.3.9)
- Slide Synchronization Data (§13.3.11)

A Slide part is permitted to have explicit relationships to the following parts defined by this Standard:

- Audio (§15.2.2)
- Chart (§14.2.1)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.8)
- Embedded Object (§15.2.9)
- Embedded Package (§15.2.10)
- Hyperlink (§15.3).
- Image (§15.2.13)
- User Defined Tags (§13.3.12)
- Video (§15.2.14)
- VML Drawing (§15.2.17)

A Slide part shall not have implicit or explicit relationships to any other part defined by this Standard.

13.3.9 Slide Layout Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.slideLayout+xml
Root Namespace:	http://schemas.openxmlformats.org/presentationml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/slideLayout

An instance of this part type contains the definition for a slide layout template for this presentation. This template defines the default appearance and positioning of drawing objects on this slide type when it is created.

A package shall contain one or more Slide Layout parts, and those parts shall be the target of an explicit relationship in the Presentation (§13.3.6) part, as well as an implicit relationship from the Slide Master (§13.3.10) part associated with this slide layout.

[*Example*: The following Slide Master part-relationship item contains relationships to several Slide Layout parts, which are stored in the ZIP items ../slideLayouts/slideLayout*N*.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/slideLayout"
    Target="../slideLayouts/slideLayout1.xml"/>
  <Relationship Id="rId2"
    Type="http://…/slideLayout"
    Target="../slideLayouts/slideLayout2.xml"/>
  <Relationship Id="rId3"
    Type="http://…/slideLayout"
    Target="../slideLayouts/slideLayout3.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be sldLayout.

[*Example*:

```
<p:sldLayout xmlns:p="…" matchingName="" type="title" preserve="1">
  <p:cSld name="Title Slide">
    …
  </p:cSld>
  <p:clrMapOvr>
    <a:masterClrMapping/>
  </p:clrMapOvr>
  <p:timing/>
  </p:sldLayout>
</p:sldMaster>
```

end example]

A Slide Layout part is permitted to have implicit relationships to the following parts defined by this Standard:

- Additional Characteristics (§15.2.1)
- Bibliography (§15.2.3)
- Custom XML Data Storage (§15.2.4)
- Slide Master (§13.3.10)
- Theme Override (§14.2.8)

- Thumbnail (§15.2.14)

A Slide Layout part is permitted to have explicit relationships to the following parts defined by this Standard:

- Audio (§15.2.2)
- Chart (§14.2.1)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.8)
- Embedded Object (§15.2.9)
- Embedded Package (§15.2.10)
- Hyperlink (§15.3).
- Image (§15.2.13)
- Video (§15.2.14)
- VML Drawing (§15.2.17)

A Slide Layout part shall not have implicit or explicit relationships to any other part defined by this Standard.

13.3.10 Slide Master Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.slideMaster+xml
Root Namespace:	http://schemas.openxmlformats.org/presentationml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/slideMaster

An instance of this part type contains the master definition of formatting, text, and objects that appear on each slide in the presentation that is derived from this slide master.

A package shall contain one or more Slide Master parts, each of which shall be the target of an explicit relationship from the Presentation (§13.3.6) part, as well as an implicit relationship from any Slide Layout (§13.3.9) part where that slide layout is defined based on this slide master. Each can optionally be the target of a relationship in a Slide Layout (§13.3.9) part as well.

[*Example*: The following Presentation part-relationship item contains a relationship to the Slide Master part, which is stored in the ZIP item slideMasters/slideMaster1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/slideMaster" Target="slideMasters/slideMaster1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be sldMaster.

[*Example*:

```
<p:sldMaster xmlns:p="…">
  <p:cSld name="">
    …
  </p:cSld>
  <p:clrMap … />
</p:sldMaster>
```

end example]

A Slide Master part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Slide Master part is permitted to have implicit relationships to the following parts defined by this Standard:

- Additional Characteristics (§15.2.1)
- Bibliography (§15.2.3)
- Custom XML Data Storage (§15.2.4)
- Slide Layout (§13.3.9)
- Theme (§14.2.7)
- Thumbnail (§15.2.14)

A Slide Master part is permitted to have explicit relationships to the following parts defined by this Standard:

- Audio (§15.2.2)
- Chart (§14.2.1)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.8)
- Embedded Object (§15.2.9)
- Embedded Package (§15.2.10)
- Hyperlink (§15.3).
- Image (§15.2.13)
- Video (§15.2.14)
- VML Drawing (§15.2.17)

A Slide Master part shall not have implicit or explicit relationships to any other part defined by this Standard.

13.3.11 Slide Synchronization Data Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.slideUpdateInfo+xml

Root Namespace:	http://schemas.openxmlformats.org/presentationml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/slideUpdateInfo

An instance of this part type contains properties specifying the current state of a slide that is being synchronized with a version of that slide stored on a central server.

A package shall contain zero or one Slide Synchronization Data part for each slide stored in the presentation, and that part shall be the target of an implicit relationship from the corresponding Slide (§13.3.8) part.

[*Example*: The following Slide part-relationship item contains a relationship to the Slide Synchronization Data part, which is stored in the ZIP item slideUpdateInfo/slideUpdateInfo1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1" Type="http://…/slideUpdateInfo"
    Target="slideUpdateInfo/slideUpdateInfo1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be sldSyncPr.

[*Example*:

```
<p:sldSyncPr xmlns:p="…" serverSldId="1"
  serverSldModifiedTime="2006-08-12T01:31:08"
  clientInsertedTime="2006-08-12T01:34:11.227" />
```

end example]

A Slide Synchronization Data part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Slide Synchronization Data part is permitted to have implicit relationships to the following parts defined by this Standard:

- Slide Synchronization Server Location (§13.4)

A Slide Synchronization Data part shall not have implicit or explicit relationships to any other part defined by this Standard.

13.3.12 User Defined Tags Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.tags+xml

Root Namespace:	http://schemas.openxmlformats.org/presentationml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/tags

An instance of this part type contains a set of user-defined properties for an object in a presentation (each property consisting of a name/value pair).

A package shall contain zero or more User Defined Tags parts, zero or one as the target of an explicit relationship from the corresponding Presentation (§13.3.6) or Slide (§13.3.8) part.

[*Example*: The following Slide part-relationship item contains a relationship to the User Defined Tags part, which is stored in the ZIP item tags/tag1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1" Type="http://…/tag"
    Target="tags/tag1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be tagLst.

[*Example*:

```
<p:tagLst xmlns:p="…" >
  <p:tag name="testTagName" val="testTagValue" />
</p:tagLst>
```

end example]

A User Defined Tags part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A User Defined Tags part shall not have implicit or explicit relationships to any other part defined by this Standard.

13.3.13 View Properties Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.viewProps+xml
Root Namespace:	http://schemas.openxmlformats.org/presentationml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/viewProps

An instance of this part type contains display properties for this presentation.

A package shall contain zero or one View Properties part, and if it exists, that part shall be the target of an implicit relationship from the Presentation (§13.3.6) part.

[*Example*: The following Presentation part-relationship item contains a relationship to the View Properties part, which is stored in the ZIP item viewProps.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId7"
    Type="http://…/viewProps" Target="viewProps.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be viewPr.

[*Example*:

```
<p:viewPr xmlns:p="…" …>
  <p:normalViewPr showOutlineIcons="0">
    …
  </p:normalViewPr>
  <p:slideViewPr>
    …
  </p:slideViewPr>
  <p:outlineViewPr>
    …
  </p:outlineViewPr>
  <p:notesTextViewPr>
    …
  </p:notesTextViewPr>
  <p:gridSpacing cx="78028800" cy="78028800"/>
</p:viewPr>
```

end example]

A View Properties part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A View Properties part shall not have implicit or explicit relationships to any other part defined by this Standard.

13.4 HTML Publish Location

Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/ htmlPubSaveAs

When a presentation specifies an external location to which an optional copy might be pushed in the HTML format, this relationship shall be used to target the location where the HTML copy of the presentation is published.

A package shall contain one HTML Publish Location relationship for each slide linked with an HTML publish location, and that relationships shall be an explicit relationship from the corresponding Presentation Properties (§13.3.7) part.

[*Example*: A Presentation Properties part, which stores an HTML Publish Location of http://www.openxmlformats.org/test.htm will contain the following relationship in that part's relationship part:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/htmlPubSaveAs"
    Target="http://www.openxmlformats.org/test.htm" type=”External”/>
</Relationships>
```

end example]

An HTML publish location shall be located external to the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `External`).

13.5 Slide Synchronization Server Location

Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/slideUpdateUrl

When a slide is being synchronized with a copy stored on a remote server, this relationship shall be used to target the location where the server copy of the slide is stored.

A package shall contain one Slide Synchronization Server Location relationship for each slide linked with server data, and that relationships shall be an implicit relationship from the corresponding Slide Synchronization Data (§13.3.11) part.

[*Example*: A Slide Synchronization Data part that stores information about a slide that is synchronized with a server located at http://www.openxmlformats.org/slides/ will contain the following relationship in that part's relationship part item:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/slideupdateUrl"
    Target="http://www.openxmlformats.org/slides/" type=”External”/>
</Relationships>
```

end example]

A slide synchronization server location shall be located external to the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `External`).

14. DrawingML

The relationship items and parts defined in this clause are used by one or more of WordprocessingML (§10), SpreadsheetML (§12), and PresentationML (§13) environments.

14.1 Glossary of DrawingML-Specific Terms

diagram — A picture or graphical representation that is displayed using a related set of color, data, layout, and style parts. Examples of diagram types are cycle, organization chart, pyramid, target, and Venn.

14.2 Part Summary

The subclauses subordinate to this one describe in detail each of the part types specific to DrawingML.

[*Note*: For convenience, information from those subclauses is summarized in the following table:

Part	Relationship Target of	Root Element	Ref.
Chart	WordprocessingML: Main Document SpreadsheetML: Drawings PresentationML: Handout Master, Notes Master, Notes Slide, Slide Layout, Slide Master, Slide All: Chart Drawing	chartSpace	§14.2.1
Chart Drawing	All: Chart	userShapes	§14.2.2
Diagram Colors	WordprocessingML: Main Document SpreadsheetML: Drawings PresentationML: Handout Master, Notes Master, Notes Slide, Slide Layout, Slide Master, Slide	colorsDef	§14.2.3
Diagram Data	WordprocessingML: Main Document SpreadsheetML: Drawings PresentationML: Handout Master, Notes Master, Notes Slide, Slide Layout, Slide Master, Slide	dataModel	§14.2.4
Diagram Layout Definition	WordprocessingML: Main Document SpreadsheetML: Drawings PresentationML: Handout Master, Notes Master, Notes Slide, Slide Layout, Slide Master, Slide	layoutDef	§14.2.5

Part	Relationship Target of	Root Element	Ref.
Diagram Style	WordprocessingML: Main Document SpreadsheetML: Drawings PresentationML: Handout Master, Notes Master, Notes Slide, Slide Layout, Slide Master, Slide	styleDef	§14.2.6
Theme	WordprocessingML: Main Document SpreadsheetML: Workbook PresentationML: Handout Master, Notes Master, Presentation, Slide Master	officeStyleSheet	§14.2.7
Theme Override	PresentationML: Notes Slide, Slide, Slide Layout	themeOverride	§14.2.8
Table Styles	PresentationML: Presentation	tblStyleLst	§14.2.9

end note]

14.2.1 Chart Part

Content Type:	application/vnd.openxmlformats-officedocument.drawingml.chart+xml
Root Namespace:	http://schemas.openxmlformats.org/drawingml/2006/chart
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/chart

An instance of this part type describes a chart.

A package shall contain a Chart part for each chart in the document. In a WordprocessingML document, each such part shall be the target of an explicit relationship in a Main Document (§11.3.10) part. In a SpreadsheetML document, each such part shall be the target of an explicit relationship in a Drawings (§12.3.8) part. In a PresentationML document, each such part shall be the target of an explicit relationship in a Handout Master (§13.3.3), Notes Master (§13.3.4), Notes Slide (§13.3.5), Slide (§13.3.8), Slide Layout (§13.3.9), or Slide Master (§13.3.10) part. This part is permitted to also be the target of an explicit relationship in a Chart Drawing (§14.2.2) part, if the chart that points at this Chart Drawing part is the target of a relationship from a Chartsheet part. In other words, the only time a chart can embed another chart is if the parent chart is part of a chartsheet.

[*Example*: The following Main Document part-relationship item contains relationships to two Chart parts, which are stored in the ZIP items ../charts/chart*N*.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
    Type="http://…/chart" Target="charts/chart1.xml"/>
  <Relationship Id="rId5"
    Type="http://…/chart" Target="charts/chart2.xml"/>
</Relationships>
```

The following Drawings part-relationship item contains a relationship to a Chart part, which is stored in the ZIP item ../charts/chart1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/relationships/chart" Target="../charts/chart1.xml"/>
</Relationships>
```

The following Slide part-relationship item contains relationships to two Chart parts, which are stored in the ZIP items ../charts/chart*N*.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
    Type="http://…/chart" Target="../charts/chart1.xml"/>
  <Relationship Id="rId5"
    Type="http://…/chart" Target="../charts/chart2.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be chartSpace.

[*Example*: chart1.xml contains the following clustered bar chart:

```
<c:chartSpace …>
  <c:chart>
    <c:title>
      …
    </c:title>
    <c:plotArea>
      <c:layout>
        …
      </c:layout>
      <c:barChart>
        …
      </c:barChart>
    </c:plotArea>
```

```
            <c:legend>
          …
        </c:legend>
      </c:chart>
      …
    </c:chartSpace>
```

end example]

For WordprocessingML and PresentationML documents, the data for a chart is not stored in the Chart part directly. Instead, it shall be stored in an embedded SpreadsheetML package (§12.2) targeted by an Embedded Package (§15.2.10) part specified by that Chart part. For SpreadsheetML documents, the data for a chart is stored directly in the Drawing's parent worksheet; no embedded SpreadsheetML package shall be used.

A Chart part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Chart part is permitted to have explicit relationships to the following parts defined by this Standard:

- Chart Drawing (§14.2.2)
- Embedded Package (§15.2.10)

A Chart part shall not have any implicit or explicit relationships to any other part defined by this Standard.

14.2.2 Chart Drawing Part

Content Type:	application/vnd.openxmlformats-officedocument.drawingml.chartshapes+xml
Root Namespace:	http://schemas.openxmlformats.org/drawingml/2006/chart
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/chartUserShapes

An instance of this part type contains all basic drawing elements (shapes) which are explicitly associated with this chart. These drawing elements are automatically moved with the chart when it is moved and resized when the chart is resized.

A package is permitted to contain one Chart Drawing part per chart part, and each such part shall be the target of an explicit relationship from a Chart (§14.2.1) part.

[*Example*: The following Chart part-relationship item contains a relationship to a Chart Drawing part, which is stored in the ZIP item ../drawings/drawing1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId2"
    Type="http://…/chartUserShapes" Target="../drawings/drawing1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be userShapes.

[*Example*:

```
<c:userShapes xmlns:cdr="…" xmlns:c="…">
  <cdr:relSizeAnchor>
    …
  </cdr:relSizeAnchor>
</c:userShapes>
```

end example]

A Chart Drawing part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Chart Drawing part is permitted to have explicit relationships to the following parts defined by this Standard:

- Chart (§14.2.1)

A Chart Drawing part shall not have any implicit or explicit relationships to any other part defined by this Standard.

14.2.3 Diagram Colors Part

Content Type:	application/vnd.openxmlformats-officedocument.drawingml.diagramColors+xml
Root Namespace:	http://schemas.openxmlformats.org/drawingml/2006/diagram
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/diagramColors

An instance of this part type contains color information for a diagram.

A package shall contain exactly one Diagram Colors part per diagram. Each Diagram Colors part shall be the target of an explicit relationship in a WordprocessingML Main Document (§11.3.10), SpreadsheetML Drawings (§12.3.8), or PresentationML Slide (§13.3.8) part.

[*Example*: The following SpreadsheetML Drawings part-relationship item contains a relationship to two Diagram Colors parts, which are stored in the ZIP items ../graphics/colors*N*.xml.

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
     Type="http://…/diagramColors" Target="../graphics/colors1.xml"/>
  <Relationship Id="rId8"
    Type="http://…/diagramColors" Target="../graphics/colors2.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be colorsDef.

[*Example*: colors1.xml contains the following:

```
<cs:colorsDef xmlns:cs="…" uniqueId="…" minVer="12.0">
  <cs:title lang="" val="Primary Accent 2"/>
  <cs:desc lang="" val="Primary Accent 2"/>
  <cs:catLst>
    <cs:cat type="accent1" pri="11200"/>
  </cs:catLst>
  <cs:styleLbl …>
    …
  </cs:styleLbl>
  …
  <cs:styleLbl …>
    …
  </cs:styleLbl>
</cs:colorsDef>
```

end example]

A Diagram Colors part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Diagram Colors part shall not have implicit or explicit relationships to any other part defined by this Standard.

14.2.4 Diagram Data Part

Content Type:	application/vnd.openxmlformats-officedocument.drawingml.diagramData+xml
Root Namespace:	http://schemas.openxmlformats.org/drawingml/2006/diagram
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/diagramData

An instance of this part type contains the semantic data for a diagram.

A package shall contain exactly one Diagram Data part per diagram. Each Diagram Data part shall be the target of an explicit relationship in a WordprocessingML Main Document (§11.3.10); a SpreadsheetML Drawings part (§12.3.8); or a PresentationML Handout Master (§13.3.3), Notes Master (§13.3.4), Notes Slide (§13.3.5), Slide (§13.3.8), Slide Layout (§13.3.9), or Slide Master (§13.3.10) part.

[*Example*: The following SpreadsheetML Drawings part-relationship item contains a relationship to two Diagram Data parts, which are stored in the ZIP items ../graphics/data*N*.xml.

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/diagramData" Target="../graphics/data1.xml"/>
  <Relationship Id="rId5"
    Type="http://…/diagramData" Target="../graphics/data2.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be dataModel.

[*Example*: data1.xml contains the following:

```
<dm:dataModel xmlns:dm="…">
  <dm:ptLst>
    …
  </dm:ptLst>
  <dm:cxnLst>
    …
  </dm:cxnLst>
  <dm:bg/>
  <dm:whole/>
</dm:dataModel>
```

end example]

A Diagram Data part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Diagram Data part is permitted to have explicit relationships to the following parts defined by this Standard:

- Image (§15.2.13)

A Diagram Data part shall not have any implicit or explicit relationships to other parts defined by this Standard.

14.2.5 Diagram Layout Definition Part

Content Type:	application/vnd.openxmlformats-officedocument.drawingml.diagramLayout+xml

Root Namespace:	http://schemas.openxmlformats.org/drawingml/2006/diagram
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/diagramLayout

An instance of this part type is a template that describes how diagram-related data is mapped to a shape.

A package shall contain exactly one Diagram Layout Definition part per diagram. Each Layout Definition part shall be the target of an explicit relationship from a WordprocessingML Main Document (§11.3.10); a SpreadsheetML Drawings part (§12.3.8); or a PresentationML Handout Master (§13.3.3), Notes Master (§13.3.4), Notes Slide (§13.3.5), Slide (§13.3.8), Slide Layout (§13.3.9), or Slide Master (§13.3.10) part. If a document contains multiple diagrams having the same graphic layout definition, each of those diagrams shall have its own copy of that Diagram Layout Definition part.

[*Example*: The following SpreadsheetML Drawings part-relationship item contains a relationship to two Diagram Layout Definition parts, which are stored in the ZIP items ../graphics/layout*N*.xml.

```
<Relationships xmlns="…">
  <Relationship Id="rId2"
    Type="http://…/diagramLayout" Target="../graphics/layout1.xml"/>
  <Relationship Id="rId6"
    Type="http://…/diagramLayout" Target="../graphics/layout2.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be layoutDef.

[*Example*: layout1.xml contains the following:

```
<lo:layoutDef xmlns:lo="…" uniqueId="…2" minVer="12.0" defStyle="">
  <lo:title lang="" val="Hierarchy 2"/>
  <lo:desc lang="" val=""/>
  <lo:catLst>
    <lo:cat type="hierarchy" pri="2000"/>
  </lo:catLst>
  <lo:sampData>
    …
  </lo:sampData>
  <lo:styleData>
    …
  </lo:styleData>
```

```
    <lo:clrData>
      …
    </lo:clrData>
    <lo:layoutNode name="Name0" styleLbl="" moveWith="">
      …
     </lo:layoutNode>
  </lo:layoutDef>
```

end example]

A Diagram Layout Definition part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Diagram Layout Definition part is permitted to have explicit relationships to the following parts and items defined by this Standard:

- Image (§15.2.13)

A Diagram Layout Definition part shall not have any implicit or explicit relationships to other parts defined by this Standard.

14.2.6 Diagram Style Part

Content Type:	application/vnd.openxmlformats-officedocument.drawingml.diagramStyle+xml
Root Namespace:	http://schemas.openxmlformats.org/drawingml/2006/diagram
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/diagramQuickStyle

An instance of this part type maps diagram semantic information to a document's theme.

A package shall contain exactly one Diagram Style part per diagram. Each Style part shall be the target of an explicit relationship from a WordprocessingML Main Document (§11.3.10); a SpreadsheetML Drawings part (§12.3.8); or a PresentationML Handout Master (§13.3.3), Notes Master (§13.3.4), Notes Slide (§13.3.5), Slide (§13.3.8), Slide Layout (§13.3.9), or Slide Master (§13.3.10) part.

[*Example*: The following SpreadsheetML Drawings part-relationship item contains a relationship to two Diagram Style parts, which are stored in the ZIP items ../graphics/quickStyle*N*.xml.

```
<Relationships xmlns="…">
  <Relationship Id="rId3"
    Type="http://…/diagramQuickStyle"
    Target="../graphics/quickStyle1.xml"/>
  <Relationship Id="rId7"
    Type="http://…/diagramQuickStyle"
    Target="../graphics/quickStyle2.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be styleDef.

[*Example*: quickStyle1.xml contains the following:

```
<qs:styleDef xmlns:qs="…" uniqueId="…" minVer="12.0">
  <qs:title lang="" val="Style 2"/>
  <qs:desc lang="" val="Style 2"/>
  <qs:catLst>
    <qs:cat type="simple" pri="10200"/>
  </qs:catLst>
  <qs:scene3d>
    …
  </qs:scene3d>
  <qs:style>
    …
  </qs:style>
  <qs:styleLbl name="…">
    …
  </qs:styleLbl>
  …
  <qs:styleLbl name="…">
    …
  </qs:styleLbl>
</qs:styleDef>
```

end example]

A Diagram Style part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Diagram Style part shall not have implicit or explicit relationships to other parts defined by this Standard.

14.2.7 Theme Part

Content Type:	application/vnd.openxmlformats-officedocument.theme+xml

Root Namespace:	http://schemas.openxmlformats.org/drawingml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/theme

An instance of this part type contains information about a document's *theme*, which is a combination of *color scheme*, *font scheme*, and *format scheme* (the latter also being referred to as *effects*). For a WordprocessingML document, the choice of theme affects the color and style of headings, among other things. For a SpreadsheetML document, the choice of theme affects the color and style of cell contents and charts, among other things. For a PresentationML document, the choice of theme affects the formatting of slides, handouts, and notes via the associated master, among other things.

A WordprocessingML or SpreadsheetML package shall contain zero or one Theme part, which shall be the target of an implicit relationship in a Main Document (§11.3.10) or Workbook (§12.3.23) part. A PresentationML package shall contain zero or one Theme part per Handout Master (§13.3.3), Notes Master (§13.3.4), Slide Master (§13.3.10) or Presentation (§13.3.6) part via an implicit relationship.

[*Example*: The following WordprocessingML Main Document part-relationship item contains a relationship to the Theme part, which is stored in the ZIP item theme/theme1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
    Type="http://…/theme" Target="theme/theme1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be officeStyleSheet.

[*Example*: theme1.xml contains the following, where the name attributes of the clrScheme, fontScheme, and fmtScheme elements correspond to the document's color scheme, font scheme, and format scheme, respectively:

```
<a:officeStyleSheet xmlns:a="…">
  <a:baseStyles>
    <a:clrScheme name="…">
      …
    </a:clrScheme>
    <a:fontScheme name="…">
      …
    </a:fontScheme>
```

```
      <a:fmtScheme name="…">
        …
      </a:fmtScheme>
    </a:baseStyles>
    <a:objectDefaults/>
  </a:officeStyleSheet>
```

end example]

A Theme part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Theme part is permitted to contain explicit relationships to the following parts defined by this Standard:

- Image (§15.2.13)

A Theme part shall not have any implicit or explicit relationships to other parts defined by this Standard.

14.2.8 Theme Override Part

Content Type:	application/vnd.openxmlformats-officedocument.themeOverride+xml
Root Namespace:	http://schemas.openxmlformats.org/drawingml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/themeOverride

An instance of this part type contains information about an object's *theme override*, which are overrides to the *color scheme*, *font scheme*, and *format scheme* (the latter also being referred to as *effects*) for a particular slide, notes slide, or handout.

A PresentationML package shall contain zero or one Theme Override part per Notes Slide (§13.3.5), Slide (§13.3.8), or Slide Layout (§13.3.9) part via an implicit relationship.

[*Example*: The following WordprocessingML Main Document part-relationship item contains a relationship to the Theme part, which is stored in the ZIP item theme/theme1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/themeOverride" Target="theme/themeoverride1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be ossOverride.

[*Example*:

```
<a:ossOverride xmlns:a="…" >
  …
</a:ossOverride>
```

end example]

A Theme Override part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Theme Override part shall not contain implicit or explicit relationships to other parts defined by this Standard.

14.2.9 Table Styles Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.tableStyles+xml
Root Namespace:	http://schemas.openxmlformats.org/drawingml/2006/main
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/tableStyles

An instance of this part type contains information about the table styles used by tables in this presentation. A table style defines characteristics such as row and column colors, heading row colors, and text.

A PresentationML package shall contain no more than one Table Styles part per Presentation (§13.3.6) part via an implicit relationship.

[*Example*: The following Presentation part-relationship item contains a relationship to a Table Styles part, which is stored in the ZIP item tableStyles.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/tableStyles" Target="tableStyles.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be tblStyleLst.

[*Example*: tablestyles.xml contains the following:

```
<a:tblStyleLst xmlns:a="…">
  <a:tblStyle>
    …
  </a:tblStyle>
</a:tblStyleLst>
```

end example]

A Table Styles part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Table Styles part shall not contain implicit or explicit relationships to other parts defined by this Standard.

15. Shared

The relationship items and parts defined in this clause are used by one or more of WordprocessingML (§10), SpreadsheetML (§12), and PresentationML (§13) environments.

15.1 Glossary of Shared Terms

control — A region of active content within an Office Open XML document.

15.2 Part Summary

The subclauses subordinate to this one describe in detail each of the shared part types.

[*Note*: For convenience, information from those subclauses is summarized in the following table:

Part	Relationship Target of	Root Element	Ref.
Additional Characteristics	Numerous PresentationML, SpreadsheetML, and WordprocessingML parts	Characteristics	§15.2.1
Audio	Numerous PresentationML, SpreadsheetML, and WordprocessingML parts	Not applicable	§15.2.2
Bibliography	Numerous PresentationML, SpreadsheetML, and WordprocessingML parts	Sources	§15.2.3
Custom XML Data Storage	Numerous PresentationML, SpreadsheetML, and WordprocessingML parts	Not applicable	§15.2.4
Custom XML Data Storage Properties	Custom XML Data Storage	datastoreItem	§15.2.5
Digital Signature Origin	WordprocessingML, SpreadsheetML, or PresentationML package	Not applicable	§15.2.6
Digital Signature XML Signature	Digital Signature Origin	Signature	§15.2.7
Embedded Control Persistence	Numerous PresentationML, SpreadsheetML, and WordprocessingML parts	Not applicable	§15.2.8
Embedded Object	Numerous PresentationML, SpreadsheetML, and WordprocessingML parts	Not applicable	§15.2.9

Part	Relationship Target of	Root Element	Ref.
Embedded Package	Numerous PresentationML, SpreadsheetML, and WordprocessingML parts	Not applicable	§15.2.10
File Properties, Extended	WordprocessingML, SpreadsheetML, or PresentationML package	Properties	§ 15.2.11.3
File Properties, Core	WordprocessingML, SpreadsheetML, or PresentationML package	coreProperties	§15.2.11.1
File Properties, Custom	WordprocessingML, SpreadsheetML, or PresentationML package	properties	§15.2.11.2
Font	WordprocessingML Font Table part, PresentationML Presentation part	Not applicable	§15.2.12
Image	Numerous PresentationML, SpreadsheetML, and WordprocessingML parts	Not applicable	§15.2.13
Printer Settings	SpreadsheetML Chartsheet, Dialogsheet, Worksheet parts, WordprocessingML Main Document or Glossary Document parts	Not applicable	§15.2.14
Thumbnail	WordprocessingML, SpreadsheetML, or PresentationML package	Not applicable	§15.2.14
Video part	Numerous PresentationML and WordprocessingML parts	Not applicable	§15.2.16

end note]

15.2.1 Additional Characteristics Part

Content Type:	application/xml
Root Namespace:	http://schemas.openxmlformats.org/officeDocument/2006/additionalCharacteristics
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/customXml

An instance of this part type contains information about additional characteristics of the producer that generated the document, when those characteristics cannot be specified using elements defined by this Standard.

A package is permitted to contain zero or one Additional Characteristics parts, and each such part shall be the target of an implicit relationship from a Main Document (§11.3.10) part in a WordprocessingML package; a Workbook (§12.3.23) part in a SpreadsheetML package; or a Handout Master (§13.3.3) , Notes Master (§13.3.4), Notes Slide (§13.3.5), Presentation (§13.3.6), Slide (§13.3.8), Slide Layout (§13.3.9), or Slide Master (§13.3.10) part in a PresentationML package.

[*Example*: The following Main Document part-relationship item contains a relationship to an Additional Characteristics part, which is stored in the ZIP item ../customXML/item2.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/customXmlData" Target="../customXML/item2.xml"/>
</Relationships>
```

end example]

An Additional Characteristics part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

An Additional Characteristics part is permitted to have implicit relationships to the following parts defined by this Standard:

- Custom XML Data Storage Properties (§15.2.5)

An Additional Characteristics part shall not have implicit or explicit relationships to any other part defined by this Standard.

15.2.2 Audio Part

<table>
<tr><td>Content Type:</td><td>Any supported audio type.

[Note: Some example content types are:
<table>
<tr><td>audio/aiff</td><td>http://developer.apple.com/documentation/QuickTime/INMAC/SOUND/imsoundmgr.30.htm</td></tr>
<tr><td>audio/midi</td><td>http://www.midi.org/about-midi/specinfo.shtml</td></tr>
<tr><td>audio/x-ms-wax</td><td>http://msdn.microsoft.com/library/en-us/wmplay10/mmp_sdk/asx_elementsintro.asp</td></tr>
</table>
end note]</td></tr>
<tr><td>Root Namespace:</td><td>not applicable</td></tr>
<tr><td>Source Relationship:</td><td>http://schemas.openxmlformats.org/officeDocument/2006/relationships/audio</td></tr>
</table>

An instance of this part type contains an audio file.

A PresentationML package is permitted to contain zero or more Sound parts, each of which shall be the target of a relationship in a Handout Master (§13.3.3), Notes Slide (§13.3.5), Notes Master (§13.3.4), Slide (§13.3.8), Slide Layout (§13.3.9), or Slide Master (§13.3.10) part-relationship item. [*Example*: The following Slide part-relationship item contains a relationship to a Sound part, which is stored as the file E:\Beethoven's Symphony No. 9.wma:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/audio/x-ms-wma"
    Target="file:///E:\Beethoven's%20Symphony%20No.%209.wma"
    TargetMode="External"/>
</Relationships>
```

end example]

An Audio part may be located within or external to the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element may be `Internal` or `External`).

An Audio part is not stored as XML; instead, it involves a relationship target that is an audio clip.

An Audio part shall not have implicit or explicit relationships to other parts defined by this Standard.

15.2.3 Bibliography Part

Content Type:	application/xml
Root Namespace:	http://schemas.openxmlformats.org/officeDocument/2006/bibliography
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/customXml

An instance of this part type contains bibliographic data for the current package.

A package is permitted to contain zero or one Bibliography part, and each such part shall be the target of an implicit relationship in a Main Document (§11.3.10) part in a WordprocessingML package; a Workbook (§12.3.23) part in a SpreadsheetML package; or a Handout Master (§13.3.3) , Notes Master (§13.3.4), Notes Slide (§13.3.5), Slide (§13.3.8), Slide Layout (§13.3.9), or Slide Master (§13.3.10) part in a PresentationML package.

[*Example*: The following Main Document part-relationship item contains a relationship to a Bibliography part, which is stored in the ZIP item ../customXML/bib1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/customXml" Target="../customXML/bib1.xml"/>
</Relationships>
```

end example]

A Bibliography part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Bibliography part is permitted to have implicit relationships to the following parts defined by this Standard:

- Custom XML Data Storage Properties (§15.2.5)

A Bibliography part shall not have implicit or explicit relationships to any other part defined by this Standard.

15.2.4 Custom XML Data Storage Part

Content Type:	application/xml
Root Namespace:	any XML allowed
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/customXml

An instance of this part type can contain arbitrary XML. As such, an instance of this part can be used to roundtrip arbitrary custom XML data with this package.

A package is permitted to contain one or more Custom XML Data Storage parts, and each such part shall be the target of an implicit relationship in a Main Document (§11.3.10) part in a WordprocessingML package; a Workbook (§12.3.23) part in a SpreadsheetML package; or a Handout Master (§13.3.3) , Notes Master (§13.3.4), Notes Slide (§13.3.5), Presentation (§13.3.6), Slide (§13.3.8), Slide Layout (§13.3.9), or Slide Master (§13.3.10) part in a PresentationML package.

[*Example*: The following Main Document part-relationship item contains a relationship to a Custom XML Data Storage part, which is stored in the ZIP item ../customXML/item1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/customXmlData" Target="../customXML/item1.xml"/>
</Relationships>
```

end example]

A Custom XML Data Storage part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Custom XML Data Storage part is permitted to have implicit relationships to the following parts defined by this Standard:

- Custom XML Data Storage Properties (§15.2.5)

A Custom XML Data Storage part shall not have implicit or explicit relationships to any other part defined by this Standard.

15.2.5 Custom XML Data Storage Properties Part

Content Type:	application/vnd.openxmlformats-officedocument.customXmlProperties+xml
Root Namespace:	http://schemas.openxmlformats.org/officeDocument/2006/customXmlDataProps
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/customXmlProps

An instance of this part type contains the set of properties which are specified for this custom XML data. These properties consist of a unique ID for the storage, as well as information on the set of XML schemas used by this custom XML data storage.

A package is permitted to contain zero or more Custom XML Data Storage Properties parts, and each such part shall be the target of an implicit relationship from a Custom XML Data Storage (§15.2.4) part.

[*Example*: The following Custom XML Data Storage part-relationship item contains a relationship to a Custom XML Data Storage Properties part, which is stored in the ZIP item itemProps1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/customXmlProps" Target="itemProps1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be datastoreItem.

[*Example*:

```
<ds:datastoreItem ds:itemID="{D85…53A}" xmlns:ds="…"/> \
```

end example]

A Custom XML Data Storage Properties part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be Internal).

A Custom XML Data Storage Properties part shall not have implicit or explicit relationships to other parts defined by this Standard.

15.2.6 Digital Signature Origin Part

Content Type:	application/vnd.openxmlformats-package.digital-signature-origin

Root Namespace:	not applicable
Source Relationship:	http://schemas.openxmlformats.org/package/2006/relationships/digital-signature/origin

The Digital Signature Origin part is the starting point for navigating through the signatures in a package.

This part shall have no contents.

A package is permitted to contain zero or one Digital Signature Origin part in a package and it shall the target of a relationship from the package-relationship item for a WordprocessingML, SpreadsheetML, or PresentationML package.

[*Example*: The following package-relationship item contains a relationship to a Digital Signature Origin part, which is stored in the ZIP item origin.sigs:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/origin" Target="../origin.sigs"/>
</Relationships>
```

end example]

A Digital Signature Origin part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Digital Signature Origin part is permitted to have implicit relationships to the following parts defined by this Standard:

- Digital Signature XML Signature (§15.2.7)

A Digital Signature Origin part shall not have implicit or explicit relationships to any other part defined by this Standard.

15.2.7 Digital Signature XML Signature Part

Content Type:	application/vnd.openxmlformats-package.digital-signature-xmlsignature+xml
Root Namespace:	http://schemas.openxmlformats.org/package/2006/digital-signature
Source Relationship:	http://schemas.openxmlformats.org/package/2006/relationships/digital-signature/signature

The Digital Signature XML Signature part contains digital signature markup.

A package is permitted to contain zero or more Digital Signature XML Signature parts, one for each digital signature in a package, and each of these shall be the target of an implicit relationship from the Digital Signature Origin (§15.2.6) part.

[*Example*: The following Digital Signature Origin part-relationship item contains a relationship to a Digital Signature XML Signature part, which is stored in the ZIP item sig1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/signature" Target="../sig1.xml"/>
</Relationships>
```

end example]

The root element for this part shall be Signature.

[*Example*:

```
<Signature xmlns="…" Id="idPackageSignature" >
  …
</Signature>
```

end example]

A Digital Signature XML Signature part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Digital Signature XML Signature part shall not have implicit or explicit relationships to any part defined by this Standard.

15.2.8 Embedded Control Persistence Part

Content Type:	Any supported control type. [*Note*: There are a number of possible control types. One example of a potential control type would be an Active X control, which would use the following content type: `application/vnd.ms-office.activeX+xml`. *end note*]
Root Namespace:	not applicable
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/control

An instance of this part contains information about an embedded control in the package. This information is provided by the specified control when asked to persist.

A package is permitted to contain one or more Embedded Control Persistence parts, and each such part shall be the target of an explicit relationship in an Endnotes (§11.3.4), Footer (§11.3.6), Footnotes (§11.3.7), Header (§11.3.9), or Main Document (§11.3.10) part-relationship item in a WordprocessingML package; a Worksheet part (§12.3.24) 15.2.17in a SpreadsheetML package; or a Handout Master (§13.3.3), Notes Slide (§13.3.5), Notes Master (§13.3.4), Slide (§13.3.8), Slide Layout (§13.3.9), Slide Master (§13.3.10) part-relationship item in a PresentationML package.

The content type of this part shall determine the format and contents of the embedded control.

[*Example*: The following example shows the persistence that could be used for an embedded control which is a Java applet within a WordprocessingML document:

```
<w:p>
  <w:r w:rsidR="005810E1">
    <w:object w:dxaOrig="1440" w:dyaOrig="1440">
      <v:shapetype id="_x0000_t75" coordsize="21600,21600" o:spt="75"
o:preferrelative="t" path="m@4@5l@4@11@9@11@9@5xe" filled="f" stroked="f">
        <v:stroke joinstyle="miter" />
        <v:formulas>
          <v:f eqn="if lineDrawn pixelLineWidth 0" />
          <v:f eqn="sum @0 1 0" />
          <v:f eqn="sum 0 0 @1" />
          <v:f eqn="prod @2 1 2" />
          <v:f eqn="prod @3 21600 pixelWidth" />
          <v:f eqn="prod @3 21600 pixelHeight" />
          <v:f eqn="sum @0 0 1" />
          <v:f eqn="prod @6 1 2" />
          <v:f eqn="prod @7 21600 pixelWidth" />
          <v:f eqn="sum @8 21600 0" />
          <v:f eqn="prod @7 21600 pixelHeight" />
          <v:f eqn="sum @10 21600 0" />
        </v:formulas>
        <v:path o:extrusionok="f" gradientshapeok="t" o:connecttype="rect" />
        <o:lock v:ext="edit" aspectratio="t" />
      </v:shapetype>
      <v:shape id="_x0000_i1027" type="#_x0000_t75"
style="width:1in;height:24pt" o:ole="">
        <v:imagedata r:id="rId4" o:title="" />
      </v:shape>
      <w:control r:id="rId5" w:name="CommandButton1" w:shapeid="_x0000_i1027"
/>
    </w:object>
  </w:r>
</w:p>
```

The relationship type for rId5 is: http://schemas.openxmlformats.org/officeDocument/2006/relationships/control

The XML content of the part referenced by rId5 could be:

```
<applet xlink:href="../../../../Program%20Files/Application"
xlink:type="simple" xlink:show="embed" xlink:actuate="onLoad"
code="CalculateApplet.class" may-script="false"/>
```

end example]

[*Example*: The following example shows the persistence that could be used for an embedded control which is an ActiveX control within a WordprocessingML document:

```
<w:p>
  <w:r w:rsidR="005810E1">
    <w:object w:dxaOrig="1440" w:dyaOrig="1440">
      <v:shapetype id="_x0000_t75" coordsize="21600,21600" o:spt="75"
o:preferrelative="t" path="m@4@5l@4@11@9@11@9@5xe" filled="f" stroked="f">
        <v:stroke joinstyle="miter" />
        <v:formulas>
          <v:f eqn="if lineDrawn pixelLineWidth 0" />
          <v:f eqn="sum @0 1 0" />
          <v:f eqn="sum 0 0 @1" />
          <v:f eqn="prod @2 1 2" />
          <v:f eqn="prod @3 21600 pixelWidth" />
          <v:f eqn="prod @3 21600 pixelHeight" />
          <v:f eqn="sum @0 0 1" />
          <v:f eqn="prod @6 1 2" />
          <v:f eqn="prod @7 21600 pixelWidth" />
          <v:f eqn="sum @8 21600 0" />
          <v:f eqn="prod @7 21600 pixelHeight" />
          <v:f eqn="sum @10 21600 0" />
        </v:formulas>
        <v:path o:extrusionok="f" gradientshapeok="t" o:connecttype="rect" />
        <o:lock v:ext="edit" aspectratio="t" />
      </v:shapetype>
      <v:shape id="_x0000_i1027" type="#_x0000_t75"
style="width:1in;height:24pt" o:ole="">
        <v:imagedata r:id="rId4" o:title="" />
      </v:shape>
      <w:control r:id="rId5" w:name="CommandButton1" w:shapeid="_x0000_i1027"
/>
    </w:object>
  </w:r>
```

```
</w:p>
```

The relationship type for rId5 is: `http://schemas.openxmlformats.org/officeDocument/2006/relationships/control`

The content type of the part referenced by rId5 could be: `application/vnd.ms-office.activeX+xml`

The XML content of the part referenced by rId5 could be:

```
<ax:ocx ax:classid="{D7053240-CE69-11CD-A777-00DD01143C57}"
    ax:persistence="persistPropertyBag"
    xmlns:ax="http://schemas.microsoft.com/office/2006/activeX">
  <ax:ocxPr ax:name="Caption" ax:value="CommandButton1" />
  <ax:ocxPr ax:name="Size" ax:value="2540;847" />
  <ax:ocxPr ax:name="FontName" ax:value="Calibri" />
  <ax:ocxPr ax:name="FontHeight" ax:value="225" />
  <ax:ocxPr ax:name="FontCharSet" ax:value="0" />
  <ax:ocxPr ax:name="FontPitchAndFamily" ax:value="2" />
  <ax:ocxPr ax:name="ParagraphAlign" ax:value="3" />
</ax:ocx>
```

end example]

An Embedded Control Persistence part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

An Embedded Control Persistence part shall not have any implicit or explicit relationships to other parts defined by this Standard.

15.2.9 Embedded Object Part

Content Type:	application/vnd.openxmlformats-officedocument.oleObject
Root Namespace:	not applicable
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/oleObject

An instance of this part type can contain an embedded object produced by any embedded object server.

A package is permitted to contain zero or more Embedded Object parts, and each such part shall be the target of an explicit relationship from a Comments (§11.3.2), Endnotes (§11.3.4), Footer (§11.3.6), Footnotes (§11.3.7), Header (§11.3.9), or Main Document (§11.3.10) part in a WordprocessingML package; a Worksheet part (§12.3.24) in a SpreadsheetML package; or a Handout Master (§13.3.3), Notes Slide (§13.3.5), Notes Master (§13.3.4), Slide (§13.3.8), Slide Layout (§13.3.9), Slide Master (§13.3.10) part in a PresentationML package.

A WordprocessingML document package is permitted to contain zero or more Embedded Object parts, each of which shall be the target of a relationship in a Main Document part-relationship item. Each Embedded Object part shall have an associated image, which appears in the document as a placeholder for the corresponding embedded object.

[*Example*: Consider the case in which a WordprocessingML document has embedded in it one video object and one audio object. The following Main Document part-relationship item contains relationships to two Embedded parts (one each for the video and audio), which are stored in the ZIP items embeddings/embeddedObject*N*.bin:

```
<Relationships xmlns="…">
  <Relationship Id="rId5"
    Type="http://…/oleObject" Target="embeddings/embeddedObject1.bin"/>
  <Relationship Id="rId7"
    Type="http://…/oleObject" Target="embeddings/embeddedObject2.bin"/>
  <Relationship Id="rId4"
    Type="http://…/image" Target="media/image1.png"/>
  <Relationship Id="rId6"
    Type="http://…/image" Target="media/image2.png"/>
</Relationships>
```

example]

A SpreadsheetML document package is permitted to contain zero or more Embedded Object parts, each of which shall be the target of a relationship in a Worksheet part-relationship item.

[*Example*: Consider the case in which a SpreadsheetML document has embedded in it one video object and one audio object on one worksheet, and another audio object embedded in another worksheet. The following Worksheet Document part-relationship item contains relationships to two Embedded Object parts (one each for the video and audio), which are stored in the ZIP items ../embeddings/embeddedObject*N*.bin:

```
<Relationships xmlns="…">
   <Relationship Id="rId2"
     Type="http://…/oleObject" Target="../embeddings/embeddedObject1.bin"/>
  <Relationship Id="rId3"
    Type="http://…/oleObject" Target="../embeddings/embeddedObject2.bin"/>
</Relationships>
```

end example]

A PresentationML document package is permitted to contain zero or more Embedded Object parts, each of which shall be the target of a relationship in a Slide part-relationship item.

[*Example*: Consider the case in which a PresentationML document has embedded in it one video object and one audio object on one slide, and another audio object embedded on another slide. The following Slide part-

relationship item contains relationships to two Embedded Object parts (one each for the video and audio), which are stored in the ZIP items ../embeddings/embeddedObject*N*.bin:

```
<Relationships xmlns="…">
  <Relationship Id="rId6"
    Type="http://…/oleObject"
    Target="../embeddings/embeddedObject1.bin"/>
  <Relationship Id="rId7"
    Type="http://…/oleObject"
    Target="../embeddings/embeddedObject2.bin"/>
</Relationships>
```

end example]

An Embedded Object part may be located within or external to the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element may be `Internal` or `External`).

An Embedded Object part is permitted to have an explicit relationship to the following parts defined by this Standard:

- Hyperlink (§15.3)

An Embedded Object part shall not have any implicit or explicit relationships to other parts defined by this Standard.

15.2.10 Embedded Package Part

Content Type:	application/vnd.openxmlformats-officedocument.package
Root Namespace:	not applicable
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/package

An instance of this part type contains a complete package. For example, a WordprocessingML document might contain a SpreadsheetML or PresentationML document, in which case, the WordprocessingML document package would contain an embedded package part that defined that SpreadsheetML or PresentationML document.

A package is permitted to contain zero or more Embedded Package parts, and each such part shall be the target of an explicit relationship from a Chart (§14.2.1), Comments (§11.3.2), Endnotes (§11.3.4), Footer (§11.3.6), Footnotes (§11.3.7), Header (§11.3.9), or Main Document (§11.3.10) part in a WordprocessingML package; a Chart (§14.2.1), or Worksheet part (§12.3.24) in a SpreadsheetML package; or a Chart (§14.2.1),

Handout Master (§13.3.3), Notes Slide (§13.3.5), Notes Master (§13.3.4), Slide (§13.3.8), Slide Layout (§13.3.9), Slide Master (§13.3.10) part in a PresentationML package.

[*Example*: The following Presentation part-relationship item contains relationships to two Embedded Package parts: one is a SpreadsheetML package, which is stored in the ZIP item embeddings/Worksheet1.xlsx, the other is a PresentationML package, which is stored in the ZIP item embeddings/Presentation2.pptx. The image files are used as document display placeholders if the consumer cannot handle the embedded package type:

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
    Type="http://…/image" Target="media/image1.emf"/>
  <Relationship Id="rId5"
    Type="http:package" Target="embeddings/Worksheet1.xlsx"/>
  <Relationship Id="rId6"
    Type="http://…/image" Target="media/image2.emf"/>
  <Relationship Id="rId7"
    Type="http://…/package" Target="embeddings/Presentation2.pptx"/>
</Relationships>
```

end example]

An Embedded Package part may be located within or external to the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element may be `Internal` or `External`).

An Embedded Package part is permitted to have an explicit relationship to the following parts defined by this Standard:

- Hyperlink (§15.3)

An Embedded Package part shall not have any implicit or explicit relationships to other parts defined by this Standard.

15.2.11 File Properties

There are three kinds of file properties: , *core*, *custom*, and *extended*. The *core file properties* of a package enable users to discover, get, and set common sets of properties from within that package, regardless of whether it's a WordprocessingML, SpreadsheetML, or PresentationML package. *Extended file properties* are specific to Office Open XML packages, while *custom file properties* are defined by the user, with each custom file property having a name, a value, and a type.

15.2.11.1 Core File Properties Part

Content Type:	application/vnd.openxmlformats-package.core-properties+xml
Root Namespace:	http://schemas.openxmlformats.org/package/2006/metadata/core-properties

Source Relationship:	http://schemas.openxmlformats.org/officedocument/2006/relationships/metadata/core-properties

Core file properties enable users to discover, get, and set common sets of properties within packages. (These properties include creator name, creation date, title, and description.) These properties are stored using the appropriate Dublin Core properties whenever possible.

A package shall contain at most one Core File Properties part, and that part shall be the target of a relationship in the package-relationship item for the document.

[*Example*: The following PresentationML's package-relationship item contains one relationship, for the Core File Properties part stored in the ZIP item core.xml):

```
<Relationships xmlns="…">
  <Relationship Id="rId3"
     Type="http://…/core-properties" Target="docProps/core.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be coreProperties.

[*Example*:

```
<cp:coreProperties xmlns:cp="…" xmlns:dc="…" >
  <dc:title>Example File</dc:title>
  <dc:creator>Tristan Davis</dc:creator>
  <cp:lastModifiedBy>Tristan Davis</cp:lastModifiedBy>
  <cp:revision>1</cp:revision>
</cp:coreProperties>
```

end example]

A Core File Properties part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Core File Properties part shall not have implicit or explicit relationships to other parts defined by this Standard.

15.2.11.2 Custom File Properties Part

Content Type:	application/vnd.openxmlformats-officedocument.custom-properties+xml
Root Namespace:	http://schemas.openxmlformats.org/officeDocument/2006/custom-properties

Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/custom-properties

An instance of this part contains the names of custom file properties that apply to the package, their values, and the types of those values. A custom file property might be the name of the client for whom the document was prepared, a date/time on which some event happened, a document number, or some Boolean status flag.

A package shall contain at most one Custom File Properties part, and that part shall be the target of a relationship in the package-relationship item for the document.

[*Example*: The following PresentationML's package-relationship item contains a relationship to a Custom File Properties part, stored in the ZIP item docProps/custom.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
  Type="http://…/custom-properties" Target="docProps/custom.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be Properties.

[*Example*: Here's some content markup from a WordprocessingML document, which contains four custom properties: `Client`, having a text value of "ACME Corp."; `Document number`, having a numeric value of 1543; `Recorded date`, having a date/time value of 2005-12-01; and `Special processing needed`, having a Boolean value of false:

```
<Properties … xmlns:vt="…">
  <property fmtid="{D5C…9AE}" pid="2" name="Client">
    <vt:lpwstr>ACME Corp.</vt:lpwstr>
  </property>
  <property fmtid="{D5C…9AE}" pid="3" name="Document number">
    <vt:i4>1543</vt:i4>
  </property>
  <property fmtid="{D5C…9AE}" pid="4" name="Recorded date">
    <vt:filetime>2005-12-01T05:00:00Z</vt:filetime>
  </property>
  <property fmtid="{D5C…9AE}" pid="5" name="Special processing needed">
    <vt:bool>false</vt:bool>
  </property>
</Properties>
```

end example]

A Custom File Properties part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Custom File Properties part shall not have implicit or explicit relationships to other parts defined by this Standard.

15.2.11.3 Extended File Properties Part

Content Type:	application/vnd.openxmlformats-officedocument.extended-properties+xml
Root Namespace:	http://schemas.openxmlformats.org/officeDocument/2006/extended-properties
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/extended-properties

An instance of this part contains properties specific to an Office Open XML document.[*Example*: A PresentationML document specifies the number of slides in this presentation when last saved by a producer. *end example*]

A package shall contain at most one Extended File Properties part, and that part shall be the target of a relationship in the package-relationship item for the document.

[*Example*:

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
    Type="http://…/extended-properties" Target="docProps/app.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be Properties.

[*Example*: Here's some content markup from a WordprocessingML document:

```
<Properties …>
  <Template>Normal.dotm</Template>
  <TotalTime>0</TotalTime>
  <Pages>1</Pages>
  <Words>3</Words>
  <Characters>22</Characters>
```

```
    <Application>Sample Producer</Application>
    <DocSecurity>0</DocSecurity>
    <Lines>1</Lines>
    <Paragraphs>1</Paragraphs>
    …
    <AppVersion>12.0000</AppVersion>
  </Properties>
```

here's some content markup from a SpreadsheetML document:

```
  <Properties …>
    <Application>Sample Producer</Application>
    <HeadingPairs>
    …
    </HeadingPairs>
    <TitlesOfParts>
    …
    </TitlesOfParts>
    <Company>Consultant</Company>
    …
  </Properties>
```

and here's some content markup from a PresentationML document:

```
  <Properties …>
    <Template>ppt_template_sdwest05</Template>
    <TotalTime>3166</TotalTime>
    <Words>37</Words>
    <Application>Sample Producer</Application>
    <PresentationFormat>On-screen Show</PresentationFormat>
    <Paragraphs>15</Paragraphs>
    <Slides>2</Slides>
    <Notes>2</Notes>
    …
    <HeadingPairs>
    …
    </HeadingPairs>
    <TitlesOfParts>
    …
    </TitlesOfParts>
    …
  </Properties>
```

end example]

A Extended File Properties part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

An Extended File Properties part shall not have implicit or explicit relationships to any other part defined by this Standard.

15.2.12 Font Part

Content Type:	application/x-fontdata application/x-font-ttf
Root Namespace:	not applicable
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/font

An instance of this part type contains a given font embedded directly into the document. (This is useful when using custom fonts or fonts that are not widely distributed.)

Fonts stored in a Font part can be stored in one of two formats, identified by the associated content type:

- `application/x-fontdata` specifies that the font shall be stored as a bitmapped font (each glyph is stored as a raster image)
- `application/x-font-ttf` specifies that the font shall be stored in the TrueType or OpenType format

A package shall contain zero or more Font parts, and for each that exists, that part shall be the target of an explicit relationship in the Font Table (§11.3.5), or Presentation (§13.3.6) part.

A Font part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Font part shall not have implicit or explicit relationships to other parts defined by this Standard.

15.2.13 Image Part

Content Type:	Any supported image type. [*Note*: Some example content types are:

image/gif	http://www.w3.org/Graphics/GIF/spec-gif89a.txt
image/png	ISO/IEC 15948:2003 http://www.libpng.org/pub/png/spec/
image/tiff	http://partners.adobe.com/public/developer/tiff/index.html#spec
image/pict	http://developer.apple.com/documentation/mac/QuickDraw/QuickDraw-2.html
image/jpeg	http://www.w3.org/Graphics/JPEG/

end note.]

Root Namespace:	Not applicable
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/image

An image can be stored in a package as a ZIP item. Image ZIP items shall be identified by an image part relationship and the appropriate content type.

A package is permitted to contain zero or more Image parts, and each such part shall be the target of an explicit relationship from a Comments (§11.3.2), Endnotes (§11.3.4), Footer (§11.3.6), Footnotes (§11.3.7), Header (§11.3.9), or Main Document (§11.3.10) part in a WordprocessingML package; a VML Drawing (§15.2.17) part in a SpreadsheetML package; or a Handout Master (§13.3.3), Notes Slide (§13.3.5), Notes Master (§13.3.4), Slide (§13.3.8), Slide Layout (§13.3.9), Slide Master (§13.3.10), or a VML Drawing (§15.2.17) part in a PresentationML package.

[*Example*: The following PresentationML's package-relationship item contains one relationship, for the slide template jpeg image stored in the ZIP item ../media/image1.jpeg:

```
<Relationships xmlns="…">
  <Relationship Id="rId8"
    Type="http://…/image" Target="../media/image1.jpeg"/>
</Relationships>
```

end example]

An Image part may be located within or external to the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element may be `Internal` or `External`).

An Image part shall not have implicit or explicit relationships to other parts defined by this Standard.

15.2.14 Printer Settings Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.printerSettings (in SpreadsheetML documents) application/vnd.openxmlformats-officedocument.wordprocessingml.printerSettings (in WordprocessingML documents)
Root Namespace:	not applicable
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/printerSettings

An instance of this part type contains information about the initialization and environment of a printer or a display device. The layout of this data structure is application-defined. [*Example*: An Office Open XML producer

on Windows might store the DEVMODE structure defined here: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdi/prntspol_8nle.asp, while an application on the Mac OS might choose to store the print record defined here: http://developer.apple.com/documentation/Carbon/Reference/CarbonPrintingManager_Ref/Reference/reference.html. *end example*]

A SpreadsheetML package is permitted to contain at most one Printer Settings part per Chartsheet, Dialogsheet, or Worksheet part, and that part shall be the target of an implicit relationship from a Chartsheet (§12.3.2), Dialogsheet (§12.3.7), or Worksheet (§12.3.24) part. A WordprocessingML package is permitted to contain zero or more Printer Settings parts, one per sectPr element, each a target of an explicit relationship from a Main Document (§11.3.10) or Glossary Document (§11.3.8) part.

[*Example*: The following SpreadsheetML Worksheet part-relationship item contains a relationship to a Printer Settings part, which is stored in the ZIP item ../printerSettings/printerSettings1.bin:

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
    Type="http://…/printerSettings"
    Target="../printerSettings/printerSettings1.bin"/>
</Relationships>
```

end example]

A Printer Settings part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Printer Settings part shall not have implicit or explicit relationships to any other part defined by this Standard.

15.2.15 Thumbnail Part

<table>
<tr><td>Content Type:</td><td>Any supported image type.

[Note: Some example content types are:
<table>
<tr><td>image/gif</td><td>http://www.w3.org/Graphics/GIF/spec-gif89a.txt</td></tr>
<tr><td>image/png</td><td>ISO/IEC 15948:2003 http://www.libpng.org/pub/png/spec/</td></tr>
<tr><td>image/tiff</td><td>http://partners.adobe.com/public/developer/tiff/index.html#spec</td></tr>
<tr><td>image/pict</td><td>http://developer.apple.com/documentation/mac/QuickDraw/QuickDraw-2.html</td></tr>
<tr><td>image/jpeg</td><td>http://www.w3.org/Graphics/JPEG/</td></tr>
</table>
end note.]</td></tr>
<tr><td>Root Namespace:</td><td>Not applicable</td></tr>
</table>

Source Relationship:	http://schemas.openxmlformats.org/package/2006/relationships/metadata/thumbnail

To help end-users identify parts of a package or the package as a whole, images, called *thumbnails*, may be stored in that package. Each thumbnail image is generated by the package producer and is stored in the package as a ZIP item.

Thumbnail ZIP items shall be identified by either a package-relationship item or a part-relationship item. Packages shall not contain more than one thumbnail relationship associated with the package as a whole, or more than one thumbnail relationship per package part.

[*Example*: The following PresentationML's package-relationship item contains one relationship, for the metafile image stored in the ZIP item thumbnail.wmf:

```
<Relationships xmlns="…">
  <Relationship Id="rId2"
    Type="http://…/thumbnail" Target="docProps/thumbnail.wmf"/>
</Relationships>
```

end example]

A Thumbnail part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Thumbnail part shall not have implicit or explicit relationships to other parts defined by this Standard.

15.2.16 Video Part

Content Type:	Any supported video type. [*Note*: Some example content types are: video/x-ms-asf — http://www.microsoft.com/windows/windowsmedia/forpros/format/asfspec.aspx video/avi — http://www.the-labs.com/Video/odmlff2-avidef.pdf video/mpg — ISO/IEC 13818 video/mpeg — ISO/IEC 13818 video/x-ms-wm — http://www.microsoft.com/windows/windowsmedia/forpros/format/asfspec.aspx video/quicktime — http://developer.apple.com/softwarelicensing/agreements/quicktime.html *end note*]
Root Namespace:	not applicable

Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/video

An instance of this part type contains a video file.

A PresentationML package is permitted to contain zero or more Video parts, each of which shall be the target of an explicit relationship in a Handout Master (§13.3.3), Notes Slide (§13.3.5), Notes Master (§13.3.4), Slide (§13.3.8), Slide Layout (§13.3.9), or Slide Master (§13.3.10) part. A WordprocessingML package is permitted to contain zero or more Video parts, each of which shall be the target of an explicit relationship from a Comments (§11.3.2), Endnotes (§11.3.4), Footer (§11.3.6), Footnotes (§11.3.7), Header (§11.3.9), or Main Document (§11.3.10) part.

[*Example*: The following Slide part-relationship item contains a relationship to a Video part, which is stored as the file E:\Video demo.avi:

```
<Relationships xmlns="…">
  <Relationship Id="rId2"
    Type="http://…/video"
    Target="file:///E:\Video%20demo.avi" TargetMode="External"/>
</Relationships>
```

end example]

A Video part is not stored as XML; instead, it involves a relationship target that is a video clip.

A Video part may be located within or external to the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element may be `Internal` or `External`).

A Video part shall not have implicit or explicit relationships to other parts defined by this Standard.

15.2.17 VML Drawing Part

Content Type:	application/vnd.openxmlformats-officedocument.vmlDrawing
Root Namespace:	not applicable
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/vmlDrawing

An instance of this part type contains markup in the Vector Markup Language (VML) syntax, which is used to provide an alternative image representation of objects stored in a SpreadsheetML or PresentationML document.

[*Note*: The VML format is a legacy format originally introduced with Office 2000 and is included and fully defined in this Standard for backwards compatibility reasons. The DrawingML format is a newer and richer

format created with the goal of eventually replacing any uses of VML in the Office Open XML formats. VML should be considered a deprecated format included in Office Open XML for legacy reasons only and new applications that need a file format for drawings are strongly encouraged to use preferentially DrawingML. *end note*]

A package is permitted to contain zero or more VML Drawing parts, each of which shall be the target of an explicit relationship in a Handout Master (§13.3.3), Notes Slide (§13.3.5), Notes Master (§13.3.4), Slide (§13.3.8), Slide Layout (§13.3.9), or Slide Master (§13.3.10) part in a PresentationML document; or a Worksheet part (§12.3.24) in a SpreadsheetML document.

[*Example*: The following SpreadsheetML's package-relationship item contains one relationship, for the VML Drawing part stored in the ZIP item ../drawings/drawing1.vml:

```
<Relationships xmlns="…">
  <Relationship Id="rId8"
    Type="http://…/vmlDrawing" Target="../drawings/drawing1.vml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be xml in the null namespace, encapsulating an arbitrary amount of VML markup as defined by this Standard.

[*Example*: Consider the following VML Drawing part:

```
<xml>
  <v:shape ...>
    ...
  </v:shape>
  ...
</xml>
```

end example]

A VML Drawing part shall be located within the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A VML Drawing part is permitted to have explicit relationships to the following parts defined by this Standard:

- Image (§15.2.13)

A VML Drawing part shall not have implicit or explicit relationships to any other part defined by this Standard.

15.3 Hyperlinks

Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/hyperlink

A hyperlink can be stored in a package as a relationship. Hyperlinks shall be identified by containing a target which specifies the destination of the given hyperlink.

[*Example*: The following WordprocessingML Footnote part's relationship part contains one relationship, for the hyperlink http://schemas.openxmlformats.org/wordprocessingml/:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/hyperlink"
    Target="http://schemas.openxmlformats.org/wordprocessingml/"
    TargetMode="External"/>
</Relationships>
```

end example]

A hyperlink target may be located within or external to the package containing the source relationship (expressed syntactically, the TargetMode attribute of the Relationship element may be `Internal` or `External`).

Annex A. Bibliography

This clause is informative.

Character Sets from IANA, as specified at http://www.iana.org/assignments/character-sets

PANOSE Classification Guide, Version 1.2, Hewlett Packard Co., 1992.

RFC 2119, Bradner, Scott, 1997: "*Key words for use in RFCs to Indicate Requirement Levels.*" http://www.ietf.org/rfc/rfc2119.txt.

RFC 2045, Borenstein, N., and N. Freed. "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies." The Internet Society. 1996. http://www.rfc-editor.org

RFC 2616, Berners-Lee, T., R. Fielding, H. Frystyk, J. Gettys, P. Leach, L. Masinter, and J. Mogul. "Hypertext Transfer Protocol—HTTP/1.1." The Internet Society. 1999. http://www.rfc-editor.org

RFC 3066, Alvestrand, H. "Tags for the Identification of Languages." The Internet Society. 2001. http://www.rfc-editor.org

RFC 3339, Klyne, G. and C. Newman "Date and Time on the Internet: Timestamps." The Internet Society. 2002. http://www.rfc-editor.org

RFC 3629, Yergeau, F. "UTF-8, a transformation format of ISO 10646." The Internet Society. 2003. http://www.rfc-editor.org

RFC 3986, Berners-Lee, T., R. Fielding, and L. Masinter. "Uniform Resource Identifier (URI): Generic Syntax." The Internet Society. 2005. http://www.rfc-editor.org

The Unicode Consortium. The Unicode Standard, Version 4.0, defined by: The Unicode Standard, Version 4.0 (Reading, MA, Addison-Wesley, 2003. ISBN 0-321-18578-1).

XML, Bray, Tim, Eve Maler, Jean Paoli, C. M. Sperberg-McQueen, and François Yergeau (editors). "Extensible Markup Language (XML) 1.0," Third Edition. World Wide Web Consortium. 2004. http://www.w3.org/TR/2004/REC-xml-20040204/xml11-20040204/)

XML Base, Marsh, Jonathan. "XML Base." World Wide Web Consortium. 2001. http://www.w3.org/TR/2001/REC-xmlbase-20010627/

XML Namespaces, Bray, Tim, Dave Hollander, Andrew Layman, and Richard Tobin (editors). "Namespaces in XML 1.1." World Wide Web Consortium. 2004. http://www.w3.org/TR/2004/REC-xml-names11-20040204/

XML Path Language Specification, Version 1.0, W3C Recommendation 16 November 1999 http://www.w3.org/TR/xpath

XML Schema Part 0: Primer Second Edition, W3C Recommendation 28 October 2004
http://www.w3.org/TR/xmlschema-0/

XML Schema Part 1: Structures Second Edition, W3C Recommendation 28 October 2004
http://www.w3.org/TR/xmlschema-1/

XML Schema Part 2: Datatypes Second Edition, W3C Recommendation 28 October 2004
http://www.w3.org/TR/xmlschema-2/

.ZIP File Format Specification from PKWARE, Inc., as specified in *appnote*, the Application Note on the Zip file format, at http://www.pkware.com.

End of informative text.

Annex B. Index

This annex is informative.

Office Open XML

Part 2: Open Packaging Conventions

December 2006

Table of Contents

Foreword

This multi-part Standard deals with Office Open XML Format-related technology, and consists of the following parts:

- Part 1: "Fundamentals"
- **Part 2: "Open Packaging Conventions" (this document)**
- Part 3: "Primer"
- Part 4: "Markup Language Reference"
- Part 5: "Markup Compatibility and Extensibility"

This part, Part 2, includes a number of annexes that refer to data files provided in electronic form only.

1. Scope

This Part (the *Open Packaging Conventions specification*) specifies a set of conventions that are used by Office Open XML documents to define the structure and functionality of a *package* in terms of a package model and a physical model.

The *package model* defines a package abstraction that holds a collection of *parts*. The parts are composed, processed, and persisted according to a set of rules. Parts can have relationships to other parts or external resources, and the package as a whole can have relationships to parts it contains or external resources. The package model specifies how the parts of a package are named and related. Parts have content types and are uniquely identified using the well-defined naming guidelines provided in this Open Packaging specification.

The *physical mapping* defines the mapping of the components of the package model to the features of a specific physical format, namely a ZIP archive.

This Open Packaging Conventions specification also describes certain features that might be supported in a package, including *core properties* for package metadata, a *thumbnail* for graphical representation of a package, and *digital signatures* of package contents.

Because this Standard will continue to evolve, packages are designed to accommodate extensions and support compatibility goals in a limited way. The versioning and extensibility mechanisms described in Part 5: "Markup Compatibility and Extensibility" support compatibility between software systems based on different versions of this Standard while allowing package creators to make use of new or proprietary features.

This Open Packaging Conventions specification specifies requirements for package implementers, producers, and consumers.

In all subsequent uses, the term "this specification" shall refer to the content of this Part.

2. Normative References

The following normative documents contain provisions, which, through reference in this text, constitute provisions of this Open Packaging specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this Open Packaging specification are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO 8601, *Data elements and interchange formats — Information interchange — Representation of dates and times*.

ISO/IEC 9594-8 *Public-key and attribute certificate frameworks* (x.509 Certificate).

ISO/IEC 10646 (all parts), *Information technology — Universal Multiple-Octet Coded Character Set (UCS)*.

3. Definitions

For the purposes of this Open Packaging specification, the following definitions apply. Other terms are defined where they appear in italic type. Terms explicitly defined in this Open Packaging specification are not to be presumed to refer implicitly to similar terms defined elsewhere.

The terms *base URI* and *relative reference* are used in accordance with RFC 3986.

access style — The style in which local access or networked access is conducted. The access styles are as follows: streaming creation, streaming consumption, simultaneous creation and consumption, and direct access consumption.

behavior — External appearance or action.

behavior, implementation-defined — Unspecified behavior where each implementation shall document that behavior, thereby promoting predictability and reproducibility within any given implementation. (This term is sometimes called “application-specific behavior”.)

behavior, unspecified —Behavior where this Open Packaging specification imposes no requirements.

communication style — The style in which package contents are delivered by a producer or received by a consumer. Communication styles include: random access and sequential delivery.

consumer — A piece of software or a device that reads packages through a package implementer. A consumer is often designed to consume packages only for a specific physical package format.

content type — Describes the content stored in a part. Content types define a media type, a subtype, and an optional set of parameters, as defined in RFC 2616.

Content Types stream — A specially-named stream that defines mappings from part names to content types. The content types stream is not itself a part, and is not URI addressable.

device — A piece of hardware, such as a personal computer, printer, or scanner, that performs a single function or set of functions.

format consumer — A consumer that consumes packages conforming to a format designer's specification.

format designer — The author of a particular file format specification built on this Open Packaging Conventions specification.

format producer — A producer that produces packages conforming to a format designer's specification.

growth hint — A suggested number of bytes to reserve for a part to grow in-place.

interleaved ordering — The layout style of a physical package where parts are broken into pieces and "mixed-in" with pieces from other parts. When delivered, interleaved packages can help improve the performance of the consumer processing the package.

layout style — The style in which the collection of parts in a physical package is laid out: either simple ordering or interleaved ordering.

local access — The access architecture in which a pipe carries data directly from a producer to a consumer on a single device.

logical item name — An abstraction that allows package implementers to manipulate physical data items consistently regardless of whether those data items can be mapped to parts or not or whether the package is laid out with simple ordering or interleaved ordering.

networked access — The access architecture in which a consumer and the producer communicate over a protocol, such as across a process boundary, or between a server and a desktop computer.

pack URI — A URI scheme that allows URIs to be used as a uniform mechanism for addressing parts within a package. Pack URIs are used as Base URIs for resolving relative references among parts in a package.

package — A logical entity that holds a collection of parts.

package implementer — Software that implements the physical input-output operations to a package according to the requirements and recommendations of this Open Packaging specification. A package implementer is used by a producer or consumer to interact with a physical package. A package implementer may be either a stand-alone API or may be an integrated component of a producer, consumer application, or device.

package model — A package abstraction that holds a collection of parts.

package relationship — A relationship whose target is a part and whose source is the package as a whole. Package relationships are found in the package relationships part named "/_rels/.rels".

part — A stream of bytes with a MIME content type and associated common properties. Typically corresponds to a file [*Example*: on a file system *end example*], a stream [*Example*: in a compound file *end example*], or a resource [*Example*: in an HTTP URI *end example*].

part name — The path component of a pack URI. Part names are used to refer to a part in the context of a package, typically as part of a URI.

physical model — A description of the capabilities of a particular physical format.

physical package format — A specific file format, or other persistence or transport mechanism, that can represent all of the capabilities of a package.

piece — A portion of a part. Pieces of different parts may be interleaved together. The individual pieces are named using a unique mapping from the part name. Piece name grammar is not equivalent to the part name grammar. Pieces are not addressable in the package model.

pipe — A communication mechanism that carries data from the producer to the consumer.

producer — A piece of software or a device that writes packages through a package implementer. A producer is often designed to produce packages according to a particular physical package format specification.

random access — A style of communication between the producer and the consumer of the package. Random access allows the consumer to reference and obtain data from anywhere within a package.

relationship —The kind of connection between a source part and a target part in a package. Relationships make the connections between parts directly discoverable without looking at the content in the parts, and without altering the parts themselves. (See also Package Relationships.)

relationships part — A part containing an XML representation of relationships.

sequential delivery — A communication style in which all of the physical bits in the package are delivered in the order they appear in the package.

signature policy — A format-defined policy that specifies what configuration of parts and relationships shall or might be included in a signature for that format and what additional behaviors that producers and consumers of that format shall follow when applying or verifying signatures following that format's signature policy.

simple ordering — A defined ordering for laying out the parts in a package in which all the bits comprising each part are stored contiguously.

simultaneous creation and consumption — A style of access between a producer and a consumer in highly pipelined environments where streaming creation and streaming consumption occur simultaneously.

stream — A linearly ordered sequence of bytes.

streaming consumption — An access style in which parts of a physical package may be processed by a consumer before all of the bits of the package have been delivered through the pipe.

streaming creation — A production style in which a producer dynamically adds parts to a package after other parts have been added without modifying those parts.

thumbnail — A small image that is a graphical representation of a part or the package as a whole.

well-known part — A part with a well-known relationship, which enables the part to be found without knowing the location of other parts.

ZIP archive — A ZIP file as defined in the ZIP file format specification. A ZIP archive contains ZIP items.

ZIP item — A ZIP item is an atomic set of data in a ZIP archive that becomes a file when the archive is uncompressed. When a user unzips a ZIP based package, the user sees an organized set of files and folders.

4. Notational Conventions

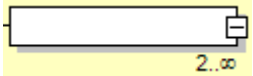
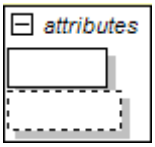
4.1 Document Conventions

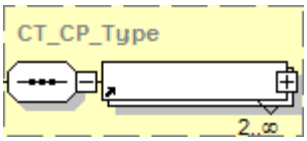
The following typographical conventions are used in this Standard:

1. The first occurrence of a new term is written in *italics*. [*Example*: … is considered *normative*. *end example*]
2. A term defined as a basic definition is written in **bold**. [*Example*: **behavior** — External … *end example*]
3. The name of an XML element is written using an Element style. [*Example*: The root element is document. *end example*]
4. The name of an XML element attribute is written using an Attribute style. [*Example*: … an id attribute. *end example*]
5. An XML element attribute value is written using a `constant-width` style. [*Example*: … value of `CommentReference`. *end example*]
6. An XML element type name is written using a Type style. [*Example*: … as values of the xsd:anyURI data type. *end example*]

4.2 Diagram Notes

In some cases, markup semantics are described using diagrams. The diagrams place the parent element on the left, with attributes and child elements to the right. The symbols are described below.

Symbol	Description
	Required element: This box represents an element that shall appear exactly once in markup when the parent element is included. The "+" and "–" symbols on the right of these boxes have no semantic meaning.
	Optional element: This box represents an element that shall appear zero or one times in markup when the parent element is included.
	Range indicator: These numbers indicate that the designated element or choice of elements can appear in markup any number of times within the range specified.
	Attribute group: This box indicates that the enclosed boxes are each attributes of the parent element. Solid-border boxes are required attributes; dashed-border boxes are optional attributes.

Symbol	Description
	Sequence symbol: The element boxes connected to this symbol shall appear in markup in the illustrated sequence only, from top to bottom.
	Choice symbol: Only one of the element boxes connected to this symbol shall appear in markup.
	Type indicator: The elements within the dashed box are of the complex type indicated.

5. Acronyms and Abbreviations

This clause is informative.

The following acronyms and abbreviations are used throughout this specification:

IEC — the International Electrotechnical Commission

ISO — the International Organization for Standardization

W3C — World Wide Web Consortium

End of informative text.

6. General Description

This Open Packaging specification is intended for use by implementers, academics, and application programmers. As such, it contains a considerable amount of explanatory material that, strictly speaking, is not necessary in a formal specification.

This Open Packaging specification is divided into the following subdivisions:

1. Front matter (clauses 1–7);
2. Overview (clause 8);
3. Main body (clauses 9-13);
4. Annexes

Examples are provided to illustrate possible forms of the constructions described. References are used to refer to related clauses. Notes are provided to give advice or guidance to implementers or programmers. Annexes provide additional information and summarize the information contained in this Open Packaging specification.

The following form the normative part of this Open Packaging specification:

- Introduction
- Clauses 1–4, 6, and 8–12
- Annex A–Annex D
- Annex F

The following form the informative part of this Open Packaging specification:

- Clauses 5 and 7
- Annex E
- Annex G–Annex J
- All notes
- All examples

Whole clauses and annexes that are informative are identified as such. Informative text that is contained within normative text is identified as either an example, or a note as specified in 4.1, “Document Conventions.”

7. Overview

This clause is informative.

This Open Packaging specification describes an abstract model and physical format conventions for the use of XML, Unicode, ZIP, and other openly available technologies and specifications to organize the content and resources of a document within a package. It is intended to support the content types and organization for various applications and is written for developers who are building systems that process package content.

In addition, this Open Packaging specification defines common services that can be included in a package, such as Core Properties and Digital Signatures.

A primary goal is to ensure the interoperability of independently created software and hardware systems that produce or consume package content and use common services. This Open Packaging specification defines the formal requirements that producers and consumers shall satisfy in order to achieve interoperability.

Various XML-based building blocks within a package make use of the conventions described in Part 5: “Markup Compatibility and Extensibility” to facilitate future enhancement and extension of XML markup. That part shall be explicitly cited by any markup specification that bases its versioning and extensibility strategy on Markup Compatibility elements and attributes.

End of informative text.

8. Package Model

A *package* is a logical entity that holds a collection of parts. The purpose of the package is to aggregate all of the pieces of a document (or other type of content) into a single object. [*Example*: A package holding a document with a picture might contain two parts: an XML markup part representing the document and another part representing the picture. *end example*] The package is also capable of storing relationships between parts.

The package provides a convenient way to distribute documents with all of their component pieces, such as images, fonts, and data. Although this Open Packaging specification defines a single-file package format, the package model allows for the future definition of other physical package representations. [*Example:* A package could be physically represented in a collection of loose files, in a database, or ephemerally in transit over a network connection. *end example*]

This Open Packaging specification also defines a URI scheme, the *pack URI*, that allows URIs to be used as a uniform mechanism for addressing parts within a package.

8.1 Parts

A *part* is a stream of bytes with the properties listed in Table 8–1. A *stream* is a linearly ordered sequence of bytes. Parts are analogous to a file in a file system or to a resource on an HTTP server.

Table 8–1. Part properties

Name	Description	Required/Optional
Name	The name of the part	Required. The package implementer shall require a part name. [M1.1]
Content Type	The type of content stored in the part	Required. The package implementer shall require a content type and the format designer shall specify the content type. [M1.2]
Growth Hint	A suggested number of bytes to reserve for the part to grow in-place	Optional. The package implementer might allow a growth hint to be provided by a producer. [O1.1]

8.1.1 Part Names

Each part has a name. *Part names* refer to parts within a package. [*Example:* The part name "/hello/world/doc.xml" contains three segments: "hello", "world", and "doc.xml". The first two segments in the sample represent levels in the logical hierarchy and serve to organize the parts of the package, whereas the

third contains actual content. Note that segments are not explicitly represented as folders in the package model, and no directory of folders exists in the package model. *end example*]

Part Name Syntax

The part name grammar is defined as follows:

```
part_name = 1*( "/" segment )
segment   = 1*( pchar )
```

`pchar` is defined in RFC 3986.

The part name grammar implies the following constraints. The package implementer shall neither create any part that violates these constraints nor retrieve any data from a package as a part if the purported part name violates these constraints.

- A part name shall not be empty. [M1.1]
- A part name shall not have empty segments. [M1.3]
- A part name shall start with a forward slash (“/”) character. [M1.4]
- A part name shall not have a forward slash as the last character. [M1.5]
- A segment shall not hold any characters other than pchar characters. [M1.6]

Part segments have the following additional constraints. The package implementer shall neither create any part with a part name comprised of a segment that violates these constraints nor retrieve any data from a package as a part if the purported part name contains a segment that violates these constraints.

- A segment shall not contain percent-encoded forward slash (“/”), or backward slash (“\”) characters. [M1.7]
- A segment shall not contain percent-encoded unreserved characters. [M1.8]
- A segment shall not end with a dot (“.”) character. [M1.9]
- A segment shall include at least one non-dot character. [M1.10]

[*Example:*

Example 8–1. A part name

```
/a/%D1%86.xml
/xml/item1.xml
```

Example 8–2. An invalid part name

```
//xml/.
```

end example]

8.1.1.1 Part Naming

A package implementer shall neither create nor recognize a part with a part name derived from another part name by appending segments to it. [M1.11] [*Example*: If a package contains a part named "/segment1/segment2/.../segment*n*", then other parts in that package shall not have names such as: "/segment1", "segment1/segment2", or "/segment1/segment2/.../segment*n*-1". *end example]*

8.1.1.2 Part Name Equivalence

Part name equivalence is determined by comparing part names as case-insensitive ASCII strings. Packages shall not contain equivalent part names and package implementers shall neither create nor recognize packages with equivalent part names. [M1.12]

8.1.2 Content Types

Every part has a *content type*, which identifies the type of content that is stored in the part. Content types define a media type, a subtype, and an optional set of parameters. Package implementers shall only create and only recognize parts with a content type; format designers shall specify a content type for each part included in the format. Content types for package parts shall fit the definition and syntax for media types as specified in RFC 2616, §3.7. [M1.13] This definition is as follows:

```
media-type = type "/" subtype *( ";" parameter )
```

where parameter is expressed as

```
attribute "=" value
```

The type, subtype, and parameter attribute names are case-insensitive. Parameter values may be case-sensitive, depending on the semantics of the parameter attribute name.

Content types shall not use linear white space either between the type and subtype or between an attribute and its value. Content types also shall not have leading or trailing white spaces. Package implementers shall create only such content types and shall require such content types when retrieving a part from a package; format designers shall specify only such content types for inclusion in the format. [M1.14]

The package implementer shall require a content type that does not include comments and the format designer shall specify such a content type. [M1.15]

Format designers might restrict the usage of parameters for content types. [O1.2]

Content types for package-specific parts are defined in Annex F, "Standard Namespaces and Content Types."

8.1.3 Growth Hint

Sometimes a part is modified after it is placed in a package. Depending on the nature of the modification, the part might need to grow. For some physical package formats, this could be an expensive operation and could damage an otherwise efficiently interleaved package. Ideally, the part should be allowed to grow in-place, moving as few bytes as possible.

To support these scenarios, a package implementer can associate a growth hint with a part. [O1.1] The *growth hint* identifies the number of bytes by which the producer predicts that the part will grow. In a mapping to a particular physical format, this information might be used to reserve space to allow the part to grow in-place. This number serves as a hint only. The package implementer might ignore the growth hint or adhere only loosely to it when specifying the physical mapping. [O1.3] If the package implementer specifies a growth hint, it is set when a part is created and the package implementer shall not change the growth hint after the part has been created. [M1.16]

8.1.4 XML Usage

All XML content of the parts defined in this Open Packaging specification shall conform to the following validation rules:

1. XML content shall be encoded using either UTF-8 or UTF-16. If any part includes an encoding declaration, as defined in §4.3.3 of the XML 1.0 specification, that declaration shall not name any encoding other than UTF-8 or UTF-16. Package implementers shall enforce this requirement upon creation and retrieval of the XML content. [M1.17]
2. The XML 1.0 specification allows for the usage of Document Type Definitions (DTDs), which enable Denial of Service attacks, typically through the use of an internal entity expansion technique. As mitigation for this potential threat, DTD declarations shall not be used in the XML markup defined in this Open Packaging specification. Package implementers shall enforce this requirement upon creation and retrieval of the XML content and shall treat the presence of DTD declarations as an error. [M1.18]
3. If the XML content contains the Markup Compatibility namespace, as described in Part 5: “Markup Compatibility and Extensibility”, it shall be processed by the package implementer to remove Markup Compatibility elements and attributes, ignorable namespace declarations, and ignored elements and attributes before applying subsequent validation rules. [M1.19]
4. XML content shall be valid against the corresponding XSD schema defined in this Open Packaging specification. In particular, the XML content shall not contain elements or attributes drawn from namespaces that are not explicitly defined in the corresponding XSD unless the XSD allows elements or attributes drawn from any namespace to be present in particular locations in the XML markup. Package implementers shall enforce this requirement upon creation and retrieval of the XML content. [M1.20]
5. XML content shall not contain elements or attributes drawn from “xml” or “xsi” namespaces unless they are explicitly defined in the XSD schema or by other means described in this Open Packaging specification. Package implementers shall enforce this requirement upon creation and retrieval of the XML content. [M1.21]

8.2 Part Addressing

Parts often contain references to other parts. [*Example*: A package might contain two parts: an XML markup file and an image. The markup file holds a reference to the image so that when the markup file is processed, the associated image can be identified and located. *end example*.*]*

8.2.1 Relative References

A relative reference is expressed so that the address of the referenced part is determined relative to the part containing the reference.

Relative references from a part are interpreted relative to the base URI of that part. By default, the base URI of a part is derived from the name of the part, as defined in §B.3.

If the format designer permits it, parts can contain Unicode strings representing references to other parts. If allowed by the format designer, format producers can create such parts and format consumers shall consume them. [O1.4] In particular, XML markup might contain Unicode strings referencing other parts as values of the xsd:anyURI data type. Format consumers shall convert these Unicode strings to URIs, as defined in Annex A, "Resolving Unicode Strings to Part Names," before resolving them relative to the base URI of the part containing the Unicode string. [M1.23]

Some types of content provide a way to override the default base URI by specifying a different base in the content. [*Example*: XML Base or HTML *end example]*. In the presence of one of these overrides, format consumers shall use the specified base URI instead of the default. [M1.24]

[*Example:*

Example 8–3. Part names and relative references

A package includes parts with the following names:

- /markup/page.xml
- /images/picture.jpg
- /images/other_picture.jpg

If /markup/page.xml contains a reference to ../images/picture.jpg, then this reference is interpreted as referring to the part name /images/picture.jpg.

end example]

8.2.2 Fragments

Sometimes it is useful to address a portion of or a specific point in a part. In URIs, a fragment identifier is used for this purpose. (See RFC 3986.)

[*Example*: In an XML part a fragment identifier might identify a portion of the XML content using an XPath expression. *end example]*

8.3 Relationships

Parts often contain references to other parts in the package and to resources outside of the package. In general, these references are represented inside the referring part in ways that are specific to the content type of the part, that is, in arbitrary markup or an application-specific encoding. This effectively hides the internal and

external links between parts from consumers that do not understand the content types of the parts containing such references.

The package introduces a higher-level mechanism to describe references from parts to other internal or external resources: relationships. *Relationships* represent the type of connection between a source part and a target resource. They make the connection directly discoverable without looking at the part contents, so they are independent of content-specific schemas and quick to resolve.

Relationships provide a second important function: relating parts without modifying their content. Sometimes relationships act as a label where the content type of the labeled part does not define a way to attach the given information. Some scenarios require information to be attached to an existing part without modifying that part, either because the part is encrypted and cannot be decrypted, or because it is digitally signed and changing it would invalidate the signature.

8.3.1 Relationships Part

Each set of relationships sharing a common source is represented by XML stored in a *Relationships part*. The Relationships part is URI-addressable and it can be opened, read, and deleted. The Relationships part shall not have relationships to any other part. Package implementers shall enforce this requirement upon the attempt to create such a relationship and shall treat any such relationship as invalid. [M1.25]

The content type of the Relationships part is defined in Annex F, “Standard Namespaces and Content Types".

8.3.2 Package Relationships

A relationship whose source is a package as a whole is known as a *package relationship*. Package relationships are used to identify the “starting” parts in a package for a given context. This method avoids relying on naming conventions for finding parts in a package.

8.3.3 Relationship Markup

Relationships are represented using Relationship elements nested in a single Relationships element. These elements are defined in the Relationships namespace, as specified in Annex F, “Standard Namespaces and Content Types". The schema for relationships is described in Annex D, "Schemas - XML Schema".

The package implementer shall require that every Relationship element has an Id attribute, the value of which is unique within the Relationships part, and that the Id type is xsd:ID, the value of which conforms to the naming restrictions for xsd:ID as described in the W3C Recommendation “XML Schema Part 2: Datatypes.” [M1.26]

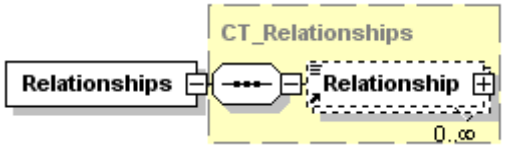
The nature of a Relationship element is identified by the Type attribute. Relationship Type is defined in the same way that namespaces are defined for XML namespaces. By using types patterned after the Internet domain-name space, non-coordinating parties can safely create non-conflicting relationship types.

Relationship types can be compared to determine whether two Relationship elements are of the same type. This comparison is conducted in the same way as when comparing URIs that identify XML namespaces: the two URIs are treated as strings and considered identical if and only if the strings have the same sequence of characters. The comparison is case-sensitive and no escaping is done or undone.

The Target attribute of the Relationship element holds a URI that points to a target resource. Where the URI is expressed as a relative reference, it is resolved against the base URI of the Relationships source part. The xml:base attribute shall not be used to specify a base URI for relationship XML content.

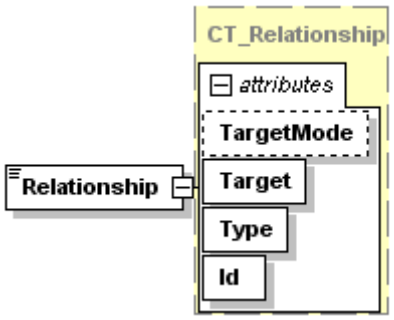
8.3.3.1 Relationships Element

The structure of a Relationships element is shown in the following diagram:

diagram	
annotation	The root element of the Relationships part.

8.3.3.2 Relationship Element

The structure of a Relationship element is shown in the following diagram:

diagram						
attributes	Name	Type	Use	Default	Fixed	Annotation
	TargetMode	ST_TargetMode	optional			The package implementer might allow a TargetMode to be provided by a producer. [O1.5] The TargetMode indicates whether or not the target describes a resource inside the package or outside the package. The valid values are “Internal” and “External”. The default value is Internal. When set to Internal, the Target attribute shall be a relative reference and that

						reference is interpreted relative to the “parent” part. For package relationships, the package implementer shall resolve relative references in the Target attribute against the pack URI that identifies the entire package resource. [M1.29] For more information, see Annex B, “Pack URI.” When set to External, the Target attribute may be a relative reference or a URI. If the Target attribute is a relative reference, then that reference is interpreted relative to the location of the package.
	Target	xsd:anyURI	required			The package implementer shall require the Target attribute to be a URI reference pointing to a target resource. The URI reference shall be a URI or a relative reference. [M1.28] Target attribute values are dependent on the TargetMode attribute value.
	Type	xsd:anyURI	required			The package implementer shall require the Type attribute to be a URI that defines the role of the relationship and the format designer shall specify such a Type. [M1.27]
	Id	xsd:ID	required			The package implementer shall require a valid XML identifier. [M1.26] The Id type is xsd:ID and it shall conform to the naming restrictions for xsd:ID as specified in the W3C Recommendation “XML Schema Part 2: Datatypes.” The value of the Id

						attribute shall be unique within the Relationships part.
annotation	Represents a single relationship.					

A format designer might allow fragment identifiers in the value of the Target attribute of the Relationship element. [O1.6] If a fragment identifier is allowed in the Target attribute of the Relationship element, a package implementer shall not resolve the URI to a scope less than an entire part. [M1.32]

8.3.4 Representing Relationships

Relationships are represented in XML in a Relationships part. Each part in the package that is the source of one or more relationships can have an associated Relationships part. This part holds the list of relationships for the source part. For more information on the Relationships namespace and relationship types, see Annex F, "Standard Namespaces and Content Types."

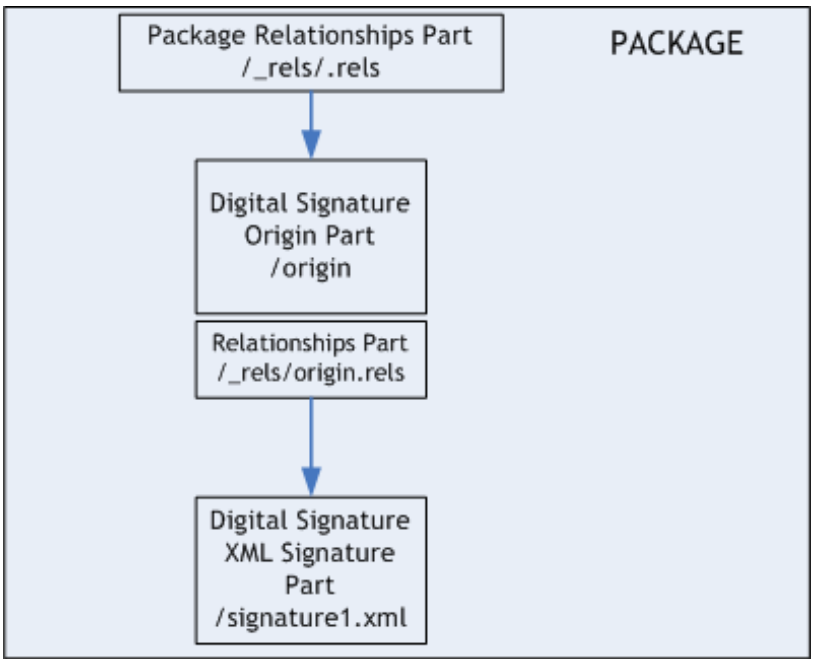
A special naming convention is used for the Relationships part. First, the Relationships part for a part in a given folder in the name hierarchy is stored in a sub-folder called "_rels". Second, the name of the Relationships part is formed by appending ".rels" to the name of the original part. Package relationships are found in the package relationships part named "/_rels/.rels".

The package implementer shall name relationship parts according to the special relationships part naming convention and require that parts with names that conform to this naming convention have the content type for a Relationships part. [M1.30]

[*Example:*

Example 8–4. Sample relationships and associated markup

The figure below shows a Digital Signature Origin part and a Digital Signature XML Signature part. The Digital Signature Origin part is targeted by a package relationship. The connection from the Digital Signature Origin to the Digital Signature XML Signature part is represented by a relationship.



The relationship targeting the Digital Signature Origin part is stored in /_rels/.rels and the relationship for the Digital Signature XML Signature part is stored in /_rels/origin.rels.

The Relationships part associated with the Digital Signature Origin contains a relationship that connects the Digital Signature Origin part to the Digital Signature XML Signature part. This relationship is expressed as follows:

```
<Relationships
   xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
   <Relationship
      Target="./Signature.xml"
      Id="A5FFC797514BC"
      Type="http://schemas.openxmlformats.org/package/2006/relationships/
         digital-signature/signature"/>
</Relationships>
```

end example]

[*Example:*

Example 8–5. Targeting resources

Relationships can target resources outside of the package at an absolute location and resources located relative to the current location of the package. The following Relationships part specifies relationships that connect a part to pic1.jpg at an external absolute location, and to my_house.jpg at an external location relative to the location of the package:

```
<Relationships
   xmlns="http://schemas.openxmlformats.org/package/2006/relationships"
```

```
    <Relationship
       TargetMode="External"
       Id="A9EFC627517BC"
       Target="http://www.custom.com/images/pic1.jpg"
       Type="http://www.custom.com/external-resource"/>
    <Relationship
       TargetMode="External"
       Id="A5EFC797514BC"
       Target="./images/my_house.jpg"
       Type="http://www.custom.com/external-resource"/>
 </Relationships>
```

end example]

[*Example:*

Example 8–6. Re-using attribute values

The following Relationships part contains two relationships, each using unique Id values. The relationships share the same Target, but have different relationship types.

```
 <Relationships
    xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
    <Relationship
       Target="./Signature.xml"
       Id="A5FFC797514BC"
       Type="http://schemas.openxmlformats.org/package/2006/
          relationships/digital-signature/signature"/>
    <Relationship
       Target="./Signature.xml"
       Id="B5F32797CC4B7"
       Type="http://www.custom.com/internal-resource"/>
 </Relationships>
```

end example]

8.3.5 Support for Versioning and Extensibility

Producers might generate relationship markup that uses the versioning and extensibility mechanisms defined in Part 5: “Markup Compatibility and Extensibility” to incorporate elements and attributes drawn from other XML namespaces. [O1.7]

Consumers shall process relationship markup in a manner that conforms to Part 5: “Markup Compatibility and Extensibility”. Producers editing relationships based on this version of the relationship markup specification shall not preserve any ignored content, regardless of the presence of any preservation attributes as defined in Part 5: “Markup Compatibility and Extensibility”. [M1.31]

9. Physical Package

In contrast to the package model that describes the contents of a package in an abstract way, the physical package refers to a package that is stored in a particular physical file format. This includes the physical model and physical mapping considerations.

The *physical model* abstractly describes the capabilities of a particular physical format and how producers and consumers can use a package implementer to interact with that physical package format. The physical model includes the *access style*, or the manner in which package input-output is conducted, as well as the *communication style*, which describes the method of interaction between producers and consumers across a communications *pipe*. The physical model also includes the *layout style*, or how part contents are physically stored within the package. The layout style can either be *simple ordering*, where the parts are arranged contiguously as atomic blocks of data, or *interleaved ordering*, where the parts are broken into individual pieces and the pieces are stored as interleaved blocks of data in an optimized fashion. The performance of a physical package design is reliant upon the physical model capabilities.

[*Note:* See Annex G, “Physical Model Design Considerations” for additional discussion of the physical model. *end note]*

Physical mappings describe the manner in which the package contents are mapped to the features of that specific physical format. Details of how package components are mapped are described, as well as common mapping patterns and mechanisms for storing part content types. This Open Packaging specification describes both the specific considerations for physical mapping to a ZIP archive as well as generic physical mapping considerations applicable to any physical package format.

9.1 Physical Mapping Guidelines

Whereas the package model defines a package abstraction, an *instance* of a package must be based on a physical representation. A *physical package format* is a particular physical representation of the package contents in a file.

Many physical package formats have features that partially match the packaging model components. In defining mappings from the package model to a physical package format, it is advisable to take advantage of any similarities in capabilities between the package model and the physical package medium while using layers of mapping to provide additional capabilities not inherently present in the physical package medium. [*Example*: Some physical package formats store parts as individual files in a file system, in which case it is advantageous to map many part names directly to identical physical file names. *end example]*

Designers of physical package formats face some common mapping problems. [*Example*: Associating arbitrary content types with parts and supporting part interleaving *end example]* Package implementers might use the common mapping solutions defined in this Open Packaging specification. [O2.3]

9.1.1 Mapped Components

The package implementer shall define a physical package format with a mapping for the required components package, part name, part content type and part contents. [M2.2] [*Note:* Not all physical package formats support the part growth hint. *end note*]

Table 9–1. Mapped components

Name	Description	Required/Optional
Package	URI-addressable resource that identifies package as a whole unit	Required. The package implementer shall provide a physical mapping for the package. [M2.2]
Part name	Names a part	Required. The package implementer shall provide a physical mapping for each part's name. [M2.2]
Part content type	Identifies the kind of content stored in the part	Required. The package implementer shall provide a physical mapping for each part's content type. [M2.2]
Part contents	Stores the actual content of the part	Required. The package implementer shall provide a physical mapping for each part's contents. [M2.2]
Part growth hint	Number of additional bytes to reserve for possible growth of part	Optional. The package implementer might provide a physical mapping for a growth hint that might be specified by a producer. [O2.2]

9.1.2 Mapping Content Types

Methods for mapping part content types to a physical format are described below.

9.1.2.1 Identifying the Part Content Type

The package implementer shall define a format mapping with a mechanism for associating content types with parts. [M2.3]

Some physical package formats have a native mechanism for representing content types. [*Example*: the content type header in MIME *end example]* For such packages, the package implementer should use the native mechanism to map the content type for a part. [S2.1]

For all other physical package formats, the package implementer should include a specially-named XML stream in the package called the *Content Types stream.* [S2.2] The Content Types stream shall not be mapped to a part by the package implementer. [M2.1] This stream is therefore not URI-addressable. However, it can be interleaved in the physical package using the same mechanisms used for interleaving parts.

9.1.2.2 Content Types Stream Markup

The Content Types stream identifies the content type for each package part. The Content Types stream contains XML with a top-level Types element, and one or more Default and Override child elements. Default elements define default mappings from the extensions of part names to content types. Override elements are used to specify content types on parts that are not covered by, or are not consistent with, the default mappings. Package producers can use pre-defined Default elements to reduce the number of Override elements on a part, but are not required to do so. [O2.4]

The package implementer shall require that the Content Types stream contain one of the following for every part in the package:

- One matching Default element
- One matching Override element
- Both a matching Default element and a matching Override element, in which case the Override element takes precedence. [M2.4]

The package implementer shall require that there not be more than one Default element for any given extension, and there not be more than one Override element for any given part name. [M2.5]

The order of Default and Override elements in the Content Types stream is not significant.

If the package is intended for streaming consumption:

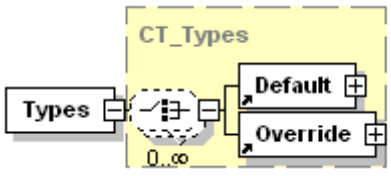
- The package implementer should not allow Default elements; as a consequence, there should be one Override element for each part in the package.
- The format producer should write the Override elements to the package so they appear before the parts to which they correspond, or in close proximity to the part to which they correspond.

[S2.3]

The package implementer can define Default content type mappings even though no parts use them. [O2.5]

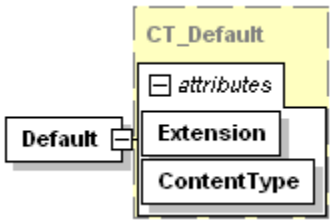
9.1.2.2.1 Types Element

The structure of a Types element is shown in the following diagram:

diagram	
annotation	The root element of the Content Types stream.

9.1.2.2.2 Default Element

The structure of a Default element is shown in the following diagram:

diagram	

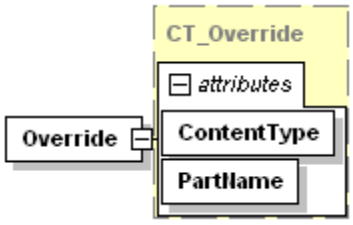
attributes

Name	Type	Use	Default	Fixed	Annotation
Extension	ST_Extension	required			A part name extension. A Default element matches any part whose name ends with a period followed by the value of this attribute. The package implementer shall require a non-empty extension in a Default element. [M2.6]
ContentType	ST_ContentType	required			A content type as defined in RFC 2616. Indicates the content type of any matching parts (unless overridden). The package implementer shall require a content type in a Default element and the format designer shall specify the content type. [M2.6]

annotation	Defines default mappings from the extensions of part names to content types.

9.1.2.2.3 Override Element

The structure of an Override element is shown in the following diagram:

diagram	

attributes	Name	Type	Use	Default	Fixed	Annotation
	ContentType	ST_ContentType	required			A content type as defined in RFC 2616. Indicates the content type of the matching part. The package implementer shall require a content type and the format designer shall specify the content type in an Override element. [M2.7]
	PartName	xs:anyURI	required			A part name. An Override element matches the part whose name is equal to the value of this attribute. The package implementer shall require a part name. [M2.7]
annotation	Specifies content types on parts that are not covered by, or are not consistent with, the default mappings.					

9.1.2.2.4 Content Types Stream Markup Example

[*Example:*

Example 9–7. Content Types stream markup

```
<Types
   xmlns="http://schemas.openxmlformats.org/package/2006/content-types">
   <Default Extension="txt" ContentType="text/plain" />
   <Default Extension="jpeg" ContentType="image/jpeg" />
   <Default Extension="picture" ContentType="image/gif" />
   <Override PartName="/a/b/sample4.picture" ContentType="image/jpeg" />
</Types>
```

The following is a sample list of parts and their corresponding content types as defined by the Content Types stream markup above.

Part name	Content type
/a/b/sample1.txt	text/plain
/a/b/sample2.jpg	image/jpeg
/a/b/sample3.picture	image/gif
/a/b/sample4.picture	image/jpeg

end example]

9.1.2.3 Setting the Content Type of a Part

When adding a new part to a package, the package implementer shall ensure that a content type for that part is specified in the Content Types stream; the package implementer shall perform the following steps to do so [M2.8]:

1. Get the extension from the part name by taking the substring to the right of the rightmost occurrence of the dot character (.) from the rightmost segment.
2. If a part name has no extension, a corresponding Override element shall be added to the Content Types stream.
3. Compare the resulting extension with the values specified for the Extension attributes of the Default elements in the Content Types stream. The comparison shall be case-insensitive ASCII.
4. If there is a Default element with a matching Extension attribute, then the content type of the new part shall be compared with the value of the ContentType attribute. The comparison might be case-sensitive and include every character regardless of the role it plays in the content-type grammar of RFC 2616, or it might follow the grammar of RFC 2616.

    a. If the content types match, no further action is required.
    b. If the content types do not match, a new Override element shall be added to the Content Types stream. .

5. If there is no Default element with a matching Extension attribute, a new Default element or Override element shall be added to the Content Types stream.

9.1.2.4 Getting the Content Type of a Part

To get the content type of a part, the package implementer shall perform the following steps [M2.9]:

1. Compare the part name with the values specified for the PartName attribute of the Override elements. The comparison shall be case-insensitive ASCII.
2. If there is an Override element with a matching PartName attribute, return the value of its ContentType attribute. No further action is required.
3. If there is no Override element with a matching PartName attribute, then

    a. Get the extension from the part name by taking the substring to the right of the rightmost occurrence of the dot character (.) from the rightmost segment.
    b. Check the Default elements of the Content Types stream, comparing the extension with the value of the Extension attribute. The comparison shall be case-insensitive ASCII.

4. If there is a Default element with a matching Extension attribute, return the value of its ContentType attribute. No further action is required.
5. If neither Override nor Default elements with matching attributes are found for the specified part name, the implementation shall not map this part name to a part.

9.1.2.5 Support for Versioning and Extensibility

The package implementer shall not use the versioning and extensibility mechanisms defined in Part 5: “Markup Compatibility and Extensibility” to incorporate elements and attributes drawn from other XML-namespaces into the Content Types stream markup. [M2.10]

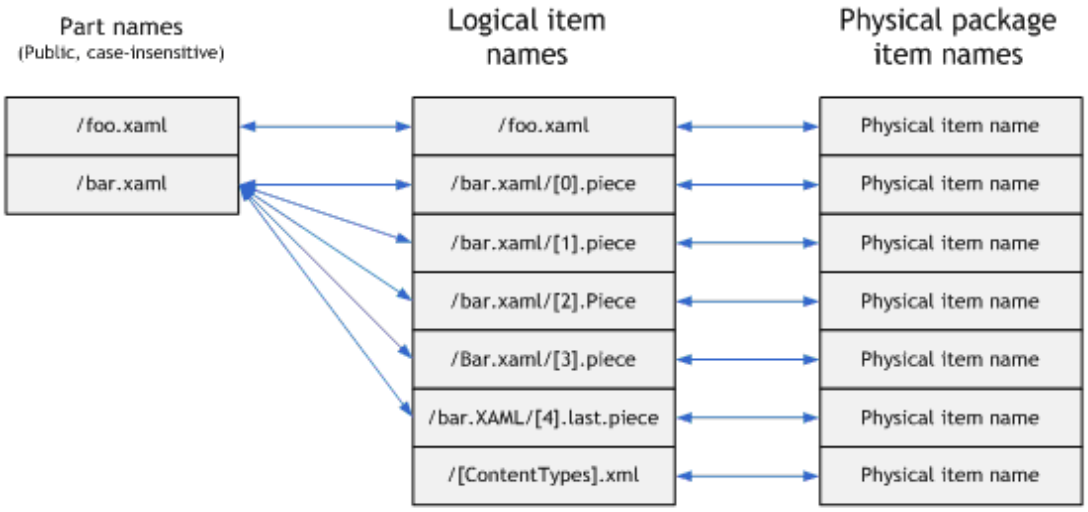
9.1.3 Mapping Part Names to Physical Package Item Names

The mapping of part names to the names of items in the physical package uses an intermediate *logical item name* abstraction. This logical item name abstraction allows package implementers to manipulate physical data items consistently regardless of whether those data items can be mapped to parts or not or whether the package is laid out with simple ordering or interleaved ordering. See §9.1.4 for interleaving details.

[*Example:*

Figure 9–1 illustrates the relationship between part names, logical item names, and physical package item names.

Figure 9–1. Part names and logical item names



end example]

9.1.3.1 Logical Item Names

Logical item names have the following syntax:

```
LogicalItemName = PrefixName [SuffixName]
PrefixName      = *AChar
AChar           = %x20-7E
SuffixName      = "/" "[" PieceNumber "]" [".last"] ".piece"
PieceNumber     = "0" | NonZeroDigit [1*Digit]
Digit           = "0" | NonZeroDigit
```

```
NonZeroDigit    = "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

[*Note:* Piece numbers identify the individual pieces of an interleaved part. *end note]*

The package implementer shall compare prefix names as case-insensitive ASCII strings. [M2.12]

The package implementer shall compare suffix names as case-insensitive ASCII strings. [M2.13]

Logical item names are considered equivalent if their prefix names and suffix names are equivalent. The package implementer shall not allow packages that contain equivalent logical item names. [M2.14] The package implementer shall not allow packages that contain logical items with equivalent prefix names and with equal piece numbers, where piece numbers are treated as integer decimal values. [M2.15]

Logical item names that use suffix names form a complete sequence if and only if:

1. The prefix names of all logical item names in the sequence are equivalent, and
2. The suffix names of the sequence start with “/[0].piece” and end with “/[n].last.piece” and include a piece for every piece number between 0 and n, without gaps, when the piece numbers are interpreted as decimal integer values.

9.1.3.2 Mapping Part Names to Logical Item Names

Non-interleaved part names are mapped to logical item names that have an equivalent prefix name and no suffix name.

Interleaved part names are mapped to the complete sequence of logical item names with an equivalent prefix name.

9.1.3.3 Mapping Logical Item Names and Physical Package Item Names

The mapping of logical item names and physical package item names is specific to the particular physical package.

9.1.3.4 Mapping Logical Item Names to Part Names

A logical item name without a suffix name is mapped to a part name with an equivalent prefix name provided that the prefix name conforms to the part name syntax.

A complete sequence of logical item names is mapped to the part name that is equal to the prefix name of the logical item name having the suffix name “/[0].piece”, provided that the prefix name conforms to the part name syntax.

The package implementer might allow a package that contains logical item names and complete sequences of logical item names that cannot be mapped to a part name because the logical item name does not follow the part naming grammar or the logical item does not have an associated content type. [O2.7] The package implementer shall not map logical items to parts if the logical item names violate the part naming rules. [M2.16]

The package implementer shall consider naming collisions within the set of part names mapped from logical item names to be an error. [M2.17]

9.1.4 Interleaving

Not all physical packages natively support interleaving of the data streams of parts. The package implementer should use the mechanism described in this Open Packaging specification to allow interleaving when mapping to the physical package for layout scenarios that support streaming consumption. [S2.4]

The interleaving mechanism breaks the data stream of a part into *pieces*, which can be interleaved with pieces of other parts or with whole parts. Pieces are named using a unique mapping from the part name, defined in §9.1.3. This enables a consumer to join the pieces together in their original order, forming the data stream of the part.

The individual pieces of an interleaved part exist only in the physical package and are not addressable in the packaging model. A piece might be empty.

An individual part shall be stored either in an interleaved or non-interleaved fashion. The package implementer shall not mix interleaving and non-interleaving for an individual part. [M2.11] The format designer specifies whether that format might use interleaving. [O2.1]

The grammar for deriving piece names from a given part name is defined by the logical item name grammar as defined in §9.1.3.1. A suffix name is mandatory.

The package implementer should store pieces in their natural order for optimal efficiency. [S2.5] The package implementer might create a physical package containing interleaved parts and non-interleaved parts. [O2.6]

[*Example:*

Example 9–8. ZIP archive contents

A ZIP archive might contain the following item names mapped to part pieces and whole parts:

```
spine.xml/[0].piece
pages/page0.xml
spine.xml/[1].piece
pages/page1.xml
spine.xml/[2].last.piece
pages/page2.xml
```

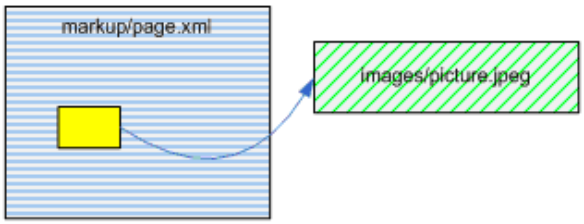
end example]

Under certain scenarios, interleaved ordering can provide important performance benefits, as demonstrated in the following example.

[*Example:*

Example 9–9. Performance benefits with interleaved ordering

The figure below contains two parts: a page part (markup/page.xml) describing the contents of a page, and an image part (images/picture.jpg) referring to an image that appears on the page.



With simple ordering, *all* of the bytes of the page part are delivered before the bytes of the image part. The figure below illustrates this scenario. The consumer is unable to display the image until it has received *all* of the page part *and* the image part. In some circumstances, such as small packages on a high-speed network, this may be acceptable. In others, having to read through all of markup/page.xml to get to the image results in unacceptable performance or places unreasonable memory demands on the consumer's system.

With interleaved ordering, performance is improved by splitting the page part into pieces and inserting the image part immediately following the reference to the image. This allows the consumer to begin processing the image as soon as it encounters the reference.

end example]

9.2 Mapping to a ZIP Archive

This Open Packaging specification defines a mapping for the ZIP archive format. Future versions of this Open Packaging specification might provide additional mappings.

A *ZIP archive* is a ZIP file as defined in the ZIP file format specification excluding all elements of that specification related to encryption, decryption, or digital signatures. A ZIP archive contains *ZIP items*. [*Note*: ZIP items become files when the archive is unzipped. When users unzip a ZIP-based package, they see a set of files and folders that reflects the parts in the package and their hierarchical naming structure. *end note*]

Table 9–2, Package model components and their physical representations, shows the various components of the package model and their corresponding physical representation in a ZIP archive.

Table 9–2. Package model components and their physical representations

Package model component	Physical representation
Package	ZIP archive file
Part	ZIP item
Part name	Stored in item header (and ZIP central directory as appropriate). See §9.2.3 for conversion rules.
Part content type	ZIP item containing XML that identifies the content types for each part according to the pattern described in §9.1.2.1.
Growth hint	Padding reserved in the ZIP Extra field in the local header that precedes the item. See §9.2.7 for a detailed description of the data structure.

9.2.1 Mapping Part Data

In a ZIP archive, the data associated with a part is represented as one or more items.

A package implementer shall store a non-interleaved part as a single ZIP item. [M3.1] When interleaved, a package implementer shall represent a part as one or more pieces, using the method described in §9.1.4. [M2.18] Pieces are named using the specified pattern, making it possible to rebuild the entire part from its constituent pieces. Each piece is stored within a ZIP archive as a single ZIP item.

In the ZIP archive, the chunk of bits that represents an item is stored contiguously. A package implementer might intentionally order the sequence of ZIP items in the archive to enable an efficient organization of the part data in order to achieve correct and optimal interleaving. [O3.1]

9.2.2 ZIP Item Names

ZIP item names are case-sensitive ASCII strings. Package implementers shall create ZIP item names that conform to ZIP archive file name grammar. [M3.2] Package implementers shall create item names that are unique within a given archive. [M3.3]

9.2.3 Mapping Part Names to ZIP Item Names

To map part names to ZIP item names the package implementer shall perform, in order, the following steps [M3.4]:

1. Convert the part name to a logical item name or, in the case of interleaved parts, to a complete sequence of logical item names.
2. Remove the leading forward slash (/) from the logical item name or, in the case of interleaved parts, from each of the logical item names within the complete sequence.

The package implementer shall not map a logical item name or complete sequence of logical item names sharing a common prefix to a part name if the logical item prefix has no corresponding content type. [M3.5]

9.2.4 Mapping ZIP Item Names to Part Names

To map ZIP item names to part names, the package implementer shall perform, in order, the following steps [M3.6]:

1. Map the ZIP item names to logical item names by adding a forward slash (/) to each of the ZIP item names.
2. Map the obtained logical item names to part names. For more information, see §9.1.3.4.

9.2.5 ZIP Package Limitations

The package implementer shall map all ZIP items to parts except MS-DOSZIP items, as defined in the ZIP specification, that are not MS-DOS files. [M3.7]

[*Note:* The ZIP specification specifies that ZIP items recognized as MS-DOS files are those with a "version made by" field and an "external file attributes" field in the "file header" record in the central directory that have a value of 0. *end note]*

In ZIP archives, the package implementer shall not exceed 65,535 bytes for the combined length of the item name, Extra field, and Comment fields. [M3.8] Accordingly, part names stored in ZIP archives are limited to 65,535 characters, subtracting the size of the Extra and Comment fields.

Package implementers should restrict part naming to accommodate file system limitations when naming parts to be stored as ZIP items. [S3.1]

[*Example:*

Examples of these limitations are:

- On Windows file systems, the asterisk ("*") and colon (":") are not valid, so parts named with this character will not unzip successfully.
- On Windows file systems, many programs can handle only file names that are less than 256 characters *including* the full path; parts with longer names might not behave properly once unzipped.
- On Unix file systems, the semicolon (";") has a special meaning, so parts with this character might not be processed as expected.

end example]

ZIP-based packages shall not include encryption as described in the ZIP specification. Package implementers shall enforce this restriction. [M3.9]

9.2.6 Mapping Part Content Type

Part content types are used for associating content types with part data within a package. In ZIP archives, content type information is stored using the common mapping pattern that stores this information in a single XML stream as follows:

- Package implementers shall store content type data in an item(s) mapped to the logical item name with the prefix_name equal to “/[Content_Types].xml” or in the interleaved case to the complete sequence of logical item names with that prefix_name. [M3.10]

Package implementers shall not map logical item name(s) mapped to the Content Types stream in a ZIP archive to a part name. [M3.11] [*Note*: Bracket characters "[" and "]" were chosen for the Content Types stream name specifically because these characters violate the part naming grammar, thus reinforcing this requirement. *end note*]

9.2.7 Mapping the Growth Hint

In a ZIP archive, the growth hint is used to reserve additional bytes that can be used to allow an item to grow in-place. The padding is stored in the Extra field, as defined in the ZIP file format specification. If a growth hint is used for an interleaved part, the package implementer should store the Extra field containing the growth hint padding with the item that represents the first piece of the part. [S3.2]

The format of the ZIP item's Extra field, when used for growth hints, is shown in Table 9–3, Structure of the Extra field for growth hints below.

Table 9–3. Structure of the Extra field for growth hints

Field	Size	Value
Header ID	2 bytes	A220
Length of Extra field	2 bytes	The signature length (2 bytes) + the padding initial value length (2 bytes) + Length of the padding (variable)
Signature (for verification)	2 bytes	A028
Padding Initial Value	2 bytes	Hex number value is set by the producer when the item is created
<padding>	[Padding Length]	Should be filled with NULL characters

9.2.8 Late Detection of ZIP Items Unfit for Streaming Consumption

Several substantial conditions that represent a package unfit for streaming consumption may be detected mid-processing by a streaming package implementer. These include:

- A duplicate ZIP item name is detected the moment the second ZIP item with that name is encountered. Duplicate ZIP item names are not allowed. [M3.3]

- In interleaved packages, an incomplete sequence of ZIP items is detected when the last ZIP item is received. Because one of the interleaved pieces is missing, the entire sequence of ZIP items cannot be mapped to a part and is therefore invalid. [M2.16]
- An inconsistency between the local ZIP item headers and the ZIP central directory file headers is detected at the end of package consumption, when the central directory is processed.
- A ZIP item that is not a file, according to the file attributes in the ZIP central directory, is detected at the end of package consumption, when the central directory is processed. Only a ZIP item that is a file shall be mapped to a part in a valid package.

When any of these conditions are detected, the streaming package implementer shall generate an error, regardless of any processing that has already taken place. Package implementers shall not generate a package containing any of these conditions when generating a package intended for streaming consumption. [M3.13]

9.2.9 ZIP Format Clarifications for Packages

The ZIP format includes a number of features that packages do not support. Some ZIP features are clarified in the package context. See Annex C, “ZIP Appnote.txt Clarifications,” for package-specific ZIP information.

10. Core Properties

Core properties enable users to get and set well-known and common sets of property metadata within packages. The core properties and the Standard that describes them are shown in Table 10–1, “Core properties”. The namespace for the properties in this table in the Open Packaging Conventions domain are defined in Annex F, “Standard Namespaces and Content Types.”

Core property elements are non-repeatable. They may be empty or omitted. The Core Properties Part may be omitted if no core properties are present.

Table 10–1. Core properties

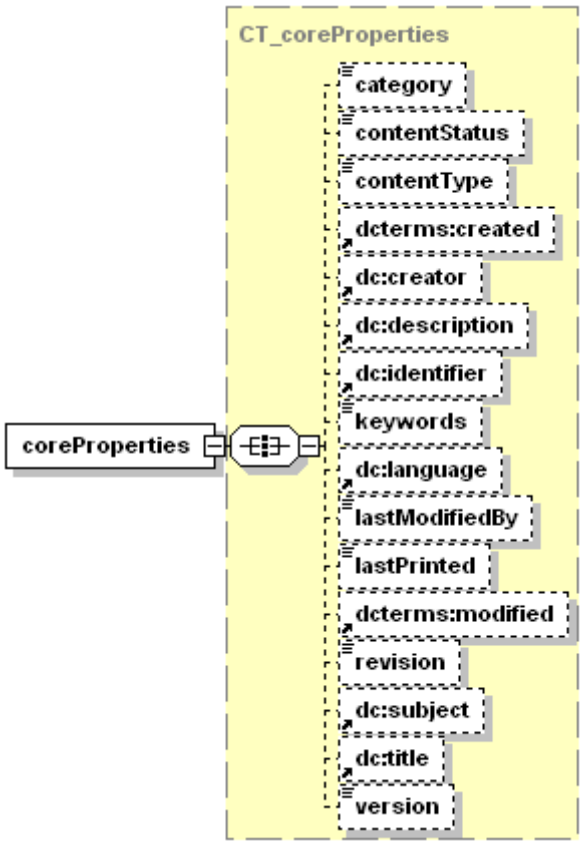
Property	Domain	Description
category	Open Packaging Conventions	A categorization of the content of this package. [*Example*: Example values for this property might include: Resume, Letter, Financial Forecast, Proposal, Technical Presentation, and so on. This value might be used by an application's user interface to facilitate navigation of a large set of documents. *end example*]
contentStatus	Open Packaging Conventions	The status of the content. [*Example*: Values might include “Draft”, “Reviewed”, and “Final”. *end example]*
contentType	Open Packaging Conventions	The type of content represented, generally defined by a specific use and intended audience. [*Example*: Values might include “Whitepaper”, “Security Bulletin”, and “Exam”. *end example]* [*Note*: This property is distinct from MIME content types as defined in RFC 2616. *end note]*
created	Dublin Core	Date of creation of the resource.
creator	Dublin Core	An entity primarily responsible for making the content of the resource.
description	Dublin Core	An explanation of the content of the resource. [*Example*: Values might include an abstract, table of contents, reference to a graphical representation of content, and a free-text account of the content. *end example]*
identifier	Dublin Core	An unambiguous reference to the resource within a given context.

Property	Domain	Description
keywords	Open Packaging Conventions	A delimited set of keywords to support searching and indexing. This is typically a list of terms that are not available elsewhere in the properties.
language	Dublin Core	The language of the intellectual content of the resource. [*Note*: IETF RFC 3066 provides guidance on encoding to represent languages. *end note*]
lastModifiedBy	Open Packaging Conventions	The user who performed the last modification. The identification is environment-specific. [*Example*: A name, email address, or employee ID. *end example]* It is recommended that this value be as concise as possible.
lastPrinted	Open Packaging Conventions	The date and time of the last printing.
modified	Dublin Core	Date on which the resource was changed.
revision	Open Packaging Conventions	The revision number. [*Example:* This value might indicate the number of saves or revisions, provided the application updates it after each revision. *end example]*
subject	Dublin Core	The topic of the content of the resource.
title	Dublin Core	The name given to the resource.
version	Open Packaging Conventions	The version number. This value is set by the user or by the application.

10.1 Core Properties Part

Core properties are stored in XML in the Core Properties part. The Core Properties part content type is defined in Annex F, “Standard Namespaces and Content Types.”

The structure of the CoreProperties element is shown in the following diagram:

diagram	
annotation	Producers might provide all or a subset of these metadata properties to describe the contents of a package.

[*Example:*

Example 10–1. Core properties markup

An example of a core properties part is illustrated by this example:

```
<coreProperties
   xmlns="http://schemas.openxmlformats.org/package/2006/metadata/
      core-properties"
   xmlns:dcterms="http://purl.org/dc/terms/"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <dc:creator>Alan Shen</dc:creator>
   <dcterms:created xsi:type="dcterms:W3CDTF">
      2005-06-12
   </dcterms:created>
   <contentType>Functional Specification</contentType>
   <dc:title>OPC Core Properties</dc:title>
```

```
    <dc:subject>Spec defines the schema for OPC Core Properties and their
       location within the package</dc:subject>
    <dc:language>eng</dc:language>
    <version>1.0</version>
    <lastModifiedBy>Alan Shen</lastModifiedBy>
    <dcterms:modified xsi:type="dcterms:W3CDTF">2005-11-23</dcterms:modified>
    <contentStatus>Reviewed</contentStatus>
    <category>Specification</category>
  </coreProperties>
```

end example]

10.2 Location of Core Properties Part

The location of the Core Properties part within the package is determined by traversing a well-defined package relationship as listed in Annex F, “Standard Namespaces and Content Types”. The format designer shall specify and the format producer shall create at most one core properties relationship for a package. A format consumer shall consider more than one core properties relationship for a package to be an error. If present, the relationship shall target the Core Properties part. [M4.1]

10.3 Support for Versioning and Extensibility

The format designer shall not specify and the format producer shall not create Core Properties that use the Markup Compatibility namespace as defined in Annex F, “Standard Namespaces and Content Types”. A format consumer shall consider the use of the Markup Compatibility namespace to be an error. [M4.2] Instead, versioning and extensibility functionality is accomplished by creating a new part and using a relationship with a new type to point from the Core Properties part to the new part. This Open Packaging specification does not provide any requirements or guidelines for new parts or relationship types that are used to extend core properties.

10.4 Schema Restrictions for Core Properties

The following restrictions apply to every XML document instance that contains Open Packaging Conventions core properties:

1. Producers shall not create a document element that contains refinements to the Dublin Core elements, except for the two specified in the schema: <dcterms:created> and <dcterms:modified> Consumers shall consider a document element that violates this constraint to be an error. [M4.3]
2. Producers shall not create a document element that contains the xml:lang attribute. Consumers shall consider a document element that violates this constraint to be an error. [M4.4] For Dublin Core elements, this restriction is enforced by applications.
3. Producers shall not create a document element that contains the xsi:type attribute, except for a <dcterms:created> or <dcterms:modified> element where the xsi:type attribute shall be present and shall hold the value dcterms:W3CDTF, where `dcterms` is the namespace prefix of the Dublin Core

namespace. Consumers shall consider a document element that violates this constraint to be an error. [M4.5]

11. Thumbnails

The format designer might allow images, called *thumbnails*, to be used to help end-users identify parts of a package or a package as a whole. These images can be generated by the producer and stored as parts. [O5.1]

11.1 Thumbnail Parts

The format designer shall specify thumbnail parts that are identified by either a part relationship or a package relationship. The producer shall build the package accordingly. [M5.1] For information about the relationship type for Thumbnail parts, see Annex F, “Standard Namespaces and Content Types.”

12. Digital Signatures

Format designers might allow a package to include digital signatures to enable consumers to validate the integrity of the contents. The producer might include the digital signature when allowed by the format designer. [O6.1] Consumers can identify the parts of a package that have been signed and the process for validating the signatures. Digital signatures do not protect data from being changed. However, consumers can detect whether signed data has been altered and notify the end-user, restrict the display of altered content, or take other actions.

Producers incorporate digital signatures using a specified configuration of parts and relationships. This clause describes how the package digital signature framework applies the W3C Recommendation "XML-Signature Syntax and Processing" (referred to here as the "XML Digital Signature specification"). In addition to complying with the XML Digital Signature specification, producers and consumers also apply the modifications specified in §12.2.4.1.

12.1 Choosing Content to Sign

Any part or relationship in a package can be signed, including Digital Signature XML Signature parts themselves. An entire Relationships part or a subset of relationships can be signed. By signing a subset, other relationships can be added, removed, or modified without invalidating the signature.

Because applications use the package format to store various types of content, application designers that include digital signatures should define signature policies that are meaningful to their users. A signature policy specifies which portions of a package should not change in order for the content to be considered intact. To ensure validity, some clients require that *all* of the parts and relationships in a package be signed. Others require that *selected* parts or relationships be signed and validated to indicate that the content has not changed. The digital signature infrastructure in packages provides flexibility in defining the content to be signed, while allowing parts of the package to remain changeable.

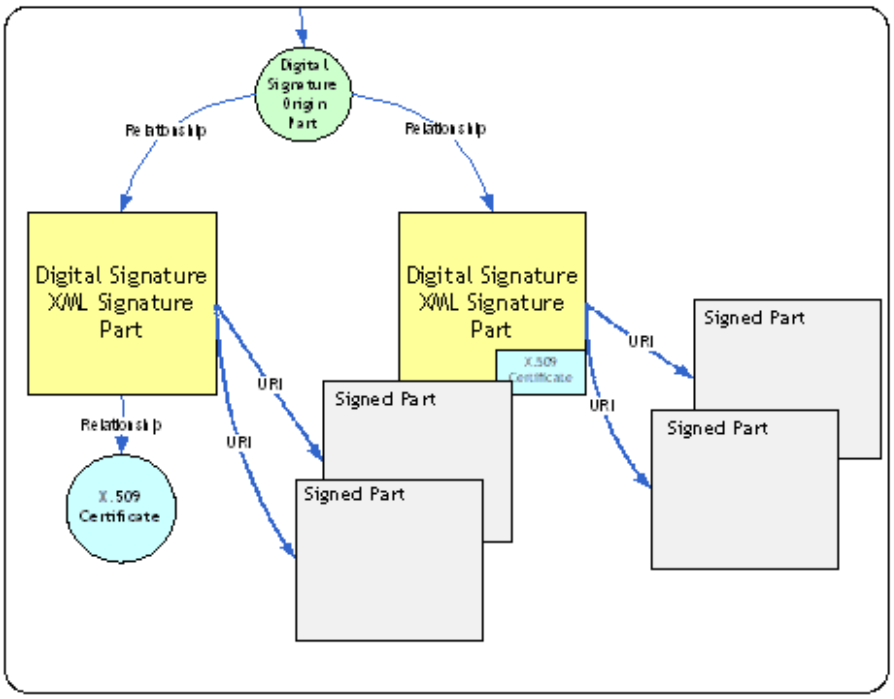
12.2 Digital Signature Parts

The digital signature parts consist of the Digital Signature Origin part, Digital Signature XML Signature parts, and Digital Signature Certificate parts. Relationship names and content types relating to the use of digital signatures in packages are defined in Annex F, "Standard Namespaces and Content Types."

[*Example:*

Figure 12–1 shows a signed package with signature parts, signed parts, and an X.509 certificate. The example Digital Signature Origin part references two Digital Signature XML Signature parts, each containing a signature. The signatures relate to the signed parts.

Figure 12–1. A signed package



end example]

12.2.1 Digital Signature Origin Part

The Digital Signature Origin part is the starting point for navigating through the signatures in a package. The package implementer shall include only one Digital Signature Origin part in a package and it shall be targeted from the package root using the well-defined relationship type specified in Annex F, “Standard Namespaces and Content Types”. [M6.1] When creating the first Digital Signature XML Signature part, the package implementer shall create the Digital Signature Origin part, if it does not exist, in order to specify a relationship to that Digital Signature XML Signature part. [M6.2] If there are no Digital Signature XML Signature parts in the package, the Digital Signature Origin part is optional. [O6.2] Relationships to the Digital Signature XML Signature parts are defined in the Relationships part. The producer should not create any content in the Digital Signature Origin part itself. [S6.1]

The producer shall create Digital Signature XML Signature parts that have a relationship from the Digital Signature Origin part and the consumer shall use that relationship to locate signature information within the package. [M6.3]

12.2.2 Digital Signature XML Signature Part

Digital Signature XML Signature parts are targeted from the Digital Signature Origin part by a relationship that uses the well-defined relationship type specified in Annex F, “Standard Namespaces and Content Types”. The Digital Signature XML Signature part contains digital signature markup. The producer might create zero or more Digital Signature XML Signature parts in a package. [O6.4]

12.2.3 Digital Signature Certificate Part

If present, the Digital Signature Certificate part contains an X.509 certificate for validating the signature. Alternatively, the producer might store the certificate as a separate part in the package, might embed it within the Digital Signature XML Signature part itself, or might not include it in the package if certificate data is known or can be obtained from a local or remote certificate store. [O6.5]

The package digital signature infrastructure supports X.509 certificate technology for signer authentication.

If the certificate is represented as a separate part within the package, the producer shall target that certificate from the appropriate Digital Signature XML Signature part by a Digital Signature Certificate relationship as specified in Annex F, "Standard Namespaces and Content Types" and the consumer shall use that relationship to locate the certificate. [M6.4] The producer might sign the part holding the certificate. [O6.6] The content types of the Digital Signature Certificate part and the relationship targeting it from the Digital Signature XML Signature part are defined in Annex F, "Standard Namespaces and Content Types", Producers might share Digital Signature Certificate parts by using the same certificate to create more than one signature. [O6.7] Producers generating digital signatures should not create Digital Signature Certificate parts that are not the target of at least one Digital Signature Certificate relationship from a Digital Signature XML Signature part. In addition, producers should remove a Digital Signature Certificate part if removing the last Digital Signature XML Signature part that has a Digital Signature Certificate relationship to it. [S6.2]

12.2.4 Digital Signature Markup

The markup described here includes a subset of elements and attributes from the XML Digital Signature specification and some package-specific markup. For a complete example of a digital signature, see §12.3.

12.2.4.1 Modifications to the XML Digital Signature Specification

The package modifications to the XML Digital Signature specification are summarized as follows:

1. The producer shall create Reference elements within a SignedInfo element that reference elements within the same Signature element. The consumer shall consider Reference elements within a SignedInfo element that reference any resources outside the same Signature element to be in error. [M6.5] The producer should only create Reference elements within a SignedInfo element that reference an Object element. [S6.5] The producer shall not create a reference to a package-specific Object element that contains a transform other than a canonicalization transform. The consumer shall consider a reference to a package-specific Object element that contains a transform other than a canonical transform to be an error. [M6.6]
2. The producer shall create one and only one package-specific Object element in the Signature element. The consumer shall consider zero or more than one package-specific Object element in the Signature element to be an error. [M6.7]

The producer shall create package-specific Object elements that contain exactly one Manifest element and exactly one SignatureProperties element. [*Note:* This SignatureProperties element can contain multiple SignatureProperty elements. *end note*] The consumer shall consider package-specific Object elements that

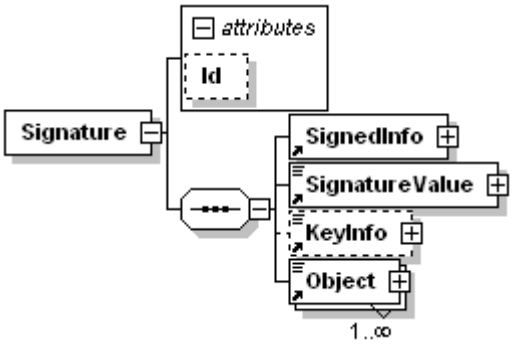
contain other types of elements to be an error. [M6.8] [*Note:* A signature may contain other Object elements that are not package-specific. *end note]*

a. The producer shall create Reference elements within a Manifest element that reference with their URI attribute only parts within the package. The consumer shall consider Reference elements within a Manifest element that reference resources outside the package to be an error. [M6.9] The producer shall create relative references to the local parts that have query components that specifies the part content type as described in §12.2.4.6. The relative reference excluding the query component shall conform to the part name grammar. The consumer shall consider a relative reference to a local part that has a query component that incorrectly specifies the part content type to be an error. [M6.10] The producer shall create Reference elements with a query component that specifies the content type that matches the content type of the referenced part. The consumer shall consider signature validation to fail if the part content type compared in a case-sensitive manner to the content type specified in the query component of the part reference does not match. [M6.11]
b. The producer shall not create Reference elements within a Manifest element that contain transforms other than the canonicalization transform and relationships transform. The consumer shall consider Reference elements within a Manifest element that contain transforms other than the canonicalization transform and relationships transform to be in error. [M6.12]
c. A producer that uses an optional relationships transform shall follow it by a canonicalization transform. The consumer shall consider any relationships transform that is not followed by a canonicalization transform to be an error. [M6.13]
d. The producer shall create exactly one SignatureProperty element with the Id attribute value set to `idSignatureTime`. The Target attribute value of this element shall be either empty or contain a fragment reference to the value of the Id attribute of the root Signature element. A SignatureProperty element shall contain exactly one SignatureTime child element. The consumer shall consider a SignatureProperty element that does not contain a SignatureTime element or whose Target attribute value is not empty or does not contain a fragment reference the Id attribute of the ancestor Signature element to be in error. [M6.14].

[*Note*: All modifications to XML Digital Signature markup occur in locations where the XML Signature schema allows any namespace. Therefore, package digital signature XML is valid against the XML Signature schema. *end note]*

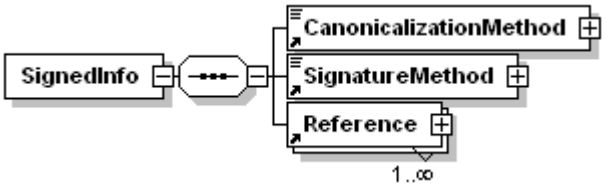
12.2.4.2 Signature Element

The structure of a Signature element is shown in the following diagram:

diagram	
namespace	http://www.w3.org/2000/09/xmldsig#
attributes	Name: Id; Type: xs:ID; Use: optional; Default: ; Fixed: ; Annotation: A unique identifier of the signature xml document.
annotation	The root element of the signature xml document stored in a signature part. The producer shall create a Signature element that contains exactly one local-data, package-specific Object element and zero or more application-specific Object elements. If a Signature element violates this constraint, a consumer shall consider this to be an error. [M6.15]

12.2.4.3 SignedInfo Element

The structure of a SignedInfo element is shown in the following diagram:

diagram	
namespace	http://www.w3.org/2000/09/xmldsig#
annotation	Specifies the data in the package that is signed. Holds one or more references to Object elements within the same Digital Signature XML Signature part. The producer shall create a SignedInfo element that contains exactly one reference to the package-specific Object element. The consumer shall consider it an error if a SignedInfo element does not contain a reference to the package-specific Object element. [M6.16]

12.2.4.4 CanonicalizationMethod Element

The structure of a CanonicalizationMethod element is shown in the following diagram:

diagram	
namespace	http://www.w3.org/2000/09/xmldsig#
attributes	Name: Algorithm; Type: xs:anyURI; Use: required; Default: ; Fixed: ; Annotation: Contains a URI that identifies the particular canonicalization algorithm.
annotation	Specifies the canonicalization algorithm applied to the SignedInfo element prior to performing signature calculations.

Since XML allows equivalent content to be represented differently, a producer should apply a canonicalization transform to the SignedInfo element when it generates it, and a consumer should apply the canonicalization transform to the SignedInfo element when validating it. [S6.3]

[*Note*: Performing a canonicalization transform ensures that SignedInfo content can be validated even if the content has been regenerated using, for example, different entity structures, attribute ordering, or character encoding.

Producers and consumers should also use canonicalization transforms for references to parts that hold XML documents. These transforms are defined using the Transformelement. *end note]*

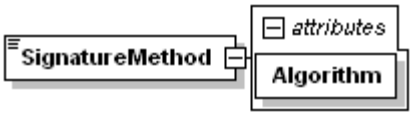
The following canonicalization methods shall be supported by producers and consumers of packages with digital signatures:

- XML Canonicalization (c14n)
- XML Canonicalization with Comments (c14n with comments)

Consumers validating signed packages shall fail the validation if other canonicalization methods are encountered. [M6.34]

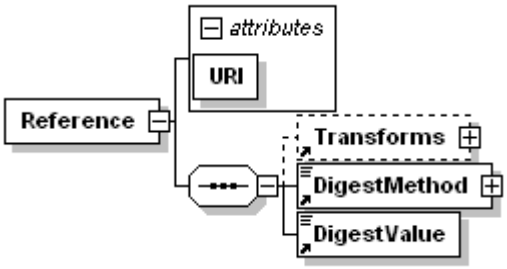
12.2.4.5 SignatureMethod Element

The structure of a SignatureMethod element is shown in the following diagram:

diagram	
namespace	http://www.w3.org/2000/09/xmldsig#
attributes	Name: Algorithm; Type: xs:anyURI; Use: required; Default: ; Fixed: ; Annotation: Contains a URI that identifies the particular algorithm for the signature method.
annotation	Defines the algorithm that is used to convert the SignedInfo element into a hashed value contained in the SignatureValueelement. Producers shall support DSA and RSA algorithms to produce signatures. Consumers shall support DSA and RSA algorithms to validate signatures. [M6.17]

12.2.4.6 Reference Element

The structure of a Reference element is shown in the following diagram:

diagram	
namespace	http://www.w3.org/2000/09/xmldsig#
attributes	Name: URI; Type: xs:anyURI; Use: required; Default: ; Fixed: ; Annotation: Within a <SignedInfo> element, this attribute contains a URI that identifies an element within the signature xml document. Within a <Manifest> element, this attribute contains a relative reference composed of a reference to a part that conforms to the part name grammar and a query component that identifies the content type of

	that part.
annotation	Specifies the object being signed, a digest algorithm, a digest value, and a list of transforms to be applied prior to digesting.

12.2.4.6.1 Usage of <Reference> Element as <Manifest> Child Element

The producer shall create a Reference element within a Manifest element with a URI attribute and that attribute shall contain a part name, without a fragment identifier. The consumer shall consider a Reference element with a URI attribute that does not contain a part name to be an error. [M6.18]

References to package parts include the part content type as a query component. The syntax of the relative reference is as follows:

```
/page1.xml?ContentType="value"
```

where value is the content type of the targeted part.

[*Note:* See §12.2.4.1 for additional requirements on Reference elements. *end note]*

[*Example:*

Example 12–2. Part reference with query component

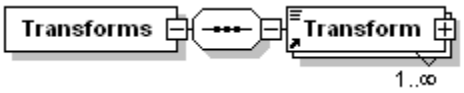
In the following example, the content type is “application/vnd.ms-package.relationships+xml”.

```
URI="/_rels/document.xml.rels?ContentType=application/vnd.ms-
package.relationships+xml"
```

end example]

12.2.4.7 Transforms Element

The structure of a Transforms element is shown in the following diagram:

diagram	
namespace	http://www.w3.org/2000/09/xmldsig#
annotation	Contains an ordered list of Transform elements that describe how the producer digested the Object data before signing it.

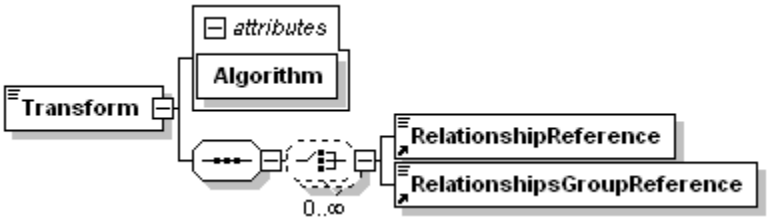
The following transforms shall be supported by producers and consumers of packages with digital signatures:

- XML Canonicalization (c14n)
- XML Canonicalization with Comments (c14n with comments)
- Relationships transform (package-specific)

Consumers validating signed packages shall fail the validation if other transforms are encountered. Relationships transforms shall only be supported by producers and consumers when the Transform element is a descendant element of a Manifest element [M6.19]

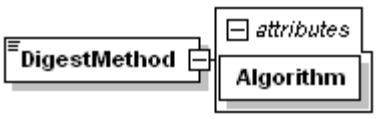
12.2.4.8 Transform Element

The structure of a Transform element is shown in the following diagram:

diagram	
namespace	http://www.w3.org/2000/09/xmldsig#
attributes	Name: Algorithm; Type: xs:anyURI; Use: required; Default: ; Fixed: ; Annotation: Contains a URI that identifies the particular transformation algorithm.
annotation	Describes how the signer obtained the Object data that was digested.

12.2.4.9 DigestMethod Element

The structure of a DigestMethod element is shown in the following diagram:

diagram	
namespace	http://www.w3.org/2000/09/xmldsig#
attributes	Name; Type; Use; Default; Fixed; Annotation

	Algorithm	xs:anyURI	required			Contains a URI that identifies the particular digest method.
annotation	Defines the algorithm that yields the DigestValue from the object data after transforms are applied. Package producers and consumers shall support RSA-SHA1 algorithms to produce or validate signatures. [M6.17]					

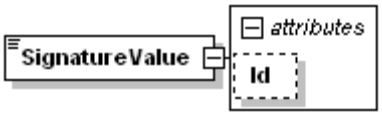
12.2.4.10 DigestValue Element

The structure of a DigestValue element is shown in the following diagram:

diagram	
namespace	http://www.w3.org/2000/09/xmldsig#
annotation	Contains the encoded value of the digest in base64.

12.2.4.11 SignatureValue Element

The structure of a SignatureValue element is shown in the following diagram:

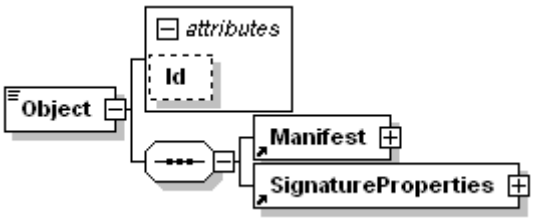
diagram						
namespace	http://www.w3.org/2000/09/xmldsig#					
attributes	Name	Type	Use	Default	Fixed	Annotation
	Id	xs:ID	optional			Contains a URI that identifies the SignatureValueelement within the signature xml document.
annotation	Contains the actual value of the digital signature in base64.					

12.2.4.12 Object Element

The Object element can be either package-specific or application-specific.

12.2.4.13 Package-Specific Object Element

The structure of a Object element is shown in the following diagram:

diagram	
namespace	http://www.w3.org/2000/09/xmldsig#
attributes	Name: Id; Type: xs:ID; Use: ; Default: ; Fixed: ; Annotation: Shall have value of "idPackageObject".
annotation	Holds the Manifest and SignatureProperties elements that are package-specific.

[*Note:* Although the diagram above shows use of the Id attribute as optional, as does the XML Digital Signature schema, for package-specific Object elements, the Id attribute shall be specified and have the value of "idPackageObject". This is a package-specific restriction over and above the XML Digital Signature schema. *end note]*

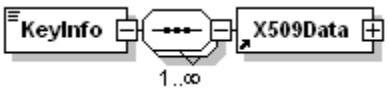
The producer shall create each Signature element with exactly one package-specific Object. For a signed package, consumers shall treat the absence of a package-specific Object, or the presence of multiple package-specific Object elements, as an invalid signature. [M6.15]

12.2.4.14 Application-Specific Object Element

The application-specific Object element specifies application-specific information. The format designer might permit one or more application-specific Object elements. If allowed by the format designer, format producers can create one or more application-specific Object elements. [O6.8] Producers shall create application-specific Object elements that contain XML-compliant data; consumers shall treat data that is not XML-compliant as an error. [M6.20] Format designers and producers might not apply package-specific restrictions regarding URIs and Transform elements to application-specific Object element. [O6.9]

12.2.4.15 KeyInfo Element

The structure of a KeyInfo element is shown in the following diagram:

diagram	
namespace	http://www.w3.org/2000/09/xmldsig#
annotation	Enables recipients to obtain the key needed to validate the signature. Can contain keys, names, certificates, and other public key management information. Producers and consumers shall use the certificate embedded in the Digital Signature XML Signature part when it is specified. [M6.21]

12.2.4.16 X509Data Element

The structure of an X509Data element is shown in the following diagram:

diagram	
namespace	http://www.w3.org/2000/09/xmldsig#
annotation	Contains one or more identifiers of X509 certificates.

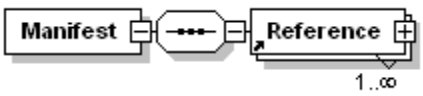
12.2.4.17 X509Certificate Element

The structure of an X509Certificate element is shown in the following diagram:

diagram	
namespace	http://www.w3.org/2000/09/xmldsig#
annotation	Contains a base64-encoded X509 certificate.

12.2.4.18 Manifest Element

The structure of a Manifest element is shown in the following diagram:

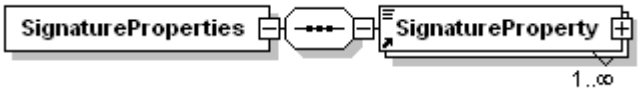
diagram	

namespace	http://www.w3.org/2000/09/xmldsig#
annotation	Contains references to the signed parts of the package. The producer shall not create a Manifest element that references any data outside of the package. The consumer shall consider a Manifest element that references data outside of the package to be in error. [M6.22]

1

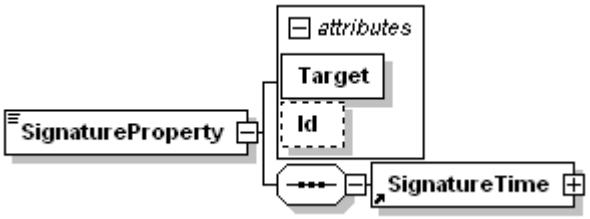
2 12.2.4.19 SignatureProperties Element

3 The structure of a SignaturePropertieselement is shown in the following diagram:

diagram	
namespace	http://www.w3.org/2000/09/xmldsig#
Annotation	Contains additional information items concerning the generation of signatures placed in SignatureProperty elements.

4 12.2.4.20 SignatureProperty Element

5 The structure of a SignatureProperty element is shown in the following diagram:

diagram	
namespace	http://www.w3.org/2000/09/xmldsig#

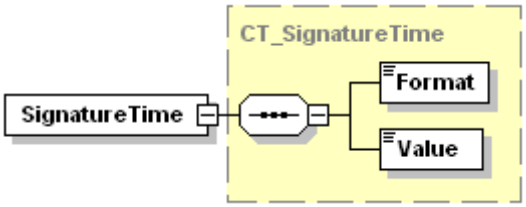
attributes

Name	Type	Use	Default	Fixed	Annotation
Target	xs:anyURI	required			Contains a unique identifier of the Signature element.
Id	xs:ID	optional			Contains signature property's unique identifier.

annotation	Contains additional information concerning the generation of signatures.

12.2.4.21 SignatureTime Element

The structure of a SignatureTime element is shown in the following diagram:

diagram	
namespace	http://schemas.openxmlformats.org/package/2006/digital-signature
annotation	Holds the date/time stamp for the signature.

12.2.4.22 Format Element

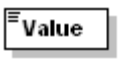
The structure of a Format element is shown in the following diagram:

diagram	
namespace	http://schemas.openxmlformats.org/package/2006/digital-signature
annotation	Specifies the format of the date/time stamp. The producer shall create a data/time format that conforms to the syntax described in the W3C Note "Date and Time Formats". The consumer shall consider a format that does not conform to the syntax described in that WC3 note to be in error. [M6.23]

The date and time format definition conforms to the syntax described in the W3C Note “Date and Time Formats.”

12.2.4.23 Value Element

The structure of a Value element is shown in the following diagram:

diagram	

namespace	http://schemas.openxmlformats.org/package/2006/digital-signature
annotation	Holds the value of the date/time stamp. The producer shall create a value that conforms to the format specified in the Format element. The consumer shall consider a value that does not conform to that format to be in error. [M6.24]

12.2.4.24 RelationshipReference Element

The structure of a RelationshipReference element is shown in the following diagram:

diagram	
namespace	http://schemas.openxmlformats.org/package/2006/digital-signature
attributes	Name: SourceId; Type: xsd:string; Use: required; Default: ; Fixed: ; Annotation: Specifies the value of the Id attribute of the Relationship element.
annotation	Specifies the Relationship element to be signed.

12.2.4.25 RelationshipsGroupReference Element

The structure of a RelationshipsGroupReference element is shown in the following diagram:

diagram	
namespace	http://schemas.openxmlformats.org/package/2006/digital-signature
attributes	Name: SourceType; Type: xsd:anyURI; Use: required; Default: ; Fixed: ; Annotation: Specifies the value of the Type attribute of Relationship elements.
annotation	Specifies that the group of Relationship elements with the specified Type value is to be signed.

Format designers might permit producers to sign individual relationships in a package or the Relationships part as a whole. [O6.10] To sign a subset of relationships, the producer shall use the package-specific relationships transform. The consumer shall use the package-specific relationships transform to validate the signature when a subset of relationships are signed. [M6.25] The transform filters the contents of the Relationships part to include only relationships that have Id values matching the specified SourceId values or Type values matching the specified SourceType values. A producer shall not specify more than one relationship transform for a particular relationships part. A consumer shall treat the presence of more than one relationship transform for a particular relationships part as an error. [M6.35]

Producers shall specify a canonicalization transform immediately following a relationships transform and consumers that encounter a relationships transform that is not immediately followed by a canonicalization transform shall generate an error. [M6.26]

12.2.4.26 Relationships Transform Algorithm

The relationships transform takes the XML document from the Relationships part and converts it to another XML document.

The package implementer might create relationships XML that contains content from several namespaces, along with versioning instructions as defined in Part 5: “Markup Compatibility and Extensibility”. [O6.11]

The relationships transform algorithm is as follows:

Step 1: Process versioning instructions

1. The package implementer shall process the versioning instructions, considering that the only known namespace is the Relationships namespace.
2. The package implementer shall remove all ignorable content, ignoring preservation attributes.
3. The package implementer shall remove all versioning instructions.

Step 2: Sort and filter relationships

1. The package implementer shall remove all namespace declarations except the Relationships namespace declaration.
2. The package implementer shall remove the Relationships namespace prefix, if it is present.
3. The package implementer shall sort relationship elements by Id value in lexicographical order, considering Id values as case-sensitive Unicode strings.
4. The package implementer shall remove all Relationship elements that do not have eitheran Id value that matches any SourceId valueor a Type value that matches any SourceType value, among the SourceId and SourceType values specified in the transform definition. Producers and consumers shall compare values as case-sensitive Unicode strings. [M6.27] The resulting XML document holds all Relationship elements that either have an Id value that matches a SourceId value *or* a Type value that matches a SourceType value specified in the transform definition.

Step 3: Prepare for canonicalization

1. The package implementer shall remove all characters between the Relationships start tag and the first Relationship start tag.
2. The package implementer shall remove any contents of the Relationship element.
3. The package implementer shall remove all characters between the last Relationship end tag and the Relationships end tag.
4. If there are no Relationship elements, the package implementer shall remove all characters between the Relationships start tag and the Relationships end tag.

12.3 Digital Signature Example

The contents of digital signature parts are defined by the W3C Recommendation "XML-Signature Syntax and Processing" with some package-specific modifications specified in §12.2.4.1.

[*Example:*

Digital signature markup for packages is illustrated in this example. For information about namespaces used in this example, see Annex F, "Standard Namespaces and Content Types."

```
<Signature Id="SignatureId" xmlns="http://www.w3.org/2000/09/xmldsig#">
   <SignedInfo>
      <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/
         REC-xml-c14n-20010315"/>
      <SignatureMethod Algorithm="http://www.w3.org/2000/09/
         xmldsig#dsa-sha1"/>
      <Reference
         URI="#idPackageObject"
         Type="http://www.w3.org/2000/09/xmldsig#Object">
         <Transforms>
            <Transform Algorithm="http://www.w3.org/TR/2001/
               REC-xml-c14n-20010315"/>
         </Transforms>
         <DigestMethod Algorithm="http://www.w3.org/2000/09/
            xmldsig#sha1"/>
         <DigestValue>...</DigestValue>
      </Reference>
      <Reference
         URI="#Application"
         Type="http://www.w3.org/2000/09/xmldsig#Object">
         <Transforms>
            <Transform Algorithm="http://www.w3.org/TR/2001/
               REC-xml-c14n-20010315"/>
         </Transforms>
         <DigestMethod
            Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
         <DigestValue>...</DigestValue>
```

```
      </Reference>
    </SignedInfo>
    <SignatureValue>...</SignatureValue>

    <KeyInfo>
      <X509Data>
        <X509Certificate>...</X509Certificate>
      </X509Data>
    </KeyInfo>

    <Object Id="idPackageObject" xmlns:pds="http://schemas.openxmlformats.org
      /package/2006/digital-signature">
      <Manifest>
        <Reference URI="/document.xml?ContentType=application/
          vnd.ms-document+xml">
          <Transforms>
            <Transform Algorithm="http://www.w3.org/TR/2001/
              REC-xml-c14n-20010315"/>
          </Transforms>
          <DigestMethod Algorithm="http://www.w3.org/2000/09/
            xmldsig#sha1"/>
          <DigestValue>...</DigestValue>
        </Reference>
        <Reference
          URI="/_rels/document.xml.rels?ContentType=application/
            vnd.ms-package.relationships+xml">
          <Transforms>
            <Transform Algorithm="http://schemas.openxmlformats.org/
              package/2005/06/RelationshipTransform">
              <pds:RelationshipReference SourceId="B1"/>
              <pds:RelationshipReference SourceId="A1"/>
              <pds:RelationshipReference SourceId="A11"/>
              <pds:RelationshipsGroupReference SourceType=
                "http://schemas.custom.com/required-resource"/>
            </Transform>
            <Transform Algorithm="http://www.w3.org/TR/2001/
              REC-xml-c14n-20010315"/>
          </Transforms>
          <DigestMethod Algorithm="http://www.w3.org/2000/09/
            xmldsig#sha1"/>
          <DigestValue>...</DigestValue>
        </Reference>
      </Manifest>
```

```
        <SignatureProperties>
           <SignatureProperty Id="idSignatureTime" Target="#SignatureId">
              <pds:SignatureTime>
                 <pds:Format>YYYY-MM-DDThh:mmTZD</pds:Format>
                 <pds:Value>2003-07-16T19:20+01:00</pds:Value>
              </pds:SignatureTime>
           </SignatureProperty>
        </SignatureProperties>
     </Object>
     <Object Id="Application">...</Object>
  </Signature>
```

end example]

12.4 Generating Signatures

The steps for signing package contents follow the algorithm outlined in §3.1 of the W3C Recommendation "XML-Signature Syntax and Processing," with some modification for package-specific constructs.

The steps below might not be sufficient for generating signatures that contain application-specific Object elements. Format designers that utilize application-specific Object elements shall also define the additional steps that shall be performed to sign the application-specific Object elements.

To generate references:

1. For each package part being signed:
    a. The package implementer shall apply the transforms, as determined by the producer, to the contents of the part. [*Note:* Relationships transforms are applied only to Relationship parts. When applied, the relationship transform filters the subset of relationships within the entire Relationship part for purposes of signing. *end note*]
    b. The package implementer shall calculate the digest value using the resulting contents of the part.
2. The package implementer shall create a Reference element that includes the reference of the part with the query component matching the content type of the target part, necessary Transform elements, the DigestMethod element and the DigestValue element.
3. The package implementer shall construct the package-specific Object element containing a Manifest element with both the child Reference elements obtained from the preceding step and a child SignatureProperties element, which, in turn, contains a child SignatureTime element.
4. The package implementer shall create a reference to the resulting package-specific Object element.

When signing Object element data, package implementers shall follow the generic reference creation algorithm described in §3.1 of the W3C Recommendation "XML-Signature Syntax and Processing". [M6.28]

To generate signatures:

1. The package implementer shall create the SignedInfo element with a SignatureMethodelement, a CanonicalizationMethod element, and at least one Reference element.
2. The package implementer shall canonicalize the data and then calculate the SignatureValue element using the SignedInfo element based on the algorithms specified in the SignedInfo element.
3. The package implementer shall construct a Signature element that includes SignedInfo, Object, and SignatureValue elements. If a certificate is embedded in the signature, the package implementer shall also include the KeyInfo element.

12.5 Validating Signatures

Consumers validate signatures following the steps described in §3.2 of the W3C Recommendation "XML-Signature Syntax and Processing." When validating digital signatures, consumers shall verify the content type and the digest contained in each Reference descendant element of the SignedInfo element, and validate the signature calculated using the SignedInfo element. [M6.29]

The steps below might not be sufficient to validate signatures that contain application-specific Object elements. Format designers that utilize application-specific Object elements shall also define the additional steps that shall be performed to validate the application-specific Object elements.

To validate references:

1. The package implementer shall canonicalize the SignedInfo element based on the CanonicalizationMethod element specified in the SignedInfo element.
2. For each Reference element in the SignedInfo element:

    a. The package implementer shall obtain the Object element to be digested.
    b. For the package-specific Object element, the package implementer shall validate references to signed parts stored in the Manifest element. The package implementer shall consider references invalid if there is a missing part. [M6.9] [*Note:* If a relationships transform is specified for a signed Relationships part, only the specified subset of relationships within the entire Relationships part are validated. *end note*]
    c. For the package-specific Object element, validation of Reference elements includes verifying the content type of the referenced part and the content type specified in the reference query component. Package implementers shall consider references invalid if these two values are different. The string comparison shall be case-sensitive and locale-invariant. [M6.11]
    d. The package implementer shall digest the obtained Object element using the DigestMethod element specified in the Reference element.
    e. The package implementer shall compare the generated digest value against the DigestValue element in the Reference element of the SignedInfo element. Package implementers shall consider references invalid if there is any mismatch. [M6.30]

To validate signatures:

1. The package implementer shall obtain the public key information from the KeyInfo element or from an external source.

2. The package implementer shall obtain the canonical form of the SignatureMethod element using the CanonicalizationMethod element. The package implementer shall use the result and the previously obtained KeyInfo element to confirm the SignatureValue element stored in the SignedInfo element. The package implementer shall decrypt the SignatureValueelement using the public key prior to comparison.

12.5.1 Signature Validation and Streaming Consumption

Streaming consumers that maintain signatures shall be able to cache the parts necessary for detecting and processing signatures. [M6.31]

12.6 Support for Versioning and Extensibility

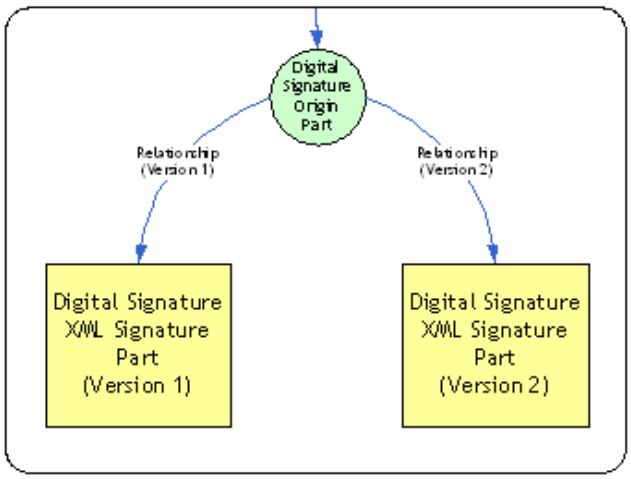
The package digital signature infrastructure supports the exchange of signed packages between current and future package clients.

12.6.1 Using Relationship Types

Future versions of the package format will specify distinct relationship types for revised signature parts. Using these relationships, producers will be able to store separate signature information for current and previous versions. Consumers will be able to choose the signature information they know how to validate.

Figure 12–2, “Part names and logical item names”, illustrates this versioning capability that will be available in future versions of the package format.

Figure 12–2. A package containing versioned signatures



12.6.2 Markup Compatibility Namespace for Package Digital Signatures

The package implementer shall not use the Markup Compatibility namespace, as specified in Annex F, “Standard Namespaces and Content Types,” within the package-specific Object element. The package implementer shall consider the use of the Markup Compatibility namespace within the package-specific Object element to be an error. [M6.32]

Format designers might specify an application-specific package part format that allows for the embedding of versioned or extended content that might not be fully understood by all present and future implementations. Producers might create such embedded versioned or extended content and consumers might encounter such content. [O6.12] [*Example*: An XML package part format might rely on Markup Compatibility elements and attributes to embed such versioned or extended content. *end example]*

If an application allows for a single part to contain information that might not be fully understood by all implementations, then the format designer shall carefully design the signing and verification policies to account for the possibility of different implementations being used for each action in the sequence of content creation, content signing, and signature verification. Producers and consumers shall account for this possibility in their signing and verification processing. [M6.33]

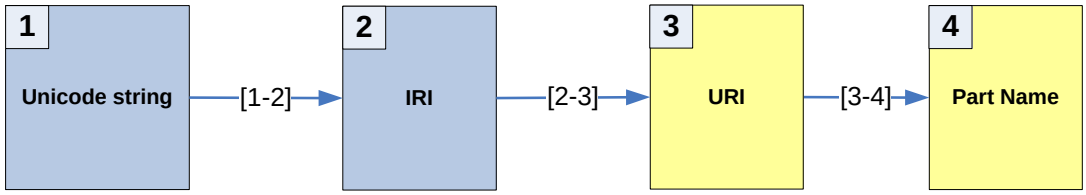
Annex A. Resolving Unicode Strings to Part Names

Package clients might use Unicode strings for referencing parts in a package. [*Example*: Values of xsd:anyURI data type within XML markup are Unicode strings. *end example]*

This annex specifies how such Unicode strings shall be resolved to part names.

The diagram below illustrates the conversion path from the Unicode string to a part name. The numbered arcs identify string transformations.

Figure A–1. Strings are converted to part names for referencing parts



A Unicode string representing a URI can be passed to the producer or consumer. The producing or consuming application shall convert the Unicode string to a URI. If the URI is a relative reference, the application shall resolve it using the base URI of the part, which is expressed using the pack scheme, to the URI of the referenced part. [M1.33]

The process for resolving a Unicode string to a part name follows Arcs [1-2], [2-3], and [3-4].

A.1 Creating an IRI from a Unicode String

With reference to Arc [1-2] in Figure A–1, a Unicode string is converted to an IRI by percent-encoding each ASCII character that does not belong to the set of reserved or unreserved characters as defined in RFC 3986.

A.2 Creating a URI from an IRI

With reference to Arc [2-3] in Figure A–1, an IRI is converted to a URI by converting non-ASCII characters as defined in Step 2 in §3.1 of RFC 3987

If a consumer converts the URI back into an IRI, the conversion shall be performed as specified in §3.2 of RFC 3987. [M1.34]

A.3 Resolving a Relative Reference to a Part Name

If the URI reference obtained in §A.2 is a URI, it is resolved in the regular way, that is, with no package-specific considerations. Otherwise, if the URI reference is a relative reference, it is resolved (with reference to Arc [3-4] in Figure A–1) as follows:

1. Percent-encode each open bracket ([) and close bracket (]).
2. Percent-encode each percent (%) character that is not followed by a hexadecimal notation of an octet value.
3. Un-percent-encode each percent-encoded unreserved character.
4. Un-percent-encode each forward slash (/) and back slash (\).
5. Convert all back slashes to forward slashes.
6. If present in a segment containing non-dot (“.”) characters, remove trailing dot (“.”) characters from each segment.
7. Replace each occurrence of multiple consecutive forward slashes (/) with a single forward slash.
8. If a single trailing forward slash (/) is present, remove that trailing forward slash.
9. Remove complete segments that consist of three or more dots.
10. Resolve the relative reference against the base URI of the part holding the Unicode string, as it is defined in §5.2 of RFC 3986. The path component of the resulting absolute URI is the part name.

A.4 String Conversion Examples

[*Example:*

Examples of Unicode strings converted to IRIs, URIs, and part names are shown below:

Unicode string	IRI	URI	Part name
/a/b.xml	/a/b.xml	/a/b.xml	/a/b.xml
/a/ц.xml	/a/ц.xml	/a/%D1%86.xml	/a/%D1%86.xml
/%41/%61.xml	/%41/%61.xml	/%41/%61.xml	/A/a.xml
/%25XY.xml	/%25XY.xml	/%25XY.xml	/%25XY.xml
/%XY.xml	/%XY.xml	/%25XY.xml	/%25XY.xml
/%2541.xml	/%2541.xml	/%2541.xml	/%2541.xml
/../a.xml	/../a.xml	/../a.xml	/a.xml
/./ц.xml	/./ц.xml	/./%D1%86.xml	/%D1%86.xml
/%2e/%2e/a.xml	/%2e/%2e/a.xml	/%2e/%2e/a.xml	/a.xml
\a.xml	%5Ca.xml	%5Ca.xml	/a.xml
\%41.xml	%5C%41.xml	%5C%41.xml	/A.xml
/%D1%86.xml	/%D1%86.xml	/%D1%86.xml	/%D1%86.xml
\%2e/a.xml	%5C%2e/a.xml	%5C%2e/a.xml	/a.xml

end example]

Annex B. Pack URI

A package is a logical entity that holds a collection of parts. This Open Packaging specification defines a way to use URIs to reference part resources inside a package. This approach defines a new scheme in accordance with the guidelines in RFC 3986.

The following terms are used as they are defined in RFC 3986: *scheme*, *authority*, *path*, *segment*, *reserved characters*, *sub-delims*, *unreserved characters*, *pchar*, *pct-encoded characters*, *query*, *fragment*, and *resource*.

B.1 Pack URI Scheme

RFC 3986 provides an extensible mechanism for defining new kinds of URIs based on new schemes. Schemes are the prefix in a URI before the colon. [*Example*: “http”, “ftp”, “file” *end example*] This Open Packaging specification defines a specific URI scheme used to refer to parts in a package: the pack scheme. A URI that uses the pack scheme is called a *pack URI*.

The pack URI grammar is defined as follows:

```
pack_URI  = "pack://" authority [ "/" | path ]
authority = *( unreserved | sub-delims | pct-encoded )
path      = 1*( "/" segment )
segment   = 1*( pchar )
```

unreserved, sub-delims, pchar and pct-encoded are defined in RFC 3986

The authority component contains an embedded URI that points to a package. The package implementer shall create an embedded URI that meets the requirements defined in RFC 3986 for a valid URI. [M7.1] §B.3 describes the rules for composing pack URIs by combining the URI of an entire package resource with a part name.

The package implementer shall not create an authority component with an unescaped colon (:) character. [M7.4] Consumer applications, based on the obsolete URI specification RFC 2396, might tolerate the presence of an unescaped colon character in an authority component. [O7.1]

The optional path component identifies a particular part within the package. The package implementer shall only create path components that conform to the part naming rules. When the path component is missing, the resource identified by the pack URI is the package as a whole. [M7.2]

In order to be able to embed the URI of the package in the pack URI, it is necessary either to replace or to percent-encode occurrences of certain characters in the embedded URI. For example, forward slashes (/) are replaced with commas (,). The rules for these substitutions are described in §B.3.

The optional query component in a pack URI is ignored when resolving the URI to a part.

A pack URI might have a fragment identifier as specified in RFC 3986. If present, this fragment applies to whatever resource the pack URI identifies.

[*Example:*

Example B–1. Using the pack URI to identify a part

The following URI identifies the "/a/b/foo.xml" part within the "http://www.openxmlformats.org/my.container" package resource:

```
pack://http%3c,,www.openxmlformats.org,my.container/a/b/foo.xml
```

end example]

[*Example:*

Example B–2. Equivalent pack URIs

The following pack URIs are equivalent:

```
pack://http%3c,,www.openxmlformats.org,my.container
pack://http%3c,,www.openxmlformats.org,my.container/
```

end example]

[*Example:*

Example B–3. A pack URI with percent-encoded characters

The following URI identifies the "/c/d/bar.xml" part within the "http://myalias:pswr@www.my.com/containers.aspx?my.container" package:

```
pack://http%3c,,myalias%3cpswr%40www.my.com,containers.aspx%3fmy.container
/c/d/bar.xml
```

end example]

B.2 Resolving a Pack URI to a Resource

The following is an algorithm for resolving a pack URI to a resource (either a package or a part):

1. Parse the pack URI into the potential three components: scheme, authority, path, as well as any fragment identifier.
2. In the authority component, replace all commas (,) with forward slashes (/).
3. Un-percent-encode ASCII characters in the resulting authority component.
4. The resultant authority component is the URI for the package as a whole.
5. If the path component is empty, the pack URI resolves to the package as a whole and the resolution process is complete.

6. A non-empty path component shall be a valid part name. If it is not, the pack URI is invalid.
7. The pack URI resolves to the part with this part name in the package identified by the authority component.

[*Example:*

Example B–4. Resolving a pack URI to a resource

Given the pack URI:

```
pack://http%3c,,www.my.com,packages.aspx%3fmy.package/a/b/foo.xml
```

The components:

```
<authority>= http%3c,,www.my.com,packages.aspx%3fmy.package
<path>= /a/b/foo.xml
```

Are converted to the package URI:

```
http://www.my.com/packages.aspx?my.package
```

And the path:

```
/a/b/foo.xml
```

Therefore, this URI refers to a part named "/a/b/foo.xml" in the package at the following URI: http://www.my.com/packages.aspx?my.package.

end example]

B.3 Composing a Pack URI

The following is an algorithm for composing a pack URI from the URI of an entire package resource and a part name.

In order to be suitable for creating a pack URI, the URI reference of a package resource shall conform to RFC 3986 requirements for valid absolute URIs.

To compose a pack URI from the absolute package URI and a part name, the following steps shall be performed, in order:

1. Remove the fragment identifier from the package URI, if present.
2. Percent-encode all percent signs (%), question marks (?), at signs (@), colons (:) and commas (,) in the package URI.
3. Replace all forward slashes (/) with commas (,) in the resulting string.
4. Append the resulting string to the string "pack://".
5. Append a forward slash (/) to the resulting string. The constructed string represents a pack URI with a blank path component.

6. Using this constructed string as a base URI and the part name as a relative reference, apply the rules defined in RFC 3986 for resolving relative references against the base URI.

The result of this operation will be the pack URI that refers to the resource specified by the part name.

[*Example:*

Example B–5. Composing a pack URI

Given the package URI:

```
http://www.my.com/packages.aspx?my.package
```

And the part name:

```
/a/foo.xml
```

The pack URI is:

```
pack://http%3c,,www.my.com,packages.aspx%3fmy.package/a/foo.xml
```

end example]

B.4 Equivalence

In some scenarios, such as caching or writing parts to a package, it is necessary to determine if two pack URIs are equivalent without resolving them.

The package implementer shall consider pack URIs equivalent if:

1. The scheme components are octet-by-octet identical after they are both converted to lowercase; *and*
2. The URIs, decoded as described in §B.2 from the authority components are equivalent (the equivalency rules by scheme, as per RFC 3986); *and*
3. The path components are equivalent when compared as case-insensitive ASCII strings.

[M7.3]

Annex C. ZIP Appnote.txt Clarifications

The ZIP specification includes a number of features that packages do not support. Some ZIP features are clarified in the context of this Open Packaging specification. Package producers and consumers shall adhere to the requirements noted below.

C.1 Archive File Header Consistency

Data describing files stored in the archive is substantially duplicated in the Local File Headers and Data Descriptors, and in the File headers within the Central Directory Record. For a ZIP archive to be a valid physical layer for a package, the package implementer shall ensure that the ZIP archive holds equal values in the appropriate fields of every File Header within the Central Directory and the corresponding Local File Header and Data Descriptor pair. [M3.14]

C.2 Table Key

- "Yes" — During consumption of a package, a "Yes" value for a field in a table in Annex C indicates a package implementer shall support reading the ZIP archive containing this record or field, however, support may mean ignoring. [M3.15] During production of a package, a "Yes" value for a field in a table in Annex C indicates that the package implementer shall write out this record or field. [M3.16]
- "No" — A "No" value for a field in a table in Annex C indicates the package implementer shall not use this record or field during consumption or production of packages. [M3.17]
- "Optional" — An "Optional" value for a record in a table in Annex C indicates that package implementers might write this record during production. [O3.2]
- "Partially, details below" — A "Partially, details below" value for a record in a table in Annex C indicates that the record contains fields that might not be supported by package implementers during production or consumption. See the details in the corresponding table to determine requirements. [M3.18]
- "Only used when needed" — The value "Only used when needed" associated with a record in a table in Annex C indicates that the package implementer shall use the record only when needed to store data in the ZIP archive. [M3.19]

Table C–1,"Support for records", specifies the requirements for package production, consumption, and editing in regard to particular top-level records or fields described in the ZIP Appnote.txt. [*Note:* Editing, in this context, means in-place modification of individual records. A format specification can require editing applications to instead modify content in-memory and re-write all parts and relationships on each save in order to maintain more rigorous control of ZIP record usage. *end note*]

Table C–1. Support for records

Record name	Supported on Consumption	Supported on Production	Pass through on editing

Record name	Supported on Consumption	Supported on Production	Pass through on editing
Local File Header	Yes (partially, details below)	Yes (partially, details below)	Yes
File data	Yes	Yes	Yes
Data descriptor	Yes	Optional	Optional
Archive decryption header	No	No	No
Archive extra data record	No	No	No
Central directory structure: File header	Yes (partially, details below)	Yes (partially, details below)	Yes
Central directory structure: Digital signature	Yes (ignore the signature data)	Optional	Optional
Zip64 end of central directory record V1 (from spec version 4.5)	Yes (partially, details below)	Yes (partially, details below, used only when needed)	Optional
Zip64 end of central directory record V2 (from spec version 6.2)	No	No	No
Zip64 end of central directory locator	Yes (partially, details below)	Yes (partially, details below, used only when needed)	Optional
End of central directory record	Yes (partially, details below)	Yes (partially, details below, used only when needed)	Yes

Table C–2, “Support for record components”, specifies the requirements for package production, consumption, and editing in regard to individual record components described in the ZIP Appnote.txt.

Table C–2. Support for record components

Record	Field	Supported on Consumption	Supported on Production	Pass through on editing
Local File Header	Local file header signature	Yes	Yes	Yes
	Version needed to extract	Yes (partially, see Table C–3)	Yes (partially, see Table C–3)	Yes (partially, see Table C–3)

Record	Field	Supported on Consumption	Supported on Production	Pass through on editing
	General purpose bit flag	Yes (partially, see Table C–5)	Yes (partially, see Table C–5)	Yes (partially, see Table C–5)
	Compression method	Yes (partially, see Table C–4)	Yes (partially, see Table C–4)	Yes (partially, see Table C–4)
	Last mod file time	Yes	Yes	Yes
	Last mod file date	Yes	Yes	Yes
	Crc-32	Yes	Yes	Yes
	Compressed size	Yes	Yes	Yes
	Uncompressed size	Yes	Yes	Yes
	File name length	Yes	Yes	Yes
	Extra field length	Yes	Yes	Yes
	File name (variable size)	Yes	Yes	Yes
	Extra field (variable size)	Yes (partially, see Table C–6)	Yes (partially, see Table C–6)	Yes (partially, see Table C–6)
Central directory structure: File header	Central file header signature	Yes	Yes	Yes
	version made by: high byte	Yes	Yes (0 = MS-DOS is default publishing value)	Yes
	Version made by: low byte	Yes	Yes	Yes
	Version needed to extract (see Table C–3 for details)	Yes (partially, see Table C–3)	Yes (1.0, 1.1, 2.0, 4.5)	Yes
	General purpose bit flag	Yes (partially, see Table C–5)	Yes (partially, see Table C–5)	Yes (partially, see Table C–5)
	Compression method	Yes (partially, see Table C–4)	Yes (partially, see Table C–4)	Yes (partially, see Table C–4)
	Last mod file time (Pass through, no interpretation)	Yes	Yes	Yes
	Last mod file date (Pass through, in interpretation)	Yes	Yes	Yes
	Crc-32	Yes	Yes	Yes
	Compressed size	Yes	Yes	Yes
	Uncompressed size	Yes	Yes	Yes
	File name length	Yes	Yes	Yes
	Extra field length	Yes	Yes	Yes

Record	Field	Supported on Consumption	Supported on Production	Pass through on editing
	File comment length	Yes	Yes (always set to 0)	Yes
	Disk number start	Yes (partial — no multi disk archives)	Yes (always 1 disk)	Yes (partial — no multi disk archives)
	Internal file attributes	Yes	Yes	Yes
	External file attributes (Pass through, no interpretation)	Yes	Yes (MS DOS default value)	Yes
	Relative offset of local header	Yes	Yes	Yes
	File name (variable size)	Yes	Yes	Yes
	Extra field (variable size)	Yes (partially, see Table C–6)	Yes (partially, see Table C–6)	Yes (partially, see Table C–6)
	File comment (variable size)	Yes	Yes (always set to empty)	Yes
Zip64 end of central directory V1 (from spec version 4.5, only used when needed)	Zip64 end of central directory signature	Yes	Yes	Yes
	Size of zip64 end of central directory	Yes	Yes	Yes
	Version made by: high byte (Pass through, no interpretation)	Yes	Yes (0 = MS-DOS is default publishing value)	Yes
	Version made by: low byte	Yes	Yes (always 4.5 or above)	Yes
	Version needed to extract (see Table C–3 for details)	Yes (4.5)	Yes (4.5)	Yes (4.5)
	Number of this disk	Yes (partial — no multi disk archives)	Yes (always 1 disk)	Yes (partial — no multi disk archives)
	Number of the disk with the start of the central directory	Yes (partial — no multi disk archives)	Yes (always 1 disk)	Yes (partial — no multi disk archives)
	Total number of entries in the central directory on this disk	Yes	Yes	Yes
	Total number of entries in the central directory	Yes	Yes	Yes

Record	Field	Supported on Consumption	Supported on Production	Pass through on editing
	Size of the central directory	Yes	Yes	Yes
	Offset of start of central directory with respect to the starting disk number	Yes	Yes	Yes
	Zip64 extensible data sector	Yes	No	Yes
Zip64 end of central directory locator (only used when needed)	Zip64 end of central dir locator signature	Yes	Yes	Yes
	Number of the disk with the start of the zip64 end of central directory	Yes (partial — no multi disk archives)	Yes (always 1 disk)	Yes (partial — no multi disk archives)
	Relative offset of the zip64 end of central directory record	Yes	Yes	Yes
	Total number of disks	Yes (partial — no multi disk archives)	Yes (always 1 disk)	Yes (partial — no multi disk archives)
End of central directory record	End of central dir signature	Yes	Yes	Yes
	Number of this disk	Yes (partial — no multi disk archives)	Yes (always 1 disk)	Yes (partial — no multi disk archives)
	Number of the disk with the start of the central directory	Yes (partial — no multi disk archive)	Yes (always 1 disk)	Yes (partial — no multi disk archive)
	Total number of entries in the central directory on this disk	Yes	Yes	Yes
	Total number of entries in the central directory	Yes	Yes	Yes
	Size of the central directory	Yes	Yes	Yes
	Offset of start of central directory with respect to the starting disk number	Yes	Yes	Yes
	ZIP file comment length	Yes	Yes	Yes
	ZIP file comment	Yes	No	Yes

1

Table C–3, “Support for Version Needed to Extract field”, specifies the detailed production, consumption, and editing requirements for the Extract field, which is fully described in the ZIP Appnote.txt.

Table C–3. Support for Version Needed to Extract field

Version	Feature	Supported on Consumption	Supported on Production	Pass through on editing
1.0	Default value	Yes	Yes	Yes
1.1	File is a volume label	Ignore	No	(rewrite/remove)
2.0	File is a folder (directory)	Ignore	No	(rewrite/remove)
2.0	File is compressed using Deflate compression	Yes	Yes	Yes
2.0	File is encrypted using traditional PKWARE encryption	No	No	No
2.1	File is compressed using Deflate64(tm)	No	No	No
2.5	File is compressed using PKWARE DCL Implode	No	No	No
2.7	File is a patch data set	No	No	No
4.5	File uses ZIP64 format extensions	Yes	Yes	Yes
4.6	File is compressed using BZIP2 compression	No	No	No
5.0	File is encrypted using DES	No	No	No
5.0	File is encrypted using 3DES	No	No	No
5.0	File is encrypted using original RC2 encryption	No	No	No
5.0	File is encrypted using RC4 encryption	No	No	No
5.1	File is encrypted using AES encryption	No	No	No
5.1	File is encrypted using corrected RC2 encryption	No	No	No
5.2	File is encrypted using corrected RC2-64 encryption	No	No	No
6.1	File is encrypted using non-OAEP key wrapping	No	No	No
6.2	Central directory encryption	No	No	No

Table C–4, “Support for Compression Method field”, specifies the detailed production, consumption, and editing requirements for the Compression Method field, which is fully described in the ZIP Appnote.txt.

Table C–4. Support for Compression Method field

Code	Method	Supported on Consumption	Supported on Production	Pass through on editing
0	The file is stored (no compression)	Yes	Yes	Yes
1	The file is Shrunk	No	No	No
2	The file is Reduced with compression factor 1	No	No	No
3	The file is Reduced with compression factor 2	No	No	No
4	The file is Reduced with compression factor 3	No	No	No
5	The file is Reduced with compression factor 4	No	No	No
6	The file is Imploded	No	No	No
7	Reserved for Tokenizing compression algorithm	No	No	No
8	The file is Deflated	Yes	Yes	Yes
9	Enhanced Deflating using Deflate64™	No	No	No
10	PKWARE Data Compression Library Imploding	No	No	No
11	Reserved by PKWARE	No	No	No

Table C–5, “Support for modes/structures defined by general purpose bit flags”, specifies the detailed production, consumption, and editing requirements when utilizing these general-purpose bit flags within records.

Table C–5. Support for modes/structures defined by general purpose bit flags

Bit	Feature	Supported on Consumption	Supported on Production	Pass through on editing
0	If set, indicates that the file is encrypted.	No	No	No

<table>
<tr><th>Bit</th><th>Feature</th><th>Supported on Consumption</th><th>Supported on Production</th><th>Pass through on editing</th></tr>
<tr><td>1, 2</td><td><table>
<tr><th>Bit 2</th><th>Bit 1</th><th></th></tr>
<tr><td>0</td><td>0</td><td>Normal (-en) compression option was used.</td></tr>
<tr><td>0</td><td>1</td><td>Maximum (-exx/-ex) compression option was used.</td></tr>
<tr><td>1</td><td>0</td><td>Fast (-ef) compression option was used.</td></tr>
<tr><td>1</td><td>1</td><td>Super Fast (-es) compression option was used.</td></tr>
</table></td><td>Yes</td><td>Yes</td><td>Yes</td></tr>
<tr><td>3</td><td>If this bit is set, the fields crc-32, compressed size and uncompressed size are set to zero in the local header. The correct values are put in the data descriptor immediately following the compressed data. (PKZIP version 2.04g for DOS only recognizes this bit for method 8 compression, newer versions of PKZIP recognize this bit for any compression method.)</td><td>Yes</td><td>Yes</td><td>Yes</td></tr>
<tr><td>4</td><td>Reserved for use with method 8, for enhanced deflating</td><td>Ignore</td><td>Bits set to 0</td><td>Yes</td></tr>
<tr><td>5</td><td>If this bit is set, this indicates that the file is compressed patched data. (Requires PKZIP version 2.70 or greater.)</td><td>Ignore</td><td>Bits set to 0</td><td>Yes</td></tr>
<tr><td>6</td><td>Strong encryption. If this bit is set, you should set the version needed to extract value to at least 50 and you must also set bit 0. If AES encryption is used, the version needed to extract value must be at least 51.</td><td>Ignore</td><td>Bits set to 0</td><td>Yes</td></tr>
<tr><td>7</td><td>Currently unused</td><td>Ignore</td><td>Bits set to 0</td><td>Yes</td></tr>
<tr><td>8</td><td>Currently unused</td><td>Ignore</td><td>Bits set to 0</td><td>Yes</td></tr>
<tr><td>9</td><td>Currently unused</td><td>Ignore</td><td>Bits set to 0</td><td>Yes</td></tr>
<tr><td>10</td><td>Currently unused</td><td>Ignore</td><td>Bits set to 0</td><td>Yes</td></tr>
<tr><td>11</td><td>Currently unused</td><td>Ignore</td><td>Bits set to 0</td><td>Yes</td></tr>
</table>

Bit	Feature	Supported on Consumption	Supported on Production	Pass through on editing
12	Reserved by PKWARE for enhanced compression	Ignore	Bits set to 0	Yes
13	Used when encrypting the Central Directory to indicate selected data values in the Local Header are masked to hide their actual values. See the section describing the Strong Encryption Specification for details.	Ignore	Bits set to 0	Yes
14	Reserved by PKWARE	Ignore	Bits set to 0	Yes
15	Reserved by PKWARE	Ignore	Bits set to 0	Yes

Table C–6, “Support for Extra field (variable size), PKWARE-reserved”, specifies the detailed production, consumption, and editing requirements for the Extra field entries reserved by PKWARE and described in the ZIP Appnote.txt.

Table C–6. Support for Extra field (variable size), PKWARE-reserved

Field ID	Field description	Supported on Consumption	Supported on Production	Pass through on editing
0x0001	ZIP64 extended information extra field	Yes	Yes	Optional
0x0007	AV Info	Ignore	No	Yes
0x0008	Reserved for future Unicode file name data (PFS)	Ignore	No	Yes
0x0009	OS/2	Ignore	No	Yes
0x000a	NTFS	Ignore	No	Yes
0x000c	OpenVMS	Ignore	No	Yes
0x000d	Unix	Ignore	No	Yes
0x000e	Reserved for file stream and fork descriptors	Ignore	No	Yes
0x000f	Patch Descriptor	Ignore	No	Yes
0x0014	PKCS#7 Store for X.509 Certificates	Ignore	No	Yes
0x0015	X.509 Certificate ID and Signature for individual file	Ignore	No	Yes

Field ID	Field description	Supported on Consumption	Supported on Production	Pass through on editing
0x0016	X.509 Certificate ID for Central Directory	Ignore	No	Yes
0x0017	Strong Encryption Header	Ignore	No	Yes
0x0018	Record Management Controls	Ignore	No	Yes
0x0019	PKCS#7 Encryption Recipient Certificate List	Ignore	No	Yes
0x0065	IBM S/390 (Z390), AS/400 (I400) attributes — uncompressed	Ignore	No	Yes
0x0066	Reserved for IBM S/390 (Z390), AS/400 (I400) attributes — compressed	Ignore	No	Yes
0x4690	POSZIP 4690 (reserved)	Ignore	No	Yes

Table C–7, “Support for Extra field (variable size), third-party extensions”, specifies the detailed production, consumption, and editing requirements for the Extra field entries reserved by third parties and described in the ZIP Appnote.txt.

Table C–7. Support for Extra field (variable size), third-party extensions

Field ID	Field description	Supported on Consumption	Supported on Production	Pass through on editing
0x07c8	Macintosh	Ignore	No	Yes
0x2605	ZipIt Macintosh	Ignore	No	Yes
0x2705	ZipIt Macintosh 1.3.5+	Ignore	No	Yes
0x2805	ZipIt Macintosh 1.3.5+	Ignore	No	Yes
0x334d	Info-ZIP Macintosh	Ignore	No	Yes
0x4341	Acorn/SparkFS	Ignore	No	Yes
0x4453	Windows NT security descriptor (binary ACL)	Ignore	No	Yes
0x4704	VM/CMS	Ignore	No	Yes
0x470f	MVS	Ignore	No	Yes
0x4b46	FWKCS MD5 (see below)	Ignore	No	Yes

Field ID	Field description	Supported on Consumption	Supported on Production	Pass through on editing
0x4c41	OS/2 access control list (text ACL)	Ignore	No	Yes
0x4d49	Info-ZIP OpenVMS	Ignore	No	Yes
0x4f4c	Xceed original location extra field	Ignore	No	Yes
0x5356	AOS/VS (ACL)	Ignore	No	Yes
0x5455	extended timestamp	Ignore	No	Yes
0x554e	Xceed unicode extra field	Ignore	No	Yes
0x5855	Info-ZIP Unix (original, also OS/2, NT, etc)	Ignore	No	Yes
0x6542	BeOS/BeBox	Ignore	No	Yes
0x756e	ASi Unix	Ignore	No	Yes
0x7855	Info-ZIP Unix (new)	Ignore	No	Yes
0xa220	Padding, Microsoft	Optional	Optional	Optional
0xfd4a	SMS/QDOS	Ignore	No	Yes

The package implementer shall ensure that all 64-bit stream record sizes and offsets have the high-order bit = 0. [M3.20]

The package implementer shall ensure that all fields that contain “number of entries” do not exceed 2,147,483,647. [M3.21]

Annex D. Schemas - XML Schema

This Open Packaging Conventions specification includes a family of schemas defined using the XML Schema 1.0 syntax. The normative definitions of these schemas reside in an accompanying file named OpenPackagingConventions-XMLSchema.zip, which is distributed in electronic form only.

If discrepancies exist between the electronic version of a schema and its corresponding representation as published in this part, Part 2, the electronic version is the definitive version.

Annex E. Schemas - RELAX NG

This clause is informative.

This Open Packaging Conventions specification includes a family of schemas defined using the RELAX NG syntax. The definitions of these schemas reside in an accompanying file named OpenPackagingConventions-RELAXNG.zip, which is distributed in electronic form only.

If discrepancies exist between the RELAX NG version of a schema and its corresponding XML Schema, the XML Schema is the definitive version.

End of informative text.

Annex F. Standard Namespaces and Content Types

The namespaces available for use in a package are listed in Table F–1, Package-wide namespaces

Table F–1. Package-wide namespaces

Description	Namespace URI
Content Types	http://schemas.openxmlformats.org/package/2006/content-types
Core Properties	http://schemas.openxmlformats.org/package/2006/metadata/core-properties
Digital Signatures	http://schemas.openxmlformats.org/package/2006/digital-signature
Relationships	http://schemas.openxmlformats.org/package/2006/relationships
Markup Compatibility	http://schemas.openxmlformats.org/markup-compatibility/2006

The content types available for use in a package are listed in Table F–2, Package-wide content types

Table F–2. Package-wide content types

Description	Content Type
Core Properties part	application/vnd.openxmlformats-package.core-properties+xml
Digital Signature Certificate part	application/vnd.openxmlformats-package.digital-signature-certificate
Digital Signature Origin part	application/vnd.openxmlformats-package.digital-signature-origin
Digital Signature XML Signature part	application/vnd.openxmlformats-package.digital-signature-xmlsignature+xml
Relationships part	application/vnd.openxmlformats-package.relationships+xml

Package implementers and format designers shall not create content types with parameters for the package-specific parts defined in this Open Packaging specification and shall treat the presence of parameters in these content types as an error. [M1.22]

The relationship types available for use in a package are listed in Table F–3, Package-wide relationship types.

Table F–3. Package-wide relationship types

Description	Relationship Type
Core Properties	http://schemas.openxmlformats.org/package/2006/relationships/metadata/core-properties
Digital Signature	http://schemas.openxmlformats.org/package/2006/relationships/digital-signature/signature

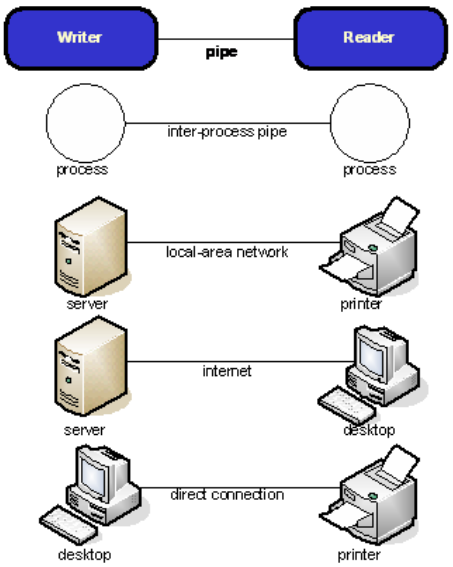
Description	Relationship Type
Digital Signature Certificate	http://schemas.openxmlformats.org/package/2006/relationships/digital-signature/certificate
Digital Signature Origin	http://schemas.openxmlformats.org/package/2006/relationships/digital-signature/origin
Thumbnail	http://schemas.openxmlformats.org/package/2006/relationships/metadata/thumbnail

Annex G. Physical Model Design Considerations

This annex is informative.

The physical model defines the ways in which packages are produced and consumed. This model is based on three components: a producer, a consumer, and a pipe between them.

Figure G–1. Components of the physical model



A *producer* is a piece of software or a device that *writes* packages. A *consumer* is a piece of software or a device that *reads* packages. A *device* is a piece of hardware, such as a printer or scanner that performs a single function or set of functions. Data is carried from the producer to the consumer by a *pipe*.

In *local access*, the pipe carries data directly from a producer to a consumer on a single device.

In *networked access* the consumer and the producer communicate with each other over a protocol. The significant communication characteristics of this pipe are speed and request latency. For example, this communication might occur across a process boundary or between a server and a desktop computer.

In order to maximize performance, designers of physical package formats consider access style, layout style, and communication style.

G.1 Access Styles

The *access style* in which local access or networked access is conducted determines the simultaneity possible between processing and input-output operations.

G.1.1 Direct Access Consumption

Direct access consumption allows consumers to request the specific portion of the package desired, without sequentially processing the preceding parts of the package. For example a byte-range request. This is the most common access style.

G.1.2 Streaming Consumption

Streaming consumption allows consumers to begin processing parts before the entire package has arrived. Physical package formats should be designed to allow consumers to begin interpreting and processing the data they receive before all of the bits of the package have been delivered through the pipe.

G.1.3 Streaming Creation

Streaming creation allows producers to begin writing parts to the package without knowing in advance all of the parts that will be written. For example, when an application begins to build a print spool file package, it may not know how many pages the package will contain. Likewise, a program that is generating a report may not know initially how long the report will be or how many pictures it will have.

In order to support streaming creation, the package implementer should allow a producer to add parts after other parts have already been added. A Consumer shall not require a producer to state how many parts they will create when they start writing. The package implementer should allow a producer to begin writing the contents of a part without knowing the ultimate length of the part.

G.1.4 Simultaneous Creation and Consumption

Simultaneous creation and consumption allows streaming creation and streaming consumption to happen at the same time on a package. Because of the benefits that can be realized within pipelined architectures that use it, the package implementer should support simultaneous creation and consumption in the physical package.

G.2 Layout Styles

The style in which parts are ordered within a package is referred to as the *layout style*. Parts can be arranged in one of two styles: simple ordering or interleaved ordering.

G.2.1 Simple Ordering

With *simple ordering*, parts are arranged contiguously. When a package is delivered sequentially, all of the bytes for the first part arrive first, followed by all of the bytes for the second part, and so on. When such a package uses simple ordering, all of the bytes for each part are stored contiguously.

G.2.2 Interleaved Ordering

With *interleaved ordering*, pieces of parts are interleaved, allowing optimal performance in certain scenarios. For example, interleaved ordering improves performance for multi-media playback, where video and audio are delivered simultaneously and inline resource referencing, where a reference to an image occurs within markup.

By breaking parts into pieces and interleaving those pieces, it is possible to optimize performance while allowing easy reconstruction of the original contiguous part.

Because of the performance benefits it provides, package implementers should support interleaving in the physical package. The package implementer might handle the internal representation of interleaving differently in different physical models. Regardless of how the physical model handles interleaving, a part that is broken into multiple pieces in the physical file is considered one logical part; the pieces themselves are not parts and are not addressable.

G.3 Communication Styles

The style in which a package and its parts are delivered by a producer or accessed by a consumer is referred to as the *communication style*. Communication can be based on sequential delivery of or random access to parts. The communication style used depends on the capabilities of both the pipe and the physical package format.

G.3.1 Sequential Delivery

With *sequential delivery*, all of the physical bits in the package are delivered in the order they appear in the. Generally, all pipes support sequential delivery.

G.3.2 Random Access

Random access allows consumers to request the delivery of a part out of sequential physical order. Some pipes are based on protocols that can enable random access. For example, HTTP 1.1 with byte-range support. In order to maximize performance, the package implementer should support random access in both the pipe and the physical package. In the absence of this support, consumers need to wait until the parts they need are delivered sequentially.

End of informative text.

Annex H. Conformance Requirements

This annex is informative.

This annex summarizes all conformance requirements for producers and consumers implementing the Open Packaging Conventions. It is intended as a convenience; the text in the referenced clause or subclause is considered normative in all cases.

Conformance requirements are divided into tables based on their general topic below. The tables contain the requirements that producers and consumers shall follow, those that they should follow, and those that are optional. Each conformance requirement is given a unique ID comprised of a letter (M – MANDATORY; S – SHOULD; O – OPTIONAL), an identifier for the topic it relates to, and a unique ID within that topic. Mandatory requirements are those stated with the normative terms "shall," "shall not," or any of their normative equivalents. Should items are those stated with the normative terms "should," "should not," or any of their normative equivalents. Optional requirements are those stated with the normative terms "can," "cannot," "might," "might not," or any of their normative equivalents.

Producers and consumers might use these IDs to report error conditions.

The top-level topics and their identifiers are described as follows:

1. Package Model requirements
2. Physical Packages requirements
3. ZIP Physical Mapping requirements
4. Core Properties requirements
5. Thumbnail requirements
6. Digital Signatures requirements
7. Pack URI requirements

Additionally, these tables identify, as does the referenced text, who is burdened with enforcing or supporting the requirement:

H.1 Package Model

Table H–1. Package model conformance requirements

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M1.1	The package implementer shall require a part name.	8.1, 8.1.1	×			

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M1.2	The package implementer shall require a content type and the format designer shall specify the content type.	8.1	×	×		
M1.3	A part name shall not have empty segments.	0	×			
M1.4	A part name shall start with a forward slash (“/”) character.	0	×			
M1.5	A part name shall not have a forward slash as the last character.	0	×			
M1.6	A segment shall not hold any characters other than pchar characters. .	0	×			
M1.7	A segment shall not contain percent-encoded forward slash (“/”), or backward slash (“\”) characters.	0	×			
M1.8	A segment shall not contain percent-encoded unreserved characters.	0	×			
M1.9	A segment shall not end with a dot (“.”) character.	0	×			
M1.10	A segment shall include at least one non-dot character	0	×			
M1.11	A package implementer shall neither create nor recognize a part with a part name derived from another part name by appending segments to it.	8.1.1.1	×			
M1.12	Part name equivalence is determined by comparing part names as case-insensitive ASCII strings. Packages shall not contain equivalent part names and package implementers shall neither create nor recognize packages with equivalent part names.	8.1.1.2	×			

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M1.13	Package implementers shall only create and only recognize parts with a content type; format designers shall specify a content type for each part included in the format. Content types for package parts shall fit the definition and syntax for media types as specified in RFC 2616, §3.7.	8.1.2	×	×		
M1.14	Content types shall not use linear white space either between the type and subtype or between an attribute and its value. Content types also shall not have leading or trailing white spaces. Package implementers shall create only such content types and shall require such content types when retrieving a part from a package; format designers shall specify only such content types for inclusion in the format.	8.1.2	×	×		
M1.15	The package implementer shall require a content type that does not include comments and the format designer shall specify such a content type.	8.1.2	×	×		
M1.16	If the package implementer specifies a growth hint, it is set when a part is created and the package implementer shall not change the growth hint after the part has been created.	8.1.3	×		×	

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M1.17	XML content shall be encoded using either UTF-8 or UTF-16. If any part includes an encoding declaration, as defined in §4.3.3 of the XML 1.0 specification, that declaration shall not name any encoding other than UTF-8 or UTF-16. Package implementers shall enforce this requirement upon creation and retrieval of the XML content.	8.1.4	×			
M1.18	DTD declarations shall not be used in the XML markup defined in this Open Packaging specification. Package implementers shall enforce this requirement upon creation and retrieval of the XML content and shall treat the presence of DTD declarations as an error.	8.1.4	×			
M1.19	If the XML content contains the Markup Compatibility namespace, as described in Part 5: “Markup Compatibility and Extensibility”, it shall be processed by the package implementer to remove Markup Compatibility elements and attributes, ignorable namespace declarations, and ignored elements and attributes before applying subsequent validation rules.	8.1.4	×			

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M1.20	XML content shall be valid against the corresponding XSD schema defined in this Open Packaging specification. In particular, the XML content shall not contain elements or attributes drawn from namespaces that are not explicitly defined in the corresponding XSD unless the XSD allows elements or attributes drawn from any namespace to be present in particular locations in the XML markup. Package implementers shall enforce this requirement upon creation and retrieval of the XML content.	8.1.4	×			
M1.21	XML content shall not contain elements or attributes drawn from “xml” or “xsi” namespaces unless they are explicitly defined in the XSD schema or by other means described in this Open Packaging specification. Package implementers shall enforce this requirement upon creation and retrieval of the XML content.	8.1.4	×			
M1.22	Package implementers and format designers shall not create content types with parameters for the package-specific parts defined in this Open Packaging specification and shall treat the presence of parameters in these content types as an error.	Annex F	×	×		

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M1.23	XML markup might contain Unicode strings referencing other parts as values of the xsd:anyURI data type. Format consumers shall convert these Unicode strings to URIs, as defined in Annex A, “Resolving Unicode Strings to Part Names,” before resolving them relative to the base URI of the part containing the Unicode string.	8.2.1				×
M1.24	Some types of content provide a way to override the default base URI by specifying a different base in the content. In the presence of one of these overrides, format consumers shall use the specified base URI instead of the default.	8.2.1				×
M1.25	The Relationships part shall not have relationships to any other part. Package implementers shall enforce this requirement upon the attempt to create such a relationship and shall treat any such relationship as invalid.	8.3.1	×			
M1.26	The package implementer shall require that every Relationship element has an Id attribute, the value of which is unique within the Relationships part, and that the Id type is xsd:ID, the value of which conforms to the naming restrictions for xsd:ID as described in the W3C Recommendation “XML Schema Part 2: Datatypes.”	8.3.3	×			

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M1.27	The package implementer shall require the Type attribute to be a URI that defines the role of the relationship and the format designer shall specify such a Type.	8.3.3.2	×	×		
M1.28	The package implementer shall require the Target attribute to be a URI reference pointing to a target resource. The URI reference shall be a URI or a relative reference.	8.3.3.2	×			
M1.29	When set to Internal, the Target attribute shall be a relative reference and that reference is interpreted relative to the “parent” part. For package relationships, the package implementer shall resolve relative references in the Target attribute against the pack URI that identifies the entire package resource.	8.3.3.2	×			
M1.30	The package implementer shall name relationship parts according to the special relationships part naming convention and require that parts with names that conform to this naming convention have the content type for a Relationships part	8.3.4	×			

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M1.31	Consumers shall process relationship markup in a manner that conforms to Part 5: “Markup Compatibility and Extensibility”. Producers editing relationships based on this version of the relationship markup specification shall not preserve any ignored content, regardless of the presence of any preservation attributes as defined in Part 5: “Markup Compatibility and Extensibility”.	8.3.5			×	×
M1.32	If a fragment identifier is allowed in the Target attribute of the Relationship element, a package implementer shall not resolve the URI to a scope less than an entire part.	8.3.3.2	×			
M1.33	A Unicode string representing a URI can be passed to the producer or consumer. The producing or consuming application shall convert the Unicode string to a URI. If the URI is a relative reference, the application shall resolve it using the base URI of the part, which is expressed using the pack scheme, to the URI of the referenced part.	Annex A			×	×
M1.34	If a consumer converts the URI back into an IRI, the conversion shall be performed as specified in §3.2 of RFC 3987.	A.2				×

Table H–2. Package model optional requirements

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
O1.1	The package implementer might allow a growth hint to be provided by a producer.	8.1, 8.1.3	×			

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
O1.2	Format designers might restrict the usage of parameters for content types.	8.1.2		×		
O1.3	The package implementer might ignore the growth hint or adhere only loosely to it when specifying the physical mapping.	8.1.3	×			
O1.4	If the format designer permits it, parts can contain Unicode strings representing references to other parts. If allowed by the format designer, format producers can create such parts and format consumers shall consume them.	8.2.1		×	×	×
O1.5	The package implementer might allow a TargetMode to be provided by a producer.	8.3.3.2	×			
O1.6	A format designer might allow fragment identifiers in the value of the Target attribute of the Relationship element.	8.3.3.2		×		
O1.7	Producers might generate relationship markup that uses the versioning and extensibility mechanisms defined in Part 5: “Markup Compatibility and Extensibility” to incorporate elements and attributes drawn from other XML namespaces.	8.3.5			×	

H.2 Physical Packages

Table H–3. Physical packages conformance requirements

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M2.1	The Content Types stream shall not be mapped to a part by the package implementer.	9.1.2.1	×[A]			

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M2.2	The package implementer shall define a physical package format with a mapping for the required components package, part name, part content type and part contents.	9.1.1	×			
M2.3	The package implementer shall define a format mapping with a mechanism for associating content types with parts.	9.1.2.1	×			
M2.4	The package implementer shall require that the Content Types stream contain one of the following for every part in the package: One matching Default element One matching Override element Both a matching Default element and a matching Override element, in which case the Override element takes precedence.	9.1.2.2	×[A]			
M2.5	The package implementer shall require that there not be more than one Default element for any given extension, and there not be more than one Override element for any given part name.	9.1.2.2	×[A]			
M2.6	The package implementer shall require a non-empty extension in a Default element. The package implementer shall require a content type in a Default element and the format designer shall specify the content type.	9.1.2.2.2	×[A]	×[A]		
M2.7	The package implementer shall require a content type and the format designer shall specify the content type in an Override element. The package implementer shall require a part name.	9.1.2.2.3	×[A]	×[A]		

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M2.8	When adding a new part to a package, the package implementer shall ensure that a content type for that part is specified in the Content Types stream; the package implementer shall perform the steps described in §9.1.2.3.	9.1.2.3	×[A]			
M2.9	To get the content type of a part, the package implementer shall perform the steps described in §9.1.2.4.	9.1.2.4	×[A]			
M2.10	The package implementer shall not use the versioning and extensibility mechanisms defined in Part 5: "Markup Compatibility and Extensibility" to incorporate elements and attributes drawn from other XML-namespaces into the Content Types stream markup.	9.1.2.5	×[A]			
M2.11	The package implementer shall not mix interleaving and non-interleaving for an individual part.	9.1.4	×[B]			
M2.12	The package implementer shall compare prefix names as case-insensitive ASCII strings.	9.1.3.1	×			
M2.13	The package implementer shall compare suffix names as case-insensitive ASCII strings.	9.1.3.1	×[B]			
M2.14	The package implementer shall not allow packages that contain equivalent logical item names.	9.1.3.1	×			
M2.15	The package implementer shall not allow packages that contain logical items with equivalent prefix names and with equal piece numbers, where piece numbers are treated as integer decimal values.	9.1.3.1	×[B]			
M2.16	The package implementer shall not map logical items to parts if the logical item names violate the part naming rules.	9.1.3.4	×			

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M2.17	The package implementer shall consider naming collisions within the set of part names mapped from logical item names to be an error.	9.1.3.4	×			
M2.18	When interleaved, a package implementer shall represent a part as one or more pieces, using the method described in §9.1.4.	9.2.1	×[B]			

Notes:

A: Only relevant if using the content type mapping strategy specified in the Open Packaging Conventions.

B: Only relevant if supporting the interleaving strategy specified in the Open Packaging Conventions.

Table H–4. Physical packages recommendations

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
S2.1	Some physical package formats have a native mechanism for representing content types. For such packages, the package implementer should use the native mechanism to map the content type for a part.	9.1.2.1	×			
S2.2	If no native method of mapping a content type to a part exists, the package implementer should include a specially-named XML stream in the package called the Content Types stream	9.1.2.1	×			

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
S2.3	If the package is intended for streaming consumption: The package implementer should not allow Default elements; as a consequence, there should be one Override element for each part in the package. The format producer should write the Override elements to the package so they appear before the parts to which they correspond, or in close proximity to the part to which they correspond.	9.1.2.2	×[A]		×[A]	
S2.4	The package implementer should use the mechanism described in this Open Packaging specification to allow interleaving when mapping to the physical package for layout scenarios that support streaming consumption.	9.1.4	×[B]			
S2.5	The package implementer should store pieces in their natural order for optimal efficiency.	9.1.4	×[B]			

Notes:

A: Only relevant if using the content type mapping strategy specified in the Open Packaging Conventions.

B: Only relevant if supporting the interleaving strategy specified in the Open Packaging Conventions.

Table H–5. Physical packages optional requirements

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
O2.1	The format designer specifies whether that format might use interleaving.	9.1.4		×		

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
O2.2	Optional. The package implementer might provide a physical mapping for a growth hint that might be specified by a producer.	9.1.1	×			
O2.3	Package implementers might use the common mapping solutions defined in this Open Packaging specification.	9.1	×			
O2.4	Package producers can use pre-defined Default elements to reduce the number of Override elements on a part, but are not required to do so.	9.1.2.2			×[A]	
O2.5	The package implementer can define Default content type mappings even though no parts use them.	9.1.2.2	×[A]			
O2.6	The package implementer might create a physical package containing interleaved parts and non-interleaved parts.	9.1.4	×			
O2.7	The package implementer might allow a package that contains logical item names and complete sequences of logical item names that cannot be mapped to a part name because the logical item name does not follow the part naming grammar or the logical item does not have an associated content type.	9.1.3.4	×[B]			

Notes:

A: Only relevant if using the content type mapping strategy specified in the Open Packaging Conventions.

B: Only relevant if supporting the interleaving strategy specified in the Open Packaging Conventions.

H.3 ZIP Physical Mapping

The requirements in Table H–6, Table H–7, and

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
S3.2	If a growth hint is used for an interleaved part, the package implementer should store the Extra field containing the growth hint padding with the item that represents the first piece of the part.	10.2.7	×			

Table J–8 are only relevant when mapping to the ZIP physical package format.

Table H–6. ZIP physical mapping conformance requirements

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M3.1	A package implementer shall store a non-interleaved part as a single ZIP item.	9.2.1	×			
M3.2	ZIP item names are case-sensitive ASCII strings. Package implementers shall create ZIP item names that conform to ZIP archive file name grammar.	9.2.2	×			
M3.3	Package implementers shall create item names that are unique within a given archive.	9.2.2	×			
M3.4	To map part names to ZIP item names the package implementer shall perform, in order, the steps described in §9.2.3.	9.2.3	×			
M3.5	The package implementer shall not map a logical item name or complete sequence of logical item names sharing a common prefix to a part name if the logical item prefix has no corresponding content type.	9.2.3	×			
M3.6	To map ZIP item names to part names, the package implementer shall perform, in order, the steps described in §9.2.4.	9.2.4	×			

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M3.7	The package implementer shall map all ZIP items to parts except MS-DOSZIP items, as defined in the ZIP specification, that are not MS-DOS files.	9.2.5	×			
M3.8	The package implementer shall map all ZIP items to parts except MS-DOSZIP items, as defined in the ZIP specification, that are not MS-DOS files. [M3.7] [*Note:* The ZIP specification specifies that ZIP items recognized as MS-DOS files are those with a "version made by" field and an "external file attributes" field in the "file header" record in the central directory that have a value of 0. *end note]* In ZIP archives, the package implementer shall not exceed 65,535 bytes for the combined length of the item name, Extra field, and Comment fields.	9.2.5	×			
M3.9	ZIP-based packages shall not include encryption as described in the ZIP specification. Package implementers shall enforce this restriction.	9.2.5	×			
M3.10	Package implementers shall store content type data in an item(s) mapped to the logical item name with the prefix_name equal to "/[Content_Types].xml" or in the interleaved case to the complete sequence of logical item names with that prefix_name.	9.2.6	×			
M3.11	Package implementers shall not map logical item name(s) mapped to the Content Types stream in a ZIP archive to a part name.	9.2.6	×			

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M3.13	Several substantial conditions that represent a package unfit for streaming consumption may be detected mid-processing by a streaming package implementer, described in §9.2.8. When any of these conditions are detected, the streaming package implementer shall generate an error, regardless of any processing that has already taken place. Package implementers shall not generate a package containing any of these conditions when generating a package intended for streaming consumption.	9.2.8	×			
M3.14	For a ZIP archive to be a valid physical layer for a package, the package implementer shall ensure that the ZIP archive holds equal values in the appropriate fields of every File Header within the Central Directory and the corresponding Local File Header and Data Descriptor pair.	Annex C	×			
M3.15	During consumption of a package, a "Yes" value for a field in a table in Annex C indicates a package implementer shall support reading the ZIP archive containing this record or field, however, support may mean ignoring.	Annex C	×			
M3.16	During production of a package, a “Yes” value for a field in a table in Annex C indicates that the package implementer shall write out this record or field.	Annex C	×			
M3.17	A “No” value for a field in a table in Annex C indicates the package implementer shall not use this record or field during consumption or production of packages.	Annex C	×			

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M3.18	A “Partially, details below” value for a record in a table in Annex C indicates that the record contains fields that might not be supported by package implementers during production or consumption. See the details in the corresponding table to determine requirements.	Annex C	×			
M3.19	The value “Only used when needed” associated with a record in a table in Annex C indicates that the package implementer shall use the record only when needed to store data in the ZIP archive.	Annex C	×			
M3.20	The value “Only used when needed” associated with a record in a table in Annex C indicates that the package implementer shall use the record only when needed to store data in the ZIP archive.	Annex C	×			
M3.21	The package implementer shall ensure that all 64-bit stream record sizes and offsets have the high-order bit = 0.	Annex C	×			

Notes:

A: Only relevant if supporting the interleaving strategy specified in the Open Packaging Conventions.

Table H–7. ZIP physical mapping recommendations

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
S3.1	Package implementers should restrict part naming to accommodate file system limitations when naming parts to be stored as ZIP items.	9.2.5	×			
S3.2	If a growth hint is used for an interleaved part, the package implementer should store the Extra field containing the growth hint padding with the item that represents the first piece of the part.	9.2.7	×			

Table H–8. ZIP physical mapping optional requirements

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
O3.1	A package implementer might intentionally order the sequence of ZIP items in the archive to enable an efficient organization of the part data in order to achieve correct and optimal interleaving.	9.2.1	×			
O3.2	An “Optional” value for a record in a table in Annex C indicates that package implementers might write this record during production.	Annex C	×			

H.4 Core Properties

The requirements in Table H–9 are only relevant if using the core properties feature.

Table H–9. Core properties conformance requirements

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M4.1	The format designer shall specify and the format producer shall create at most one core properties relationship for a package. A format consumer shall consider more than one core properties relationship for a package to be an error. If present, the relationship shall target the Core Properties part.	10.2		×	×	×
M4.2	The format designer shall not specify and the format producer shall not create Core Properties that use the Markup Compatibility namespace as defined in Annex F, “Standard Namespaces and Content Types”. A format consumer shall consider the use of the Markup Compatibility namespace to be an error.	10.3		×	×	×

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M4.3	Producers shall not create a document element that contains refinements to the Dublin Core elements, except for the two specified in the schema: <dcterms:created> and <dcterms:modified> Consumers shall consider a document element that violates this constraint to be an error.	10.4			×	×
M4.4	Producers shall not create a document element that contains the xml:lang attribute. Consumers shall consider a document element that violates this constraint to be an error.	10.4			×	×
M4.5	Producers shall not create a document element that contains the xsi:type attribute, except for a <dcterms:created> or <dcterms:modified> element where the xsi:type attribute shall be present and shall hold the value dcterms:W3CDTF, where `dcterms` is the namespace prefix of the Dublin Core namespace. Consumers shall consider a document element that violates this constraint to be an error.	10.4			×	×

H.5 Thumbnail

The requirements in Table H–10 and Table H–11 are only relevant if using the thumbnail feature.

Table H–10. Thumbnail conformance requirements

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M5.1	The format designer shall specify thumbnail parts that are identified by either a part relationship or a package relationship. The producer shall build the package accordingly.	11.1		×	×	

Table H–11. Thumbnail optional requirements

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
O5.1	The format designer might allow images, called thumbnails, to be used to help end-users identify parts of a package or a package as a whole. These images can be generated by the producer and stored as parts.	11		×	×	

H.6 Digital Signatures

The requirements in Table H–12, Table H–13, and Table H–14 are only relevant if using the digital signatures feature.

Table H–12. Digital Signatures conformance requirements

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M6.1	The package implementer shall include only one Digital Signature Origin part in a package and it shall be targeted from the package root using the well-defined relationship type specified in Annex F, “Standard Namespaces and Content Types”.	12.2.1	×			
M6.2	When creating the first Digital Signature XML Signature part, the package implementer shall create the Digital Signature Origin part, if it does not exist, in order to specify a relationship to that Digital Signature XML Signature part.	12.2.1	×			
M6.3	The producer shall create Digital Signature XML Signature parts that have a relationship from the Digital Signature Origin part and the consumer shall use that relationship to locate signature information within the package.	12.2.1			×	×

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M6.4	If the certificate is represented as a separate part within the package, the producer shall target that certificate from the appropriate Digital Signature XML Signature part by a Digital Signature Certificate relationship as specified in Annex F, “Standard Namespaces and Content Types” and the consumer shall use that relationship to locate the certificate.	12.2.3			×	×
M6.5	The producer shall create Reference elements within a SignedInfo element that reference elements within the same Signature element. The consumer shall consider Reference elements within a SignedInfo element that reference any resources outside the same Signature element to be in error.	12.2.4.1			×	×
M6.6	The producer shall not create a reference to a package-specific Object element that contains a transform other than a canonicalization transform. The consumer shall consider a reference to a package-specific Object element that contains a transform other than a canonical transform to be an error.	12.2.4.1			×	×
M6.7	The producer shall create one and only one package-specific Object element in the Signature element. The consumer shall consider zero or more than one package-specific Object element in the Signature element to be an error.	12.2.4.1			×	×

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M6.8	The producer shall create package-specific Object elements that contain exactly one Manifest element and exactly one SignatureProperties element. [*Note:* This SignatureProperties element can contain multiple SignatureProperty elements. *end note*] The consumer shall consider package-specific Object elements that contain other types of elements to be an error.	12.2.4.1			×	×
M6.9	The producer shall create Reference elements within a Manifest element that reference with their URI attribute only parts within the package. The consumer shall consider Reference elements within a Manifest element that reference resources outside the package to be an error.	12.2.4.1			×	×
M6.10	The producer shall create relative references to the local parts that have query components that specifies the part content type as described in §12.2.4.6. The relative reference excluding the query component shall conform to the part name grammar. The consumer shall consider a relative reference to a local part that has a query component that incorrectly specifies the part content type to be an error.	12.2.4.1			×	×

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M6.11	The producer shall create Reference elements with a query component that specifies the content type that matches the content type of the referenced part. The consumer shall consider signature validation to fail if the part content type compared in a case-sensitive manner to the content type specified in the query component of the part reference does not match.	12.2.4.1			×	×
M6.12	The producer shall not create Reference elements within a Manifest element that contain transforms other than the canonicalization transform and relationships transform. The consumer shall consider Reference elements within a Manifest element that contain transforms other than the canonicalization transform and relationships transform to be in error.	12.2.4.1			×	×
M6.13	A producer that uses an optional relationships transform shall follow it by a canonicalization transform. The consumer shall consider any relationships transform that is not followed by a canonicalization transform to be an error.	12.2.4.1			×	×

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M6.14	The producer shall create exactly one SignatureProperty element with the Id attribute value set to `idSignatureTime`. The Target attribute value of this element shall be either empty or contain a fragment reference to the value of the Id attribute of the root Signature element. A SignatureProperty element shall contain exactly one SignatureTime child element. The consumer shall consider a SignatureProperty element that does not contain a SignatureTime element or whose Target attribute value is not empty or does not contain a fragment reference the Id attribute of the ancestor Signature element to be in error.	12.2.4.1			×	×
M6.15	The producer shall create a Signature element that contains exactly one local-data, package-specific Object element and zero or more application-specific Object elements. If a Signature element violates this constraint, a consumer shall consider this to be an error.	12.2.4.2			×	×
M6.16	The producer shall create a SignedInfo element that contains exactly one reference to the package-specific Object element. The consumer shall consider it an error if a SignedInfo element does not contain a reference to the package-specific Object element.	12.2.4.3			×	×
M6.17	Producers shall support DSA and RSA algorithms to produce signatures. Consumers shall support DSA and RSA algorithms to validate signatures.	12.2.4.5			×	×

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M6.18	The producer shall create a Reference element within a Manifest element with a URI attribute and that attribute shall contain a part name, without a fragment identifier. The consumer shall consider a Reference element with a URI attribute that does not contain a part name to be an error.	12.2.4.6			×	×
M6.19	The following transforms shall be supported by producers and consumers of packages with digital signatures: • XML Canonicalization (c14n) • XML Canonicalization with Comments (c14n with comments) • Relationships transform (package-specific) Consumers validating signed packages shall fail the validation if other transforms are encountered. Relationships transforms shall only be supported by producers and consumers when the Transform element is a descendant element of a Manifest element	12.2.4.7			×	×
M6.20	Producers shall create application-specific Object elements that contain XML-compliant data; consumers shall treat data that is not XML-compliant as an error.	12.2.4.14			×	×
M6.21	Producers and consumers shall use the certificate embedded in the Digital Signature XML Signature part when it is specified.	12.2.4.15			×	×

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M6.22	The producer shall not create a Manifest element that references any data outside of the package. The consumer shall consider a Manifest element that references data outside of the package to be in error.	12.2.4.18			×	×
M6.23	The producer shall create a data/time format that conforms to the syntax described in the W3C Note "Date and Time Formats". The consumer shall consider a format that does not conform to the syntax described in that WC3 note to be in error.	12.2.4.22			×	×
M6.24	The producer shall create a value that conforms to the format specified in the Format element. The consumer shall consider a value that does not conform to that format to be in error.	12.2.4.23			×	×
M6.25	To sign a subset of relationships, the producer shall use the package-specific relationships transform. The consumer shall use the package-specific relationships transform to validate the signature when a subset of relationships are signed.	12.2.4.25			×	×
M6.26	Producers shall specify a canonicalization transform immediately following a relationships transform and consumers that encounter a relationships transform that is not immediately followed by a canonicalization transform shall generate an error.	12.2.4.25			×	×

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M6.27	When applying a relationships transform for digital signatures, the package implementer shall remove all Relationship elements that do not have eitheran Id value that matches any SourceId valueor a Type value that matches any SourceType value, among the SourceId and SourceType values specified in the transform definition. Producers and consumers shall compare values as case-sensitive Unicode strings.	12.2.4.26			×	×
M6.28	When signing Object element data, package implementers shall follow the generic reference creation algorithm described in §3.1 of the W3C Recommendation “XML-Signature Syntax and Processing”.	12.4	×			
M6.29	When validating digital signatures, consumers shall verify the content type and the digest contained in each Reference descendant element of the SignedInfo element, and validate the signature calculated using the SignedInfo element.	12.5				×
M6.30	The package implementer shall compare the generated digest value against the DigestValue element in the Reference element of the SignedInfo element. Package implementers shall consider references invalid if there is any mismatch.	12.5	×			
M6.31	Streaming consumers that maintain signatures shall be able to cache the parts necessary for detecting and processing signatures.	12.5.1				×

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M6.32	The package implementer shall not use the Markup Compatibility namespace, as specified in Annex F, "Standard Namespaces and Content Types," within the package-specific Object element. The package implementer shall consider the use of the Markup Compatibility namespace within the package-specific Object element to be an error.	12.6.2	×			
M6.33	If an application allows for a single part to contain information that might not be fully understood by all implementations, then the format designer shall carefully design the signing and verification policies to account for the possibility of different implementations being used for each action in the sequence of content creation, content signing, and signature verification. Producers and consumers shall account for this possibility in their signing and verification processing.	12.6.2		×	×	×
M6.34	The following canonicalization methods shall be supported by producers and consumers of packages with digital signatures: XML Canonicalization (c14n) XML Canonicalization with Comments (c14n with comments) Consumers validating signed packages shall fail the validation if other canonicalization methods are encountered.	12.2.4.4			×	×
M6.35	A producer shall not specify more than one relationship transform for a particular relationships part. A consumer shall treat the presence of more than one relationship transform for a particular relationships part as an error.	12.2.4.25			×	×

Table H–13. Digital signatures recommendations

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
S6.1	The producer should not create any content in the Digital Signature Origin part itself.	12.2.1			×	
S6.2	Producers generating digital signatures should not create Digital Signature Certificate parts that are not the target of at least one Digital Signature Certificate relationship from a Digital Signature XML Signature part. In addition, producers should remove a Digital Signature Certificate part if removing the last Digital Signature XML Signature part that has a Digital Signature Certificate relationship to it.	12.2.3			×	
S6.3	For digital signatures, a producer should apply a canonicalization transform to the SignedInfo element when it generates it, and a consumer should apply the canonicalization transform to the SignedInfo element when validating it.	12.2.4.4			×	×
S6.4	Producers and consumers should also use canonicalization transforms for references to parts that hold XML documents.	12.2.4.4			×	×
S6.5	The producer should only create Reference elements within a SignedInfo element that reference an Object element.	12.2.4.1			×	

Table H–14. Digital signatures optional requirements

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
O6.1	Format designers might allow a package to include digital signatures to enable consumers to validate the integrity of the contents. The producer might include the digital signature when allowed by the format designer.	12		×	×	

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
O6.2	If there are no Digital Signature XML Signature parts in the package, the Digital Signature Origin part is optional.	12.2.1			×	
O6.4	The producer might create zero or more Digital Signature XML Signature parts in a package.	12.2.2			×	
O6.5	Alternatively, the producer might store the certificate as a separate part in the package, might embed it within the Digital Signature XML Signature part itself, or might not include it in the package if certificate data is known or can be obtained from a local or remote certificate store.	12.2.3			×	
O6.6	The producer might sign the part holding the certificate.	12.2.3			×	
O6.7	Producers might share Digital Signature Certificate parts by using the same certificate to create more than one signature.	12.2.3			×	
O6.8	The format designer might permit one or more application-specific Object elements. If allowed by the format designer, format producers can create one or more application-specific Object elements.	12.2.4.14		×	×	
O6.9	Format designers and producers might not apply package-specific restrictions regarding URIs and Transform elements to application-specific Object element.	12.2.4.14		×	×	
O6.10	Format designers might permit producers to sign individual relationships in a package or the Relationships part as a whole.	12.2.4.25		×	×	

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
O6.11	The package implementer might create relationships XML that contains content from several namespaces, along with versioning instructions as defined in Part 5: “Markup Compatibility and Extensibility”.	12.2.4.26	×			
O6.12	Format designers might specify an application-specific package part format that allows for the embedding of versioned or extended content that might not be fully understood by all present and future implementations. Producers might create such embedded versioned or extended content and consumers might encounter such content.	12.6.2		×	×	×

H.7 Pack URI

Table H–15. Pack URI conformance requirements

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M7.1	The authority component contains an embedded URI that points to a package. The package implementer shall create an embedded URI that meets the requirements defined in RFC 3986 for a valid URI.	B.1	×			
M7.2	The optional path component identifies a particular part within the package. The package implementer shall only create path components that conform to the part naming rules. When the path component is missing, the resource identified by the pack URI is the package as a whole.	B.1	×			

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
M7.3	The package implementer shall consider pack URIs equivalent if: The scheme components are octet-by-octet identical after they are both converted to lowercase; *and* The URIs, decoded as described in §B.2 from the authority components are equivalent (the equivalency rules *by* scheme, as per RFC 3986); and The path components are equivalent when compared as case-insensitive ASCII strings.	B.4	×			
M7.4	The package implementer shall not create an authority component with an unescaped colon (:) character.	B.1	×			

Table H–16. Pack URI optional requirements

ID	Rule	Reference	Package Implementer	Format Designer	Format Producer	Format Consumer
O7.1	Consumer applications, based on the obsolete URI specification RFC 2396, might tolerate the presence of an unescaped colon character in an authority component.	B.1				×

End of informative text.

Annex I. Bibliography

The bibliography is informative.

The following documents are useful references for implementers and users of this Open Packaging specification, in addition to the normative references:

ISO/IEC Directives Part 2, Rules for the structure and drafting of International Standards, Fourth edition, 2001, ISBN 92-67-01070-0.

The Unicode Standard, Version 3.0, by the Unicode Consortium; Addison-Wesley Publishing Co, ISBN 0-201-61633-5, February 2000. The latest version can be found at the Unicode Consortium's web site, www.unicode.org, at this writing.

Dublin Core Element Set v1.1. http://purl.org/dc/elements/1.1/

Dublin Core Terms Namespace. http://purl.org/dc/terms/

Extensible Markup Language (XML) 1.0 (Third Edition), W3C Recommendation, 04 February 2004.

Namespaces in XML 1.1, W3C Recommendation, 4 February 2004.

RFC 2616 *Hypertext Transfer Protocol—HTTP/1.1*, The Internet Society, Berners-Lee, T., R. Fielding, H. Frystyk, J. Gettys, P. Leach, L. Masinter, and J. Mogul, 1999, http://www.rfc-editor.org.

RFC 3986 *Uniform Resource Identifier (URI): Generic Syntax*, The Internet Society, Berners-Lee, T., R. Fielding, and L. Masinter, 2005, http://www.rfc-editor.org.

RFC 3987 *Internationalized Resource Identifiers (IRIs)*, The Internet Society, Duerst, M. and M. Suignard, 2005, http://www.rfc-editor.org.

RFC 4234 *Augmented BNF for Syntax Specifications: ABNF*, The Internet Society, Crocker, D., (editor), 2005, http://www.rfc-editor.org.

W3C NOTE 19980827, *Date and Time Formats*, Wicksteed, Charles, and Misha Wolf, 1997, http://www.w3.org/TR/1998/NOTE-datetime-19980827.

XML Base, W3C Recommendation, 27 June 2001.

XML Path Language (XPath), Version 1.0, W3C Recommendation, 16 November 1999.

XML Schema Part 1: Structures, W3C Recommendation, 28 October 2004.

XML Schema Part 2: Datatypes, W3C Recommendation, 28 October 2004.

XML-Signature Syntax and Processing, W3C Recommendation, 12 February 2002.

ZIP File Format Specification, Version 6.2.1, PKWARE Inc., 2005.

End of informative text.

Annex J. Index

This annex is informative.

End of informative text.

to be used to help end-users identify parts of a package or a package as a whole. These images can be generated by the producer and stored as parts.

Office Open XML

Part 3: Primer

December 2006

Table of Contents

Foreword

This multi-part Standard deals with Office Open XML Format-related technology, and consists of the following parts:

- Part 1: "Fundamentals"
- Part 2: "Open Packaging Conventions"
- **Part 3: "Primer"(this document)**
- Part 4: "Markup Language Reference"
- Part 5: "Markup Compatibility and Extensibility"

Introduction

This Part is one piece of a specification that describes a family of XML schemas, collectively called *Office Open XML*, which define the XML vocabularies for word-processing, spreadsheet, and presentation documents, as well as the packaging of documents that conform to these schemas.

The goal is to enable the implementation of the Office Open XML formats by the widest set of tools and platforms, fostering interoperability across office productivity applications and line-of-business systems, as well as to support and strengthen document archival and preservation, all in a way that is fully compatible with the large existing investments in Microsoft Office documents.

1. Scope

This Part contains a detailed introduction to the following Office Open XML topics:

- WordprocessingML
- SpreadsheetML
- PresentationML
- DrawingML
- VML
- Various shared MLs

The organization of this Part is much the same as its corresponding reference Part, Part 4, and is intended as a gentle introduction to Part 4.

2. Introduction to WordprocessingML

This clause is informative.

This clause contains a detailed introduction to the structure of a WordprocessingML document.

2.1 Stories

A WordprocessingML document is composed of a collection of stories. Each *story* represents a distinct region of text within the document. The following kinds of region exist: comment (§2.14.5), endnote (§2.12.2), footer (§2.11.2), footnote (§2.12.1), frame, glossary document (§2.13), header (§2.11.1), main story (§2.2), subdocument (§2.18.2), and text box (§2.18.1).

With one exception (a glossary document), all stories in a document utilize a common set of properties that determine the presentation of the contents of each story. These properties include font information, style definitions, numbering definitions, and document settings.

2.2 Basic Document Structure

The main document story of the simplest WordprocessingML document consists of the following XML elements:

- document — The root element for a WordprocessingML's main document part, which defines the main document story.
- body — The container for the collection of block-level structures that comprise the main story.
- p — A paragraph.
- r — A run.
- t — A range of text.

A *run* is a region of text in a story with a common set of properties. The text in a WordprocessingML document must be contained within one or more runs. A *paragraph* is a collection of one or more runs that is displayed as a unit. A run must be contained within a paragraph.

Consider the following Main Document XML for a simple WordprocessingML document:

```
<?xml version="1.0"?>
<w:document xmlns:w="…">
  <w:body>
    <w:p>
      <w:r>
        <w:t>Hello, world.</w:t>
      </w:r>
    </w:p>
  </w:body>
</w:document>
```

2.3 Main Document Story

The contents of the main document story—the only story that is required in a valid WordprocessingML document—are encapsulated within the body element. The content of the main document body is a collection of block-level structures, which are those WordprocessingML elements that can contain and/or be sibling elements with a WordprocessingML paragraph.

Within the document body, the valid set of block level content is:

- Paragraphs
- Tables
- Custom markup (custom XML, structured document tags)
- Section properties
- Annotations (comments, revision markers, range permission markers)
- Alternate format chunks

Each of these block-level content constructs (the 'building blocks' of WordprocessingML) is defined in the following subclauses.

2.3.1 Document Backgrounds

As well as containing a body, a document element can also contain the definition of the document's background via the background element and its contents. This background applies to all printed pages within this document. A document background in WordprocessingML can have a single color, as well as the application of various drawing effects such as color gradient or pattern, and a tiled or stretched image. All background information in a WordprocessingML document is stored using the Vector Markup Language (VML) syntax. The single exception to this is the background color, which is stored natively in WordprocessingML using the bgColor attribute.

Consider a simple background in WordprocessingML, which consists of a single color with a gradient fill applied:

```
<w:background w:bgColor="5C83B4">
  <v:background id="_x0000_s1025" o:bwmode="white" fillcolor="#5c83b4
    [3204] o:targetscreensize="800,600">
    <v:fill color2="fill darken(118) method="linear sigma" focus="100%"
      type="gradient"/>
  </v:background>
</w:bgPict>
```

The background consists of two components: a background fill color of RGB value 5C83B4, and the background gradient stored as a VML transformation.

2.4 Paragraphs and Rich Formatting

2.4.1 Paragraphs

The most basic unit of block-level content within a WordprocessingML document, *paragraphs* are stored using the p element. A *paragraph* defines a distinct division of content that begins on a new line. A paragraph can contain three pieces of information: optional paragraph properties, inline content (typically runs), and a set of optional revision IDs used to compare the content of two documents.

Consider the paragraph fragment "*The quick brown fox jumped …*" which is centered on a paragraph. As all the text in the paragraph is emphasized using italics, in the XML, the contents of the paragraph will have the justify-center property, and each run within the paragraph (as well as the run properties for the paragraph mark) stores the italics property; for example:

```
<w:p>
  <w:pPr>
    <w:jc w:val="center"/>
    <w:rPr>
      <w:i/>
    </w:rPr>
  </w:pPr>
  <w:r>
    <w:rPr>
      <w:i/>
    </w:rPr>
    <w:t>The quick brown fox jumped…</w:t>
  </w:r>
</w:p>
```

Notice that each run specifies the character formatting information for its contents, and the paragraph specifies the paragraph level formatting (the center-justification). It is also notable that since leading and trailing whitespace is not normally significant in XML, some runs require a designating specifying that their whitespace is significant via the xml:space element.

A paragraph's properties are specified via the pPr element. Some examples of paragraph properties are alignment, border, hyphenation override, indentation, line spacing, shading, text direction, and widow/orphan control.

It should also be noted that a pPr element may contain a set of run properties within a rPr element – these properties are applied to the run which contains the glyph which represents the paragraph mark and not the entire paragraph.

2.4.2 Runs

The next level of the document hierarchy is the *run*, which defines a region of text with a common set of properties, represented by the r element. An r element allows the producer to combine breaks, styles, or formatting properties, applying the same information to all the parts of the run.

Just as a paragraph can have properties, so too can a run. All of the elements inside an r element have their properties controlled by a corresponding optional rPr run properties element, which must be the first child of the r element. In turn, the rPr element is a container for a set of property elements that are applied to the rest of the children of the r element. The elements inside the rPr container element allow the consumer to control whether the text in the following t elements is bold, underlined, or visible, for example. Some examples of run properties are bold, border, character style, color, font, font size, italic, kerning, disable spelling/grammar check, shading, small caps, strikethrough, text direction, and underline.

Consider the following run within a WordprocessingML document:

```
<w:r>
  <w:rPr>
    <w:b/>
    <w:i/>
  </w:rPr>
  <w:t>quick</w:t>
</w:r>
```

The run specifies two formatting properties in its run contents: bold and italic. These properties are therefore applied to all content within this run.

A producer can break a run into an arbitrary number of smaller runs, provided each smaller run uses the same set of properties, without changing the content of the document.

Consider the content "only one word is emphasized" in a WordprocessingML document. An efficient producer could choose to output this content using two runs, as follows:

```
<w:r>
  <w:t xml:space="preserve">only one word is </w:t>
</w:r>
```

```
<w:r>
  <w:rPr>
    <w:i/>
  <w:rPr>
  <w:t>emphasized</w:t>
</w:r>
```

However, a less efficient producer might use four runs, as follows:

```
<w:r>
  <w:t>only one</w:t>
</w:r>
<w:r>
  <w:t xml:space="preserve"> word is </w:t>
</w:r>
<w:r>
  <w:rPr>
    <w:i/>
  <w:rPr>
  <w:t>empha</w:t>
</w:r>
<w:r>
  <w:rPr>
    <w:i/>
  <w:rPr>
  <w:t>sized</w:t>
</w:r>
```

Although the latter example uses four runs rather than two, the net run information applied to each region of text is identical, and both are equally valid.

Of course, a run might need to be broken. For example, the properties of only some the text in that run are changed, requiring the changed part to be put into its own run. Another example involves the insertion of some sort of marker into the middle of an existing run. That requires the run be broken into two with the marker inserted between them.

The following run contains two sentences:

```
<w:r>
  <w:t>Hello, world. How are you, today?</w:t>
</w:r>
```

If the first two words are bolded in these sentences, the run will need to be broken into two runs in order to store the formatting, as follows:

```
<w:r>
  <w:rPr>
    <w:b/>
  </w:rPr>
  <w:t xml:space="preserve">Hello, world. </w:t>
</w:r>
<w:r>
  <w:t>How are you, today?</w:t>
</w:r>
```

Apart from text, a run can also contain numerous kinds of textual content (§2.4.3) A run can also contain a set of revision IDs used for document "merge and compare".

2.4.3 Run Content

The lowest level of this hierarchy is *run content*, that content that can be stored within a single run in a document. In WordprocessingML, the types of run content include:

- Text
- Deleted text
- Soft line breaks
- Field codes
- Deleted field codes
- Footnote/endnote reference marks
- Simple fields
- Page numbers
- Tabs
- Ruby text
- DrawingML content
- Embedded objects
- Pictures

2.4.3.1 Text

The most common run content is the t element, which is the container for the text that makes up the document's content. A t element can contain an arbitrary amount of text, up to and including the entire document's contents. However, typically, long runs of text are broken up into paragraphs and strings of text having different formats, or are interrupted by line breaks, graphics, tables, and other items. A t element must be enclosed within an r element; i.e., a run of text. An r element can contain multiple t elements, interspersed among other elements.

Aside from the t element, there are three types of text in WordprocessingML:

- delText - Deleted text
- instrText - Field codes

- delInstrText - Deleted field codes

These four types of text are defined using unique elements in WordprocessingML so that simple consumers can determine the text of the document simply by grabbing the contents of the t node, without needing to check where revisions start and end, etc. to determine the state of the text contents.

It is also notable that these are the only elements in a WordprocessingML document's main document part that can contain a XML text node.

2.4.4 Formatting Property Values

Most of the children of an rPr or pPr element have a single val attribute that is limited to a specific set of values. For example, the b (bold) element causes the text that follows it to be bold when the b element has a val attribute with value on. If the val attribute isn't present for the b element, it defaults to "on". Therefore, <w:b/> is equivalent to <w:b w:val="on"/>.

Aside from the default values, which are documented with each element, this is particularly important when specifying the difference between omitting a formatting property and explicitly turning it off.

For example, consider the following run:

```
<w:r>
  <w:rPr>
    <w:b w:val="off"/>
  </w:rPr>
  <w:t xml:space="preserve">Hello, world. </w:t>
</w:r>
```

This run explicitly declares that the bold property is turned off for this text, as opposed to the following run:

```
<w:r>
  <w:t xml:space="preserve">Hello, world. </w:t>
</w:r>
```

This run says nothing about the bold property. This distinction is particularly important when dealing with content that is formatting using styles - if the content was not contained in a styled paragraph, both would be identical. However, in the case where the paragraph is styled, the former would never be bold regardless of the style information, whereas the latter would express the bold property as set by the style, since it's omission of the bold property means "whatever the underlying formatting is".

Some elements have val attributes that offer a richer set of choices than on and off; the u (underline) element is one such element. In this case, the same rules apply, the omission of the property simply means use the underlying properties.

2.5 Tables

Another type of block-level content in WordprocessingML, A *table* is a set of paragraphs (and other block-level content) arranged in *rows* and *columns*.

2.5.1 Introduction

Tables in WordprocessingML are defined via the tbl element, which is analogous to the HTML `<table>` tag. The table element specifies the location of a table present in the document.

A tbl element has two elements that define its properties: tblPr, which defines the set of table-wide properties (such as style and width), and tblGrid, which defines the grid layout of the table. A tbl element can also contain an arbitrary non-zero number of rows, where each row is specified with a tr element. Each tr element can contain an arbitrary non-zero number of cells, where each cell is specified with a tc element.

Consider an empty one-cell table (i.e.,; a table with one row, one column) and 1 point borders on all sides:

This table is represented by the following WordprocessingML:

```
<w:tbl>
  <w:tblPr>
    <w:tblW w:w="5000" w:type="pct"/>
    <w:tblBorders>
      <w:top w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:left w:val="single" w:sz="4 w:space="0" w:color="auto"/>
      <w:bottom w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:right w:val="single" w:sz="4" w:space="0" w:color="auto"/>
    </w:tblBorders>
  </w:tblPr>
  <w:tblGrid>
    <w:gridCol w:w="10296"/>
  </w:tblGrid>
  <w:tr>
    <w:tc>
      <w:tcPr>
        <w:tcW w:w="0" w:type="auto"/>
      </w:tcPr>
      <w:p/>
    </w:tc>
  </w:tr>
</w:tbl>
```

This table specifies table-wide properties of 100% of page width (tblW's type attribute specifies how the width value in the w attribute shall be interpreted—pct specifies a measurement of fiftieths of a percent) and the set of table borders (tblBorders), the table grid which defines a set of shared vertical edges within the table (discussed later), and a single row.

2.5.2 Table Properties

The tblPr element defines table-wide properties, properties which are applied to each row and cell in the table. The complete set of table-wide properties can be found on the definition for the tblPr element.

Consider the following simple WordprocessingML table:

This table defines outside and inside table borders, etc; and is set to 100% of page width - both table-wide properties. The resulting table is represented by the following WordprocessingML:

```
<w:tbl>
  <w:tblPr>
    <w:tblW w:w="0" w:type="auto"/>
    <w:tblBorders>
      <w:top w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:left w:val="single" w:sz="4 w:space="0" w:color="auto"/>
      <w:bottom w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:right w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:insideH w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:insideV w:val="single" w:sz="4" w:space="0" w:color="auto"/>
    </w:tblBorders>
  </w:tblPr>
  <w:tblGrid>
    ...
  </w:tblGrid>
  <w:tr>
    ...
  </w:tr>
</w:tbl>
```

In this example, the tblW element defines the total width of the table, which, in this case, is set to a type of auto, which specifies that the table should be sized to fit its contents. The tblBorders element specifies each of the table's borders, and specifies a one point border on the top, left, bottom, right and inside horizontal and vertical border. The table-wide properties can be overwritten on an individual row basis by specifying table property overrides within the table row properties.

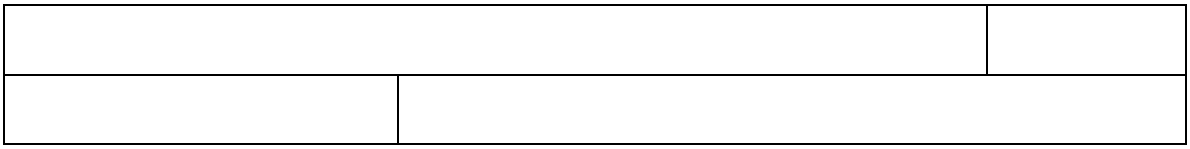
2.5.3 Table Grid

The tblGrid element defines the *grid* for the table. All columns in the table (including the space before and after a row) reference this grid. Each gridCol defines a single grid column within the table's layout, which is used to define the presence of a vertical line within the table. A tblGrid element can contain an arbitrary number of gridCol elements, where each gridCol element represents one grid column in the table and defines a single grid entry. When cells are laid out within this table, as discussed below, all cells will be forced to snap the shared column edges defined by this grid.

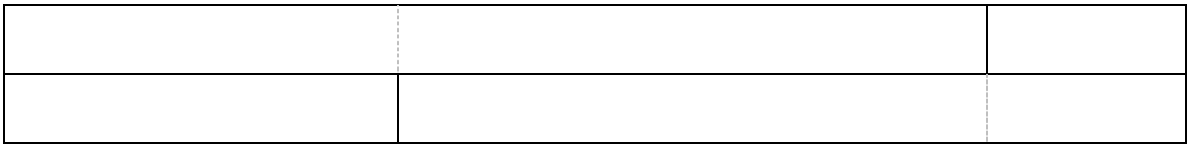
Returning to the earlier 'one-cell empty table' example, the table has one column with a width of 10,296 twentieths of a point. This measurement (twentieths of a point, or twips) is frequently used in WordprocessingML, and translates to 1/1440th of an inch (one-twentieth of a point, which is itself 1/72nd of an inch):.

```
<w:tblGrid>
  <w:gridCol w:w="10296"/>
</w:tblGrid>
```

Consider the following, more complex table that has two rows and two columns; the columns are not aligned:



This table is represented by laying out the cells on a table grid consisting of three table grid columns, each grid column representing a logical vertical column in the table:



The dashed lines represent the virtual vertical continuations of each table grid column, and the resulting table grid is represented as the following in WordprocessingML:

```
<w:tblGrid>
  <w:gridCol w:w="2952"/>
  <w:gridCol w:w="4416"/>
  <w:gridCol w:w="1488"/>
</w:tblGrid>
```

```
<w:tr>
  <w:tc>
    <w:tcPr>
      <w:tcW w:w="7368" w:type="dxa"/>
      <w:gridSpan w:val=”2”/>
    </w:tcPr>
    <w:p/>
  </w:tc>
  <w:tc>
    <w:tcPr>
      <w:tcW w:w="1488" w:type="dxa"/>
    </w:tcPr>
    <w:p/>
  </w:tc>
</w:tr>
<w:tr>
  <w:tc>
     <w:tcPr>
      <w:tcW w:w="2952" w:type="dxa"/>
    </w:tcPr>
    <w:p/>
  </w:tc>
  <w:tc>
    <w:tcPr>
      <w:tcW w:w="5904" w:type="dxa"/>
      <w:gridSpan w:val=”2”/>
    </w:tcPr>
    <w:p/>
  </w:tc>
</w:tr>
```

Notice that each of the cells which do not span one grid column (i.e., span two adjacent vertical lines) must specify this fact by supplying a gridSpan element with a value which determines how many grid columns this cell will span. Each gridCol element represents a shared 'column' in a table (to which the cells will snap) even if it doesn't appear visually.

2.5.4 Table Rows and Cells

A table row is defined using a tr element, which is analogous to the HTML <tr> tag. The tr element acts as a container for a row of cells with the table's content.

A tr element has one formatting child element, trPr, which defines the row properties (such as the row's width) and whether it can split across a page. Each property is defined by an individual child element under the trPr element. The complete set of table row properties can be found on the definition for the trPr element. As

well, a table row can contain two types of content: custom markup (custom XML or structured document tags), and table cells.

The cells in a row contain the table's content and are defined by tc elements, which are analogous to HTML <td> tags.

A tc element has one formatting child element, tcPr, which defines the properties for the cell. Each unique property is specified by a child element of this element. The complete set of table cell properties can be found on the definition for the tcPr element. As well, a table cell can contain any valid block-level content, which allows for the nesting of paragraphs and tables within table cells.

In the example below, the tcW element defines the width of the cell, where the attribute w is the value in twips. Here the width of the cell is 8,856 units, where *units* are defined by the attribute type. In this case, dxa represents twips.

```
<w:tr>
  <w:tc>
    <w:tcPr>
      <w:tcW w:w="8856" w:type="dxa"/>
    </w:tcPr>
    <w:p/>
  </w:tc>
</w:tr>
```

The tc element contains the cell's content, which, in this case, is an empty p element.

Consider a table having one cell, which contains the text "Hello, world":

Hello, world

This table's content is represented by the following XML:

```
<w:tr>
  <w:tc>
    <w:tcPr>
      <w:tcW w:w="1770" w:type="dxa"/>
    </w:tcPr>
    <w:p>
      <w:r>
        <w:t>Hello, World</w:t>
      </w:r>
    </w:p>
  </w:tc>
</w:tr>
```

At both the row and cell levels, the properties must also specify how the rows and cells will be placed on the table grid.

The trPr element contains information about the number of grid units which should be omitted ('skipped') before and after the row is complete using the gridBefore and gridAfter elements, allowing rows to start at different columns on the grid, as well as a preferred width for that leading/trailing space using the wBefore and wAfter elements. The tcPr element also contains grid information pertaining to how many grids a cell spans using the gridSpan element, which determines how many grid units are consumed by the current cell, as well as a preferred width for that cell using the tcW element.

In the earlier complex table having two rows of two differently sized cells, a consumer shall represent that table containing three grid columns (one per distinct vertical line). Consider the following XML for the first row of that table:

```
<w:tr>
  …
  <w:tc>
    <w:tcPr>
      <w:tcW w:w="5145" w:type="dxa" />
      <w:gridSpan w:val="2" />
    </w:tcPr>
    <w:p />
  </w:tc>
  <w:tc>
    <w:tcPr>
      <w:tcW w:w="2145" w:type="dxa" />
    </w:tcPr>
    <w:p/>
  </w:tc>
</w:tr>
```

Again, the gridSpan element is the number of grid columns that cell spans when being laid out on the table grid. In this example, the first cell of the first row contains two grid columns. As well, the cell specifies its preferred width using the tcW element, which tells the consumer the width desired by that cell at layout time.

It is important to note that every width in a table is a preferred width - because the table must satisfy the grid at all times, conflicting table properties must be resolved by overriding preferred widths in a specific manner, shown below.

2.5.5 Table Layout

Given the information shown in the table shown above, the table is specified as a series of properties:

- Table-level properties (e.g., preferred width)
- Table column grid

- Row-level properties (e.g., grid units before/after row start/end)
- Cell-level properties (e.g., number of grid units spanned)

In order to manipulate this set of properties into a table, the following logics are used, depending on the type of table:

2.5.6 Fixed Width Tables

The first type of table is a fixed width table, a table that does not dynamically resize based on its contents. In a fixed width table, the table information is used in the following manner:

- The table grid is used to create the set of shared columns in the table and their initial widths as defined in the tblGrid element
- The table's total width is defined based on the tblW property – if it is set to `auto` or `nil`, then the width is not yet determined and will be specified using the row and cell information.
- The first table row is read and the initial number of grid units before the row starts is skipped. The width of the skipped grid columns is set using the wBefore property.
- The first cell is placed on the grid, and the width of the specified grid column span set by gridSpan is set based on the tcW property.
- Each additional cell is placed on the grid.
- If at any stage, the preferred width requested for the cells exceeds the preferred width of the table, then each grid column is proportionally reduced in size to fit the table width.
- If the grid is exceeded (e.g., tblGrid specifies three grid columns, but the second cell has a gridSpan of three), the grid is dynamically increased with a default width for the new grid column.
- For each subsequent row, cells are placed on the grid, and each grid column is adjusted to be the maximum value of the requested widths (if the widths do not agree) by adding width to the last cell that ends with that grid column. Again, if at any point, the space requested for the cells exceeds the width of the table, then each grid column is proportionally reduced in size to fit the table width.

2.5.7 AutoFit Tables

In an AutoFit table (one which specifies that it should "AutoFit to table contents"), the table information is used in the following manner:

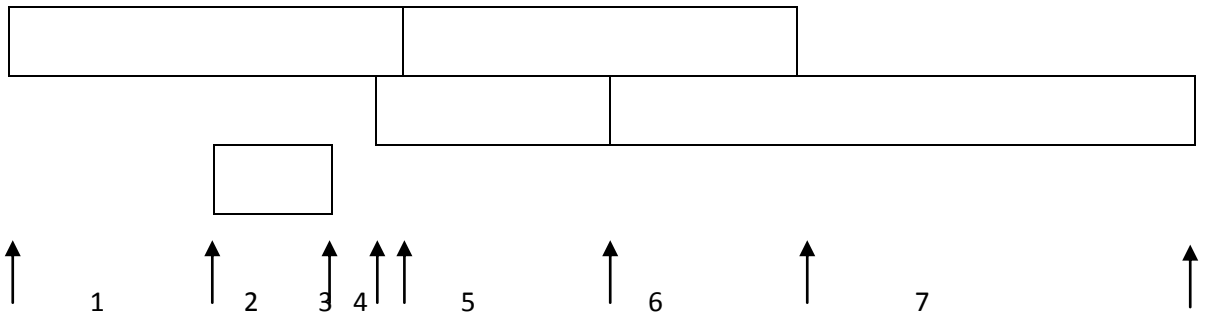
- Perform the steps above to lay out the fixed width version of the table.
- Calculate the minimum content width - the width of the cell's contents including all possible line breaking locations (or the cell's width, if the width of the content is smaller), and the maximum content width -the width of the cell's contents (assuming no line breaking not generated by explicit line breaks).
- The minimum and maximum content width of all cells that span a single grid column is the minimum and maximum content width of that column.
- For cells which span multiple grid columns, enlarge all cells which it spans as needed to meet that cell's minimum width.

- If any cell in a grid column has a preferred width, the first such width overrides the maximum width of the column's contents.
- Place the text in the cells in the table, respecting the minimum content width of each cell's content. If a cell's minimum content width exceeds the cell's current width, preferences are overridden as follows:
- First, override the column widths by making all other grid columns proportionally smaller until each it at its minimum width. This cell may then grow to any width between its own minimum and maximum width.
- Next, override the preferred table width until the table reaches the page width.
- Finally, force a line break in each cell's contents as needed

2.5.8 Complex Table Example

The properties above are best illustrated by example:

As shown above, table cells can be merged horizontally. This is represented with a single table cell whose gridSpan property defines the number of grid units consumed by that table cell for the current row. Consider the following fixed width table, which makes extensive use of resized and merged cells on what is actually just a seven-column grid. (The arrows point to each (invisible) vertical line of the grid and the numbers refer to the grid columns):



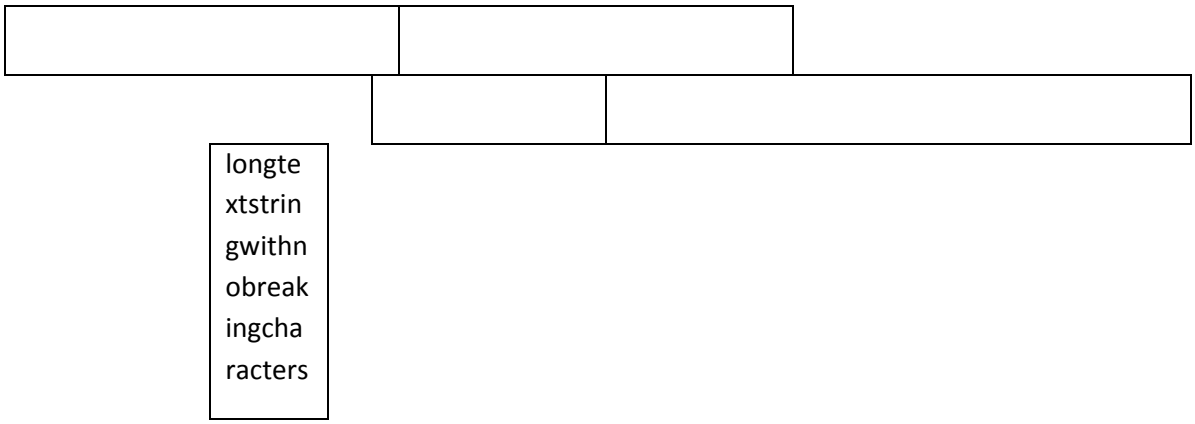
Although the table is visually complex, the standard rules apply: the first cell in the table is simply a cell which spans four grid units horizontally, as specified in the gridSpan element, and whose preferred with is 2952 twips, specified in the tcW element:

```
<w:tc>
  <w:tcPr>
    <w:tcW w:w="2952" w:type="dxa"/>
    <w:gridSpan w:val="4"/>
  </w:tcPr>
  <w:p/>
</w:tc>
```

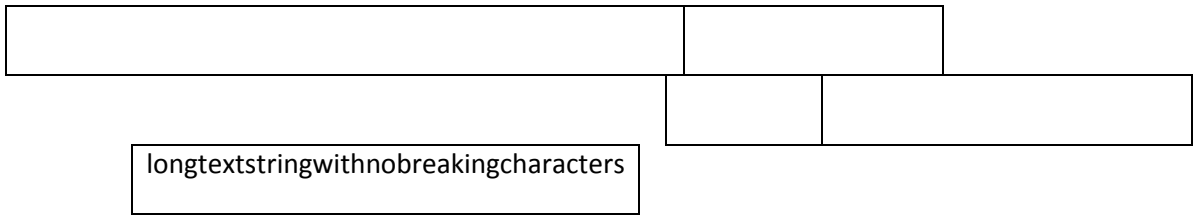
Similarly, all cells indented from the stand and end of the grid specify that indent using the gridBefore and gridAfter elements. For example, the XML for the second row in the table shows that that row starts three grid units into the table:

```
<w:tr>
  <w:trPr>
    <w:gridBefore w:val="3"/>
    <w:wBefore w:w="2748" w:type="dxa"/>
  </w:trPr>
  …
</w:tr>
```

If we take this fixed width table and introduce a long string into the single cell in row 3, we see that the presence of this text does not affect cell widths:



If we now turn on the AutoFit property and type into the cell in row three, which spans only grid column two, we see that the algorithm for this AutoFit table causes all cells in grid column two to increase in size, proportionally decreasing the other grid columns' size to accommodate the long non-breaking string in the last cell:



Each of the other grid columns was reduced, but since all columns are not at their minimum size, the table width is not increased even though the table is not yet at the page width.

2.5.9 Vertically Merged Cells

Although the previous examples may have implied that tables have strict definition of rows, table cells can also be merged vertically. The tcPr element may contain the vmerge element that defines the extent of vertically merged grid columns within a table. A vmerge element with its val attribute set to `restart` marks the start of a vertically merged cell range. A vmerge element with the val attribute set to `continue` (the default value)

marks the continuation of a vertically merged grid column. Cells between the first and last merged cell that are part of the vertical merge each must have a vmerge element to continue the vertical merge.

For example, consider a table with two rows and two columns:

First cell, first row	Last cell, first row
First cell, second row	Last cell, second row

Merging the two rows in the second column will result in the following table:

First cell, first row	Last cell, first row Last cell, second row
First cell, second row	

The last cell in the first row starts a merge that is completed in the cell below it, resulting in the following WordprocessingML:

```
<w:tr>
  <w:tc>
    <w:p>
      <w:r>
        <w:t>First cell, first row</w:t>
      </w:r>
    </w:p>
  </w:tc>
  <w:tc>
    <w:tcPr>
      <w:vmerge w:val="restart"/>
    </w:tcPr>
    <w:p>
      <w:r>
        <w:t>Last cell, first row</w:t>
      </w:r>
    </w:p>
    <w:p>
      <w:r>
        <w:t>Last cell, second row</w:t>
      </w:r>
    </w:p>
  </w:tc>
</w:tr>
```

```
<w:tr>
  <w:tc>
    <w:p>
      <w:r>
        <w:t>First cell, second row</w:t>
      </w:r>
    </w:p>
  </w:tc>
  <w:tc>
    <w:tcPr>
      <w:vmerge/>
    </w:tcPr>
    <w:p/>
  </w:tc>
</w:tr>
```

As shown, the vmerge with a value of `restart` begins (or restarts) a merged region, and the cell with no value is merged with the one above.

2.6 Custom Markup

Within a WordprocessingML document, it is often necessary for specific documents to contain semantic information beyond the presentation information specified by this Office Open XML specification. For example, an invoice document may wish to specify that a particular sentence of text is a customer name, in order for that information to be easily extracted from the document without the need to parse the text using regular expression matching or similar. For those cases, multiple facilities are provided for the insertion and round-tripping of customer defined semantics within a WordprocessingML document.

There are three distinct forms in which customer-defined semantics can be inserted into a WordprocessingML document, each with their own specific intended usage:

- Smart tags
- Custom XML markup
- Structured document tags (content controls)

The usage and presentation of each of these forms is described in the following sections.

2.6.1 Smart Tags

The first example of customer-defined semantics which can be embedded in a WordprocessingML document are smart tags. Smart tags allow semantic information to be added around an arbitrary run or set of runs within a document to provide information about the type of data contained within.

Consider the following text in a WordprocessingML document, with a smart tag around the stock symbol 'CNTS' (where the smart tag is displayed using a purple dotted underline):

This is a stock symbol: MSFT

This text would translate to the following WordprocessingML markup:

```
<w:p w:rsidR="00672474" w:rsidRDefault="00672474">
  <w:r>
    <w:t xml:space="preserve">This is a stock symbol: </w:t>
  </w:r>
  <w:smartTag w:uri="http://schemas.openxmlformats.org/2006/smarttags"
     w:element="stockticker">
    <w:r>
      <w:t>MSFT</w:t>
    </w:r>
  </w:smartTag>
</w:p>
```

As shown above, the smart tag is delimited by the smartTag element, which surrounds the run (or runs) which contain the text which is part of the smart tag.

The smart tag itself carries two required pieces of information, which together contain the customer semantics for this smart tag.

The first of these is the namespace for this smart tag (contained in the uri attribute). This allows the smart tag to specify a URI which should be round-tripped with this smart tag and be available to a consumer. It is intended to be used to specify a family of smart tags to which this one belongs – for example, in the sample above, the smart tag belongs to the `http://schemas.openxmlformats.org/2006/smarttags` namespace.

The second of these is the element name for this smart tag (contained in the element attribute). This allows the smart tag to specify a name which should be round-tripped with this smart tag and again available to a consumer. It is intended to be used to specify a unique name for this type of smart tag – for example, in the sample above, the smart tag specifies that its data is of type `stockticker`.

As well as the required information specified above, a smart tag can also contain any number of additional properties in namespace/name/value sets by adding them to the smart tag's property bag.

Using the example above, adding a new property called `fullCompanyName` with no namespace and value `Microsoft Corporation` to the smart tag would mean augmenting the output to add the smartTagPr element with this new property as follows:

```
    <w:smartTag w:uri="http://schemas.openxmlformats.org/2006/smarttags"
       w:element="stockticker">
    <w:smartTagPr>
      <w:attr w:name="fullCompanyName" w:val="Microsoft Corporation"/>
    </w:smartTagPr>
    <w:r>
      <w:t>MSFT</w:t>
    </w:r>
  </w:smartTag>
```

The resulting XML, as seen above, simply adds an attr element which specifies the property and value for the property bag.

A producer can embed a smart tag around any run-level content in a WordprocessingML document in order to embed additional information about the family and type of the data contained within. This allows 'tagging' of specific regions of a document with these semantics without need to provide context beyond the information provided in the uri and element attributes.

A consumer can read this smart tag data and provide additional functionality around these namespace/element pairs, which may or may not be specific to that smart tag type in the document. Examples of this functionality include: the ability to add/remove this markup via a user interface, ability to provide actions to operating in the context of this data type, etc.

2.6.2 Custom XML Markup

The next example of customer-defined semantics which can be embedded in a WordprocessingML document is custom XML markup. Custom XML markup allows the application of the XML elements defined in any valid XML Schema file to be applied to the contents of a WordprocessingML document in one of two locations: around a paragraph or set of paragraphs (at the block level); or around an arbitrary run or set of runs within a document (at the inline level) to provide semantics to that content within the context and structures defined by the associated XML Schema definition file.

The distinction between custom XML markup and smart tags is based on the fact that custom XML markup corresponds with the contents of a custom XML schema; which means that as shown below, custom XML markup can be used at the block-level to mark up the contents of a document on levels beyond that of one or more runs as well as on the inline (run) level. It can also be validated against a custom XML schema by a producer at run time.

Consider a simple XML Schema which defines two elements: a root element of invoice, and a child element of customerName - the first defining that this file's contents are an invoice, and the second specifying that the enclosed text as a customer's name:

invoice This is an invoice.

And this is a customer name: customerName Tristan Davis invoice

This output would translate to the following WordprocessingML markup:

```
<w:customXml w:uri="http://www.contoso.com/2006/invoice" w:element="invoice">
  <w:p>
    <w:r>
      <w:t>This is an invoice.</w:t>
    </w:r>
  </w:p>
  <w:p>
    <w:r>
      <w:t xml:space="preserve">And this is a customer name: </w:t>
    </w:r>
    <w:customXml w:uri="http://www.contoso.com/2006/invoice"
w:element="customerName">
      <w:r>
        <w:t>Tristan Davis</w:t>
      </w:r>
    </w:customXml>
  </w:p>
</w:customXml>
```

As shown above, each of the XML elements from the customer-supplied XML schema is represented within the document output as a customXml element.

Similar to the smart tag example above, a custom XML element in a document has two required attributes.

The first is the uri attribute, whose contents specify the namespace of the custom XML element in the document. In the example above, the elements each belong to the `http://www.contoso.com/2006/invoice namespace`.

The second is the element attribute, whose contents specify the name of the custom XML element at this location in the document. In the example above, the root element is called invoice and the child element is called customerName.

As well as the required information specified above, custom XML elements can also specify any number of attributes (as specified in the associated XML Schema) on the element. To add this information, the customXmlPr (properties on the custom XML element) specify one or more attr elements.

Using the example above, we can add a type attribute to the customerName element as follows:

```
<w:customXml w:uri="http://www.contoso.com/2006/invoice"
w:element="customerName">
  <w:customXmlPr>
    <w:attr w:uri="http://www.contoso.com/2006/invoice" w:name="type"
w:val="individual"/>
  </w:customXmlPr>
  <w:r>
    <w:t>Tristan Davis</w:t>
  </w:r>
</w:customXml>
```

The resulting XML, as seen above, simply adds an attr element which specifies the attribute for the custom XML element.

A producer can embed a custom XML element around or with block-level or run-level content in a WordprocessingML document in order to embed the structure of the customer-defined XML Schema within the WordprocessingML content. This allows 'tagging' of specific regions of a document with the semantics from this schema, while ensuring that the resulting file can be validated to the WordprocessingML schemas.

A consumer can read this custom XML markup and provide additional functionality around this customer-defined XML markup, which may or may not be specific to that type of XML in the document. Examples of this functionality include: the ability to add/remove this XML markup via a user interface, ability to provide actions to operating in the context of this data type, etc.

Each custom XML element is analogous to an XML element in the specified XML schema, and can be nested arbitrarily to any depth in the document. This facility is limited only by the XML Schema file itself, and the contents of the current document.

2.6.3 Structured Document Tags

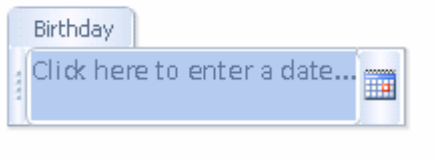
The final example of customer-defined semantics which can be embedded in a WordprocessingML document is the structured document tag (SDT).

As shown above, smart tags and custom XML markup each provide a facility for embedding customer-defined semantics into the document: smart tags, via the ability to provide a basic namespace/name for a run or set of runs within a documents; and custom XML markup, via the ability to tag the document with XML elements and attributes specified by any valid XML Schema file.

However, each of these techniques, while they each provide a way to add the desired semantic information, does not provide a way to affect the presentation or interaction within the document. To bridge these two worlds, structured document tags allow both the specification of customer semantics as well as the ability to influence the presentation of that data in the document.

This means that the customer can define the semantics and context of the tag, but can then use a rich set of pre-defined properties to define its behavior and appearance within the WordprocessingML document's presentation.

Consider a region which should be tagged with the semantic of "birthday", for the user to enter their date or birth into the document. Ideally, this region would also utilize a date picker to allow the user to enter the date from a calendar::



This content would translate to the following WordprocessingML markup:

```
<w:sdt>
  <w:sdtPr>
    <w:alias w:val="Birthday"/>
    <w:id w:val="8775518"/>
    <w:placeholder>
      <w:docPart w:val="DefaultPlaceholder_22479095"/>
    </w:placeholder>
    <w:showingPlcHdr/>
    <w:date>
      <w:dateFormat w:val="M/d/yyyy"/>
      <w:lid w:val="EN-US"/>
    </w:date>
  </w:sdtPr>
  <w:sdtContent>
    <w:p>
      <w:r>
        <w:rPr>
          <w:rStyle w:val="PlaceholderText"/>
        </w:rPr>
        <w:t>Click here to enter a date...</w:t>
      </w:r>
    </w:p>
  </w:sdtContent>
</w:sdt>
```

As shown above, each of the structured document tags in the WordprocessingML file is represented using the sdt element.

Within a structured document tag, there are two child elements which contain the definition and the content of this SDT. The first of these is the sdtPr element, which contains the set of properties specified for this structured document tag. The second is the sdtContent element, which contains all the content which is contained within this structured document tag.

2.6.3.1 Structured Document Tag Properties

Within the SDT's properties, various properties can be set which affect the appearance and behavior of this content in the document. These properties can be divided into four groups:

- Shared properties
- Locking properties
- Structured document tag type
- Type-specific properties

The complete set of properties for a structured document tag are found on the sdtPr element.

The first group is properties shared by all types of SDTs. These include, but are not limited to, the semantic name for the SDT, a unique ID (as an integer) that is round-tripped and allows the control to be uniquely identified across sessions, and a reference to a document building block that should be displayed as placeholder text.

The next group is the locking properties for the tag – these specify whether any consumer should allow the contents of the SDT to be edited, or the SDT itself to be deleted from the document.

The next group, the structured document tag's type, specifies how the content should be expressed in a document. These types include: plain text (all contents are of one formatting), rich text, date picker, combo box, drop-down list, and image. Each of the types provides user interface restrictions that restrict the contents to only those specified by the type (e.g., the picture cannot contain text).

Finally, the type-specific properties contain properties that are sensible in the context of that type. For example, the date format for a date picker or the drop-down list entries for a drop-down list/combo box. Type-specific properties are stored as children of the type's element.

Referring to the example above, the date properties are stored underneath the date element, as follows:

```
<w:sdtPr>
  ...
  <w:date>
    <w:dateFormat w:val="M/d/yyyy"/>
    <w:lid w:val="EN-US"/>
  </w:date>
</w:sdtPr>
```

This ensures that these properties are only available in the appropriate context(s).

2.6.3.2 Structured Document Tag Content

The second child of the sdt element is the sdtContent element, which contains all the content which is contained within this structured document tag.

2.6.3.3 XML Mapping

An additional property for SDTs allows their contents to be stored in another part (in particular, in the custom XML data storage within the file). The presence of the dataBinding element specifies that the contents of this SDT are simply a cache of the data stored at a particular XML element in a particular custom XML data storage part.

2.7 Sections

Within the main document story, there is also often a need for groupings of content on a basis larger than a paragraph (for example, ensuring that a specific set of paragraphs and tables are printed in landscape view, while ensuring that the remainder of the document is printed in portrait view). In order to group this content, a document can be divided into multiple *sections*, each of which defines a region of content in the document and allows the application of a set of section-level properties.

Consider a WordprocessingML document with two paragraphs of content, the first of which should be displayed on a page printed in portrait view, and the second of which should be displayed on a page printed in landscape view (the page content should be rotated 90 degrees to the left on the underlying page).

In order to have each of these paragraphs on different pages having different page orientation characteristics, this document would be split into two sections. Looking at the WordprocessingML for the example above:

```
<w:body>
  <w:p>
    <w:pPr>
      <w:sectPr>
        <w:pgSz w:w="12240" w:h="15840"/>
        <w:pgMar w:top="1440" w:right="1800" w:bottom="1440"
          w:left="1800" w:header="720" w:footer="720" w:gutter="0"/>
        <w:cols w:space="720"/>
        <w:docGrid w:linePitch="360"/>
      </w:sectPr>
    </w:pPr>
    <w:r>
      <w:t>This is sentence one.</w:t>
    </w:r>
  </w:p>
```

```
    <w:p>
      <w:r>
        <w:t>This is sentence two.</w:t>
      </w:r>
    </w:p>
    <w:sectPr>
      <w:pgSz w:w="15840" w:h="12240" w:orient="landscape"/>
      <w:pgMar w:top="1800" w:right="1440" w:bottom="1800"
        w:left="1440" w:header="720" w:footer="720" w:gutter="0"/>
      <w:cols w:space="720"/>
      <w:docGrid w:linePitch="360"/>
    </w:sectPr>
  </w:body>
```

This syntax defines two sections using two distinct sectPr elements: the first has a page size of 12,240 twentieths of a point wide and 15,640 twentieths of a point tall; the second has a page size of 15,640 twentieths of a point wide and 12,240 twentieths of a point tall, and is oriented in landscape mode.

2.7.1 Section Properties

As shown above, the end of a section is defined as a set of properties applied to the last paragraph in that section—converting that paragraph mark into a *section break* (i.e., a paragraph that closes a section).

Those properties are contained in a sectPr element, which is located within the paragraph properties (the pPr element) for the final paragraph in that section. Within the definition of section properties, the properties to be applied to that section (including, but not limited to, page size and orientation, line numbering settings, margins, and columns) are specified. The complete set of section properties is located on the definition for the sectPr element.

The only exception to this rule is the final set of section properties in this document. These are stored as the last child of the body element. This is done because the document's last paragraph must specify paragraph properties, and this syntax enforces that the final set of section properties are specified.

Going back to our example, the first section break is defined within the last paragraph for that section, but the last section properties are stored after the final paragraph.

```
<w:body>
  <w:p
    <w:pPr>
      <w:sectPr>
        <w:pgSz w:w="12240" w:h="15840"/>
        <w:pgMar w:top="1440" w:right="1800" w:bottom="1440"
          w:left="1800" w:header="720" w:footer="720" w:gutter="0"/>
        <w:cols w:space="720"/>
        <w:docGrid w:linePitch="360"/>
      </w:sectPr>
    </w:pPr>
    <w:r>
      <w:t>This is sentence one.</w:t>
    </w:r>
  </w:p>
  <w:p>
    <w:r>
      <w:t>This is sentence two.</w:t>
    </w:r>
  </w:p>
  <w:sectPr>
    <w:pgSz w:w="15840" w:h="12240" w:orient="landscape"/>
    <w:pgMar w:top="1800" w:right="1440" w:bottom="1800"
      w:left="1440" w:header="720" w:footer="720" w:gutter="0"/>
    <w:cols w:space="720"/>
    <w:docGrid w:linePitch="360"/>
  </w:sectPr>
</w:body>
```

2.7.2 Section Breaks

As well as specifying the section's properties, the type of section break is specified using the type element. WordprocessingML supports four distinct types of section breaks:

- *Next page section breaks* (the default if type is not specified), which begin the new section on the following page.
- *Odd page section breaks*, which begin the new section on the next odd-numbered page.
- *Even page section breaks*, which begin the new section on the next even-numbered page.
- *Continuous section breaks*, which begin the new section on the following paragraph. This means that continuous section breaks might not specify certain page-level section properties, since they must be inherited from the following section. These breaks, however, can specify other section properties, such as line numbering and footnote/endnote settings.

2.8 Styles

After looking at the primary elements of block-level content in a WordprocessingML file, it is now necessary to look at the information stored in the document that affects how this content is displayed.

The first such group of information is *styles*. Within a WordprocessingML file, *styles* are predefined sets of paragraph and/or character properties which can be applied to text within the document. This allows the formatting properties to be stored and managed independently from the content, allowing the look of document content to be changed in a single location (e.g., the look of all first-level headings is changed by changing the style with styleId `Heading1` rather than looking for and changing each paragraph in the document).

The Normal paragraph style in a word processing document can have any number of formatting properties, e.g., font face = Times New Roman; font size = 12pt; paragraph justification = left). All paragraphs that reference this paragraph style would automatically inherit these properties.

2.8.1 Styles Part

Style information in a WordprocessingML document is stored in the Styles part within the package, which is stored via an implicit relationship from the Main Document or Glossary Document part of relationship type `http://schemas.openxmlformats.org/wordprocessingml/2006/styles` and has a content type of `vnd-openxmlformats.officedocument.wordprocessingml-styles+xml`.

The styles part stores two types of style information for the document:

- Style definitions
- Latent style information

2.8.2 Style Definitions

Each style defined within a WordprocessingML document requires a *style definition*. The style definition contains all of the information needed by a consumer to store and display that style within a WordprocessingML document, and is defined using the style element. The style definition for any style in WordprocessingML can be divided into three segments (Note: the complete definition of style properties can be found on the reference for the style element):

- Common style properties
- Style ‘types’
- Type specific properties

Common style properties refer to the set of properties which can be used regardless of the type of style; for example, the style name, additional aliases for the style, a style ID (used by the document content to refer to the style), if style is hidden, if style is locked, etc.

Consider a style called Heading 1 in a document as follows:

```
<w:style w:type="paragraph" w:styleId="Heading1">
  <w:name w:val="heading 1"/>
  <w:basedOn w:val="Normal"/>
  <w:next w:val="Normal"/>
  <w:link w:val="Heading1Char"/>
  <w:priority w:val="1"/>
  <w:qformat/>
  <w:rsid w:val="00F303CE"/>
  …
</w:style>
```

Above the formatting information specific to this style type are a set of common style properties which define information shared by all style types.

Style types refer to the property on a style that defines the type of style created with this style definition. WordprocessingML supports six types of style definitions:

- Paragraph styles
- Character styles
- Linked styles (paragraph + character)
- Table styles
- Numbering styles
- Default paragraph + character properties

Consider a style called Heading 1 in a document as follows:

```
<w:style w:type="paragraph" w:styleId="Heading1">
  <w:name w:val="heading 1"/>
  <w:basedOn w:val="Normal"/>
  <w:next w:val="Normal"/>
  <w:link w:val="Heading1Char"/>
  <w:priority w:val="1"/>
  <w:qformat/>
  <w:rsid w:val="00F303CE"/>
  …
</w:style>
```

The type attribute has a value of `paragraph`, which indicates that the following style definition is a paragraph style.

2.8.3 Paragraph Styles

The first type of style definition, *paragraph styles* are styles that apply to the contents of an entire paragraph as well as the paragraph mark. This definition implies that the style can define both character properties (properties that apply to text within the document) as well as paragraph properties (properties which apply to

the positioning and appearance of the paragraph). Paragraph styles cannot be referenced by runs within a document, they must be referenced by the pStyle element within a paragraph's paragraph properties (pPr) element.

A paragraph style has three defining type-specific characteristics:

- The type attribute on the style has a value of paragraph, which indicates that the following style definition is a paragraph style.
- The next element defines an editing behavior which supplies the paragraph style to be automatically applied to the next paragraph when ENTER is pressed at the end of a paragraph of this style.
- The style specifies both paragraph-level and character-level properties using the pPr and rPr elements, respectively. In this case, the run properties are the set of properties applied to each run in the paragraph.

The paragraph style is then applied to paragraphs by referencing the styleId attribute value for this style in the paragraph properties' pStyle element.

Consider a paragraph style titled "Test Paragraph Style" which defines: paragraph spacing = double, paragraph indent = 1" (first line only); font = Algerian, font color = red, font size = 20 points. The resulting style definition would be:

```
<w:style w:type="paragraph" w:styleId="TestParagraphStyle">
  <w:name w:val="Test Paragraph Style"/>
  <w:qformat/>
  <w:rsid w:val="00F85845"/>
  <w:pPr>
    <w:spacing w:line="480" w:lineRule="auto"/>
    <w:ind w:firstLine="1440"/>
  </w:pPr>
  <w:rPr>
    <w:rFonts w:ascii="Algerian" w:hAnsi="Algerian"/>
    <w:color w:val="ED1C24"/>
    <w:sz w:val="40"/>
  </w:rPr>
</w:style>
```

Notice that the character properties for the style are under the rPr element, and the paragraph properties are under the pPr element.

The document content for a paragraph of this style would be:

```
<w:p>
  <w:pPr>
    <w:pStyle w:val="TestParagraphStyle"/>
  </w:pPr>
  <w:r>
    <w:t xml:space="preserve">Here is some fancy Text</w:t>
  </w:r>
</w:p>
```

The pStyle element links the paragraph with the style definition.

2.8.4 Character Styles

The next type of style definition, *character styles* are styles which apply to the contents of one or more runs of text within a document's contents. This definition implies that the style can only define character properties (properties which apply to text within a paragraph) because it cannot be applied to paragraphs. Character styles can only be referenced by runs within a document, and they must be referenced by the rStyle element within a run's run properties element.

A character style has two defining type-specific characteristics:

- The type attribute on the style has a value of `character`, which indicates that the following style definition is a character style.
- The style specifies only character-level properties using the rPr element. In this case, the run properties are the set of properties applied to each run which is of this style.

The character style is then applied to runs by referencing the styleId attribute value for this style in the run properties' rStyle element.

Consider a character style titled "Test Character Style" which defines; font = Courier New, font color = yellow; underline. The resulting style definition would be:

```
<w:style w:type="character" w:styleId="TestCharacterStyle">
  <w:name w:val="Test Character Style"/>
  <w:priority w:val="99"/>
  <w:qformat/>
  <w:rsid w:val="00E77BF0"/>
  <w:rPr>
    <w:rFonts w:ascii="Courier New" w:hAnsi="Courier New"/>
    <w:color w:val="FFF200"/>
    <w:u w:val="single"/>
  </w:rPr>
</w:style>
```

Notice that the character properties applied using this style are under the rPr element. The document content for a paragraph with a run of this style would be:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">The following text is in the </w:t>
  </w:r>
  <w:r>
    <w:rPr>
      <w:rStyle w:val="TestCharacterStyle"/>
    </w:rPr>
    <w:t>character style</w:t>
  </w:r>
  <w:r>
    <w:t>.</w:t>
  </w:r>
</w:p>
```

The rStyle element in the second run links that run with the style definition, inheriting the formatting properties for that run.

2.8.5 Linked Styles

The next type of style definition, *linked styles* are actually a paired combination of styles which can be applied to the contents of one or more runs of text within a document's contents or the entire contents of one or more paragraphs in a WordprocessingML document. This definition implies that the style can define both a set of character properties (properties which apply to text within a paragraph) as well as a set of paragraph properties (properties which apply to the positioning and appearance of the paragraph) because it must be possible to apply the style to paragraphs as well as characters.

In order to accomplish these dual uses, a linked style is actually a pairing of a paragraph style and a character style in the WordprocessingML document. Each style exists uniquely within the styles part, but are linked by the link element, which specifies that these styles are each half of a linked style definition and should be treated as one style at runtime.

A typical example of the use of a linked style is a quote style - if the style is applied to a paragraph, the quoted text should be indented additionally to create a block quote effect, but if the style is applied to text in a paragraph, only the character level effects should be applied.

Consider the following two styles which comprise a linked style pairing:

```
<w:style w:type="paragraph" w:styleId="TestLinkedStyle">
  <w:name w:val="Test Linked Style"/>
  <w:link w:val="TestLinkedStyleChar"/>
  <w:qformat/>
  <w:rsid w:val="009C1646"/>
```

```
      <w:pPr>
        <w:spacing w:line="480" w:lineRule="auto"/>
        <w:ind w:left="1440"/>
      </w:pPr>
      <w:rPr>
        <w:rFonts w:ascii="Arial" w:hAnsi="Arial"/>
        <w:color w:val="22B14C"/>
      </w:rPr>
    </w:style>
    <w:style w:type="character" w:styleId="TestLinkedStyleChar">
      <w:name w:val="Test Linked Style Char"/>
      <w:link w:val="TestLinkedStyle"/>
      <w:rsid w:val="009C1646"/>
      <w:rPr>
        <w:rFonts w:ascii="Arial" w:hAnsi="Arial"/>
        <w:color w:val="22B14C"/>
      </w:rPr>
    </w:style>
```

The link element in the paragraph style specifies `TestLinkedStyleChar`, the styleId of the paired character style, and the link element in the character style specifies `TestLinkedStyle`, the styleId of the paired paragraph style, creating a linked style combination.

Paragraph-level instances of linked styles can only be referenced by paragraphs within a document, and they must be referenced by the pStyle element within the paragraph's paragraph properties element (pPr), which must referrence the paragraph version of the linked style. Character-level instances of linked styles can only be referenced by a run's run properties element (rPr) within a document, and they must be referenced by the rStyle element within the run properties element which must reference the character version of the linked style.

Consider a linked style titled "Test Linked Style" which defines; font = Arial, font color = green; paragraph spacing = double, indent = 1" left. The resulting style definitions would be:

```
    <w:style w:type="paragraph" w:styleId="TestLinkedStyle">
      <w:name w:val="Test Linked Style"/>
      <w:link w:val="TestLinkedStyleChar"/>
      <w:qformat/>
      <w:rsid w:val="009C1646"/>
      <w:pPr>
        <w:pStyle w:val="TestLinkedStyle"/>
        <w:spacing w:line="480" w:lineRule="auto"/>
        <w:ind w:left="1440"/>
      </w:pPr>
```

```
    <w:rPr>
      <w:rFonts w:ascii="Arial" w:hAnsi="Arial"/>
      <w:color w:val="22B14C"/>
    </w:rPr>
  </w:style>
  <w:style w:type="character" w:styleId="TestLinkedStyleChar">
    <w:name w:val="Test Linked Style Char"/>
    <w:link w:val="TestLinkedStyle"/>
    <w:rsid w:val="009C1646"/>
    <w:rPr>
      <w:rFonts w:ascii="Arial" w:hAnsi="Arial"/>
      <w:color w:val="22B14C"/>
    </w:rPr>
  </w:style>
```

Notice that the linked style definition is composed of the paragraph style, which specifies both the run and paragraph properties, and the character style, which specifies only the run properties. The document content for a paragraph with this linked style would be:

```
  <w:p>
    <w:pPr>
      <w:pStyle w:val="TestLinkedStyle"/>
    </w:pPr>
    <w:r>
      <w:t xml:space="preserve">A para version of Test Linked Style.</w:t>
    </w:r>
  </w:p>
```

The pStyle element in the paragraph's properties links the paragraph with the paragraph version of the linked style definition.

The document content for a paragraph with a run of this linked style would be:

```
  <w:p>
    <w:r>
        <w:t xml:space="preserve">Next run is character version of </w:t>
    </w:r>
    <w:r>
      <w:rPr>
        <w:rStyle w:val="TestLinkedStyleChar"/>
      </w:rPr>
      <w:t>Test Linked Style</w:t>
    </w:r>
```

```
    <w:r>
      <w:t>.</w:t>
    </w:r>
  </w:p>
```

The rStyle element in the second run's properties links the run with the character version of the linked style definition.

2.8.6 Numbering Styles

Numbering styles are style definitions which specify common style properties for a multi-level numbering format within a document. This means that a numbering style defines only a single paragraph property: a reference to a numbering definition stored in the document's numbering part, using the numPr element.

Unlike paragraph and character styles, numbering styles are never directly referenced by content in the document – instead, an abstract numbering definition (covered in the numbering topic of this section) specifies that it is actually the underlying numbering information for a numbering style.

Consider a numbering style "Test Numbering Style":

```
<w:style w:type="numbering" w:styleId="TestNumberingStyle">
  <w:name w:val="Test Numbering Style" />
  <w:priority w:val="99" />
  <w:rsid w:val="0045009F" />
  <w:pPr>
    <w:numPr>
      <w:numId w:val="1" />
    </w:numPr>
  </w:pPr>
</w:style>
```

The only information specified in the numbering style definition is a reference to the numbering definition for the numbering information which is defined by this numbering style.

2.8.7 Table Styles

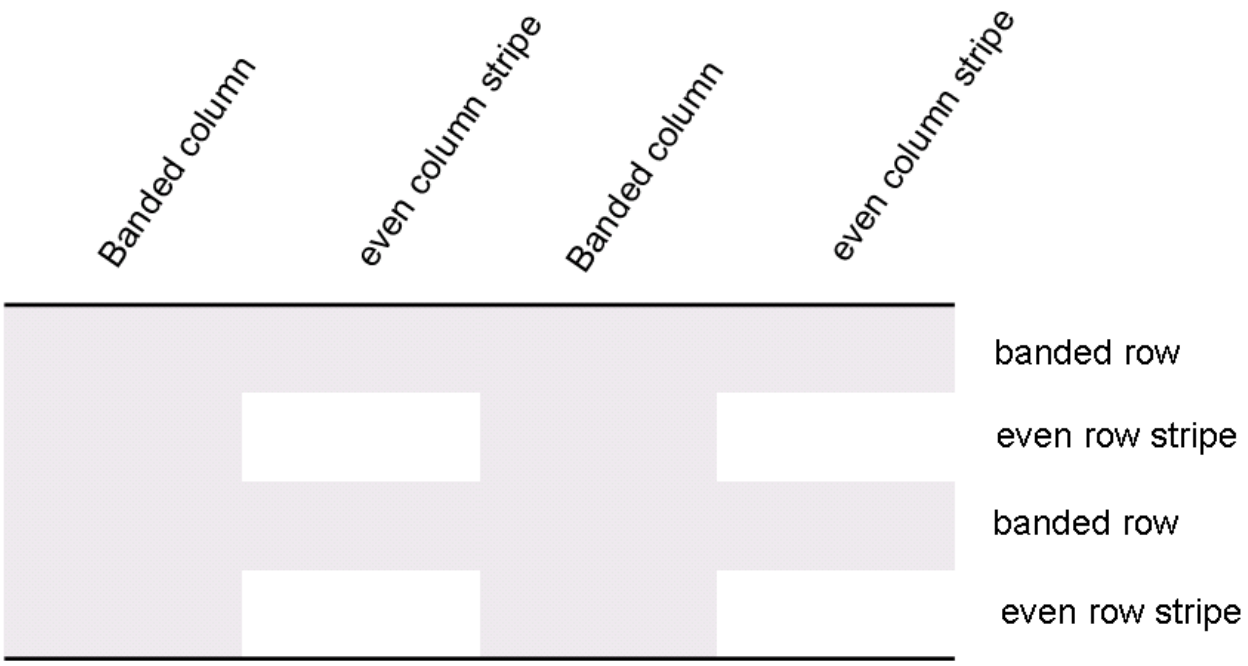
The last conventional type of style definition, table styles are styles which apply to the contents of zero or more tables within a document. This definition implies that the style can only define table properties (properties which apply to the table and its constituent rows and cells), however a table style can also define paragraph properties (properties which apply to the positioning and appearance of paragraphs) as well as character properties (properties which apply to runs) for all of the paragraphs and runs within the specified table in the document. Table styles can only be referenced by tables within a document, and they must be referenced by the tblStyle element within a table's table properties (tblPr) element.

Like the style definitions discussed above, table styles specify all of the properties that can be applied to a table, as well as paragraph and character properties for the table's contents. However, unlike other style definitions, table styles allow for the definition of conditional formats for different regions of the table.

These table conditional formats are applied to different regions of the table as follows:

Top left cell	Header row	Top right cell
First column	Table body	Last column
Bottom left cell	Footer row	Bottom right cell

All rows in the table can also have conditional formatting on an alternating row/column basis as well as follows:



When specified, these conditional formats are applied in the following order (therefore subsequent formats override properties on previous formats):

- Whole table
- Banded columns, even column banding
- Banded rows, even row banding
- First row, last row
- First column, last column
- Top left, top right, bottom left, bottom right

Consider a table style “Test Table Style” defined as follows: all cells with 1pt table borders on all sides, 0.1” cell margins on left and right of cells, and 0” cell margins on top and bottom of cells, as well as header row specific formatting of: red shading, bold text as follows:

```
<w:style w:type="table" w:styleId="TestTableStyle">
  <w:name w:val="Test Table Style"/>
  <w:basedOn w:val="TableNormal"/>
  <w:priority w:val="99"/>
  <w:rsid w:val="00340CC4"/>
  <w:tblPr>
    <w:tblBorders>
      <w:top w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:left w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:bottom w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:right w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:insideH w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:insideV w:val="single" w:sz="4" w:space="0" w:color="auto"/>
    </w:tblBorders>
    <w:tblCellMar>
      <w:top w:w="0" w:type="dxa"/>
      <w:left w:w="108" w:type="dxa"/>
      <w:bottom w:w="0" w:type="dxa"/>
      <w:right w:w="108" w:type="dxa"/>
    </w:tblCellMar>
  </w:tblPr>
  <w:tblStylePr w:type="firstRow">
    <w:rPr>
      <w:b/>
    </w:rPr>
    <w:tcPr>
      <w:shd w:val="clear" w:color="auto" w:fill="ED1C24"/>
    </w:tcPr>
  </w:tblStylePr>
</w:style>
```

The tblPr element holds the formatting which is applied to the entire table, and the tblStylePr element with a type attribute value of `firstRow` holds the formatting for the first table row, specifically the bold run property and red cell shading.

An individual instance of a table defines an association with a table style using the tblStyle element in the table's properties (tblPr), as discussed above. However, individual tables can choose whether to apply the following aspects of the table's conditional formats individually:

- First row
- Last row
- First column
- Last column
- Row banding
- Column banding

The use or omission conditional formats are specified using the tblLook element, which contains a bitmask representing which properties are applied and omitted.

Consider two tables using the table style "Style2"; one which specifies that it should only use the header row and footer row conditional formatting properties from the table style, and the other which specifies that it should use the header row, footer row, and banded row conditional formatting:

```
<w:tbl>
  <w:tblPr>
    <w:tblStyle w:val="Style2"/>
    <w:tblW w:w="0" w:type="auto"/>
    <w:tblLook w:val="0660"/>
  </w:tblPr>
  …
</w:tbl>
…
<w:tbl>
  <w:tblPr>
    <w:tblStyle w:val="Style2"/>
    <w:tblW w:w="0" w:type="auto"/>
    <w:tblLook w:val="0460"/>
  </w:tblPr>
  …
</w:tbl>
```

The tables each specify the appropriate set of conditional formats using the tblLook element, as seen by the identical table styles in the tblStyle element, and different tblLook values.

2.8.8 Default Document Paragraph and Character Properties

The final type of style in a WordprocessingML document is the default paragraph and character properties for the document. Although this is not a style in the strict sense of the word (because this property set cannot directly be applied to text) it defines the basic set of formatting properties which are inherited by paragraphs and runs in the document.

The following section, entitled Style Inheritance, explains exactly how the default document paragraph and character properties influence the appearance of all content in the document.

2.8.9 Style Inheritance

In order to compile the complete set of paragraph and character properties specified by any given style (as appropriate), a consumer must follow the rule of style inheritance to determine each property in that set.

Style inheritance states that styles of any given type may inherit from other styles of that type, and therefore a consumer must 'build up' the style information by following the inheritance tree. This inheritance is defined via the basedOn element, which specifies the styleId of the parent style.

The "Tristan Test" paragraph style can inherit properties from the "Heading 1" paragraph style, which itself can inherit properties from the "Normal" paragraph style.

To build up the resulting style, a consumer must trace the hierarchy (following each basedOn value) back to a style which has no basedOn element (is not based on another style). The resulting style is then constructed by following each level in the tree, applying the specified paragraph and/or character properties as appropriate. When properties conflict, they are overridden by each subsequent level (this includes turning OFF a property set at an earlier level). Properties which are not specified simply do not change those specified at earlier levels.

Consider a character style "Green" which specifies only that the text color is green, but inherits from another character style "Base" which defines a font face of Arial, as well as bold:

```
<w:style w:type="character" w:styleId="Green">
  <w:name w:val="Green" />
  <w:basedOn w:val="Base" />
  <w:rPr>
    <w:color w:val="22B14C" />
  </w:rPr>
</w:style>
  …
../Local Settings/Temp/styles.xml<w:style w:type="character" w:styleId="Base">
  <w:name w:val="Base" />
  <w:rPr>
    <w:rFonts w:ascii="Arial" w:hAnsi="Arial" />
    <w:b />
  </w:rPr>
</w:style>
```

The definition of the Green character style has a basedOn element which specifies the Base style. This means that any use of the Green style is defined as bold, green, Arial text.

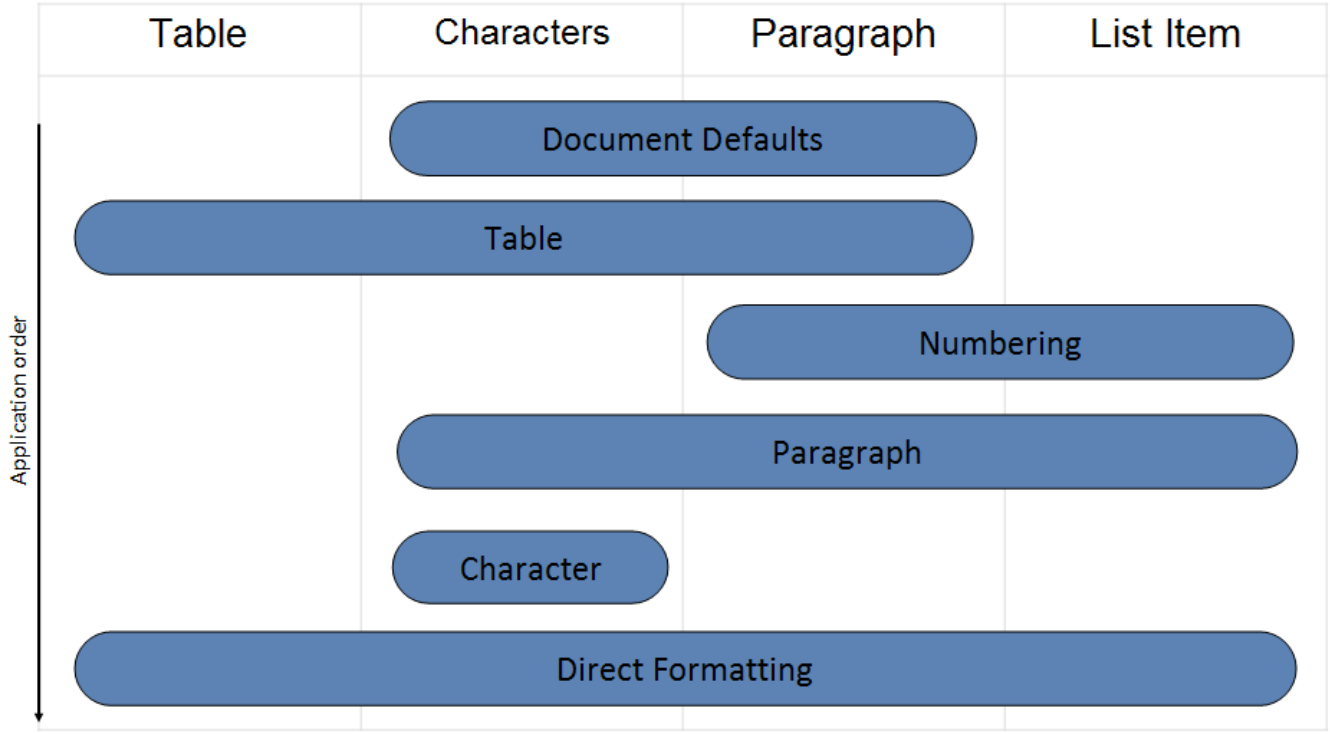
Conversely, a producer should not output any property on a style which has already been set by a previous level of the style hierarchy, as well as those which match the document defaults. This means that if the document defaults or any previous level in a style's hierarchy specify a property which is unchanged at this level, that property should not be part of the style definition in the resulting WordprocessingML. Adding a property at multiple levels in the style hierarchy is not invalid, but unnecessarily duplicative as the setting is already applied to the text, resulting in an unnecessary increase to file size.

If the document default font is Bauhaus 93 and the Heading 1 style also specifies the Bauhaus 93 font, then a producer should not output any rFonts element for the Heading 1 style definition, because that formatting is inherited from the document defaults.

2.8.10 Style Application

With the various flavors of styles available, multiple style types can be applied to the same content within a file, which means that properties must be applied in a specific deterministic order. As with inheritance, the resulting formatting properties set by one type can be unchanged, removed, or altered by following types.

The following table illustrates the order of application of these defaults, and which properties are impacted by each:



This process can be described as follows: First, the document defaults are applied to all runs and paragraphs in the document. Next, the table style properties are applied to each table in the document, following the conditional formatting inclusions and exclusions specified per table. Next, numbered item and paragraph properties are applied to each paragraph formatted with a numbering style. Next, paragraph and run properties are applied to each paragraph as defined by the paragraph style. Next, run properties are applied to each run with a specific character style applied. Finally, we apply direct formatting (paragraph or run properties not from styles).

2.8.11 Latent Styles

The final piece of information stored in the styles part in the document, aside from style definition information, is *latent style* information.

Latent styles are all styles contained in a document's template which have not yet been instantiated (used) in the current instance of the document.

In WordprocessingML, there are often properties which must be set on all styles in a document template regardless of whether they are being used: for example, whether or not the style can be applied in the current document (locked state), UI sorting priority, whether the style should be shown in the user interface, etc. In order for the document to function correctly, it is essential that this information is stored within the document, so that a consumer can determine the necessary style information from the document alone (without access to the template). However, it would be grossly inefficient for the document to store all style information for all styles simply to store this information, so latent styles are used to store just the necessary style properties without caching all style information in the document.

In order to do this efficiently, the document declares a latentStyles element in the styles part which defines the default properties applied to all latent styles in the document. All styles whose properties do not match the default for the set of style properties which must be defined for all styles are explicitly defined using the lsdException element.

Consider the following latent style information stored in a document's styles part:

```
<w:latentStyles w:defLockedState="off" w:defPriority="99"
  w:defSemiHidden="on" w:defUnhideWhenUsed="on" w:defQFormat="off"
  w:count="180">
  <w:lsdException w:name="Normal" w:unhideWhenUsed="off"
    w:qformat="on"/>
  <w:lsdException w:name="heading 1" w:semiHidden="off" w:priority="1"/>
  <w:lsdException w:name="heading 2" w:priority="1"
    w:unhideWhenUsed="on"/>
  <w:lsdException w:name="heading 3" w:semiHidden="off/>
  <w:lsdException w:name="heading 4" w:priority="1" w:qformat="on"/>
  <w:lsdException w:name="heading 5" w:priority="1" w:qformat="on"/>
```

```
    <w:lsdException w:name="heading 6" w:priority="1" w:qformat="on"/>
    <w:lsdException w:name="heading 7" w:priority="1" w:qformat="on"/>
    <w:lsdException w:name="heading 8" w:priority="1" w:qformat="on"/>
    <w:lsdException w:name="heading 9" w:priority="1" w:qformat="on"/>
    <w:lsdException w:name="Normal Indent" w:priority="6" w:qformat="on"/>
  </w:latentStyles>
```

The attributes on the latentStyles element define the properties applied to all latent styles for this document. All styles whose properties do not match the default latent styles properties are explicitly defined using the values on the lsdException elements.

2.9 Fonts

2.9.1 Font References

Within a WordprocessingML document, font face information can be referenced by any set of run properties, both as part of a style definition or direct formatting on one or more runs in the document's contents. This reference is established by referencing the primary name of the font face that is used in the rFonts element of the run properties, linking that run with the desired font face.

For example, consider a run of text that has been directly formatted to use the `Arial Black` font face. This setting would be specified as follows on the run's properties:

```
  <w:r>
    <w:rPr>
      <w:rFonts w:ascii="Arial Black" w:hAnsi="Arial Black" />
    </w:rPr>
    <w:t>This run of text uses the Arial Black font face.</w:t>
  </w:r>
```

The rFonts element specifies that the run should be formatted using the `Arial Black` font face. Applications can then look up and use the font with primary name of `Arial Black` when formatting this run.

2.9.2 Font Reference Types

In the example above, two attributes were present, both referring to the font face with primary name `Arial Black`. This simple case illustrates the ability for a WordprocessingML document to store up to four fonts which may be used on the contents of a run, as follows:

- ASCII font
- High ANSI font
- East Asian font
- Complex Script font

Each of these font faces is used to format the characters in the run that fall under their purview:

The *ASCII font* formats all characters in the ASCII range (character values 0–127). This font is specified using the ascii attribute on the rFonts element.

The *East Asian font* formats all characters that belong to Unicode sub ranges for East Asian languages. This font is specified using the eastAsia attribute on the rFonts element.

The *complex script font* formats all characters that belong to Unicode sub ranges for complex script languages. This font is specified using the cs attribute on the rFonts element.

The *high ANSI font* formats all characters that belong to Unicode sub ranges other than those explicitly included by one of the groups above. This font is specified using the hAnsi attribute on the rFonts element.

For example, consider a run of text defined as follows:

```
<w:r>
  <w:rPr>
    <w:rFonts w:ascii="Arial Black" w:hAnsi="Arial Black" w:cs="Arial"
      w:eastAsia="SimSun"/>
  </w:rPr>
  …
</w:r>
```

The rFonts element specifies that the contents of this run are formatted as follows:

- Complex script characters used the `Arial` font
- East Asian characters used the `SimSun` font
- All other characters used the `Arial Black` font

2.9.3 Ambiguous Characters

When classifying characters into one of the four slots defined above, it is likely that the classification of some characters will be ambiguous (the resulting classification would be equally applicable for one or more font slots).

To handle this, the font face information can also include a hint, which specifies how ambiguous mappings are resolved into a font slot. This information is stored on the hint attribute on the rFonts element, and specifies the bucket into which these ambiguous characters fall.

For example, if the hint attribute has a value of `eastAsia`, then all ambiguous characters shall be formatted using the East Asian font face.

2.9.4 Font Table

Within a document, the *font table* contains information about the fonts used in the document to allow:

- Applications to perform substitution with the most appropriate possible font when the desired font face is not available on the system. Since some fonts are commercially distributed, it is possible for a

document to be formatted with one or more fonts that are not available depending on the machine opening the current system. This information allows the application that cannot locate the desired font to perform the most appropriate possible match.

- Embedding of fonts in the document to prevent the need for font substitution

The font table part is stored via an implicit relationship from either the main document part or the glossary document part, and has a relationship type of http://schemas.openxmlformats.org/wordprocessingml/2006/fontTable, and a content type of vnd-openxmlformats.officedocument.wordprocessingml-fontTable+xml.

2.9.5 Font Substitution Data

The first classification of data stored in the font table are an optional set of font metrics which are queried from the font and stored in the document such that future applications can utilize them when the desired font is not available. If a particular font face cannot be located on the current system, then this data is used to substitute a font that most appropriately matches its characteristics.

For example, consider the font substitution data stored for the `Arial Black` font:

```
<w:font w:name="Arial Black">
  <w:panose1 w:val="020B0A04020102020204" />
  <w:charset w:val="00" />
  <w:family w:val="swiss" />
  <w:pitch w:val="variable" />
  <w:sig w:usb0="00000287" w:usb1="00000000" w:usb2="00000000"
    w:usb3="00000000" w:csb0="0000009F" w:csb1="00000000" />
</w:font>
```

This data is linked to the font face with a name of `Arial Black` via the name attribute, and stores the following information about the font (see the reference material on fonts for more details):

- The font's Panose-1 number
- The character set of the font
- The font's family
- The font's pitch
- The code pages and Unicode sub ranges supported by the font

2.9.6 Font Embedding

As well as providing information about the font's metrics, applications may be directed to embed the contents of a font (partially or as a whole) into a document, a process known as font embedding. *Font embedding* literally embeds an obfuscated version of the font into the file so that it may be retrieved and used to view the contents of this document - but the obfuscation ensures that the font cannot be extracted and used for any other document (as it may have a commercial license).

Within the font table, when a font is embedded there are explicit relationships to each font form needed:

- Regular
- Bold
- Italic
- Bold + Italic

Each form is obfuscated using the mechanism described in the reference material on this subject.

2.9.7 Theme Fonts

As well as storing standard font face information, run properties may store an abstraction for font face information known as theme fonts. *Theme fonts* are values that specify that the font face information for a run is not stored in the attribute value using the appropriate font face name, but is rather a reference into the document's theme part, allowing font face information to be stored and managed centrally as part of the theme data. It is appropriate to think of theme fonts as a "style for fonts" in the same way in which a style is a reference to the formatting that is stored centrally in another part.

Theme fonts are specified using the theme attribute variants in the rFonts element, rather than storing the actual font face name.

For example, consider a run of text defined as follows:

```
<w:r>
  <w:rPr>
    <w:rFonts w:asciiTheme="minorHAnsi" w:hAnsiTheme="minorHAnsi" />
  </w:rPr>
  …
</w:r>
```

The rFonts element's attribute values of asciiTheme and hAnsiTheme both store a reference to a theme font stored in the document's theme part (i.e., there is no font with the primary name `minorHAnsi`).

Once this information has been established, it is combined with the theme language data stored in the document's settings to resolve the appropriate theme fonts from the theme part. The syntax and format of the theme part are stored in the DrawingML syntax and discussed in that section.

2.10 Numbering

Numbering in WordprocessingML refers to symbols—Arabic numerals, Roman numerals, symbol characters ("bullets"), text strings, etc.—that are used to label individual paragraphs of text.

The following two paragraphs each contain numbering as defined by WordprocessingML: the first uses an Arabic numeral, the second a symbol character:

1. This is a paragraph with numbering information.

- This is also a paragraph with numbering information.

2.10.1 Numbering Part

Numbering information in a WordprocessingML document is stored in the Numbering part within the package, which is stored via an implicit relationship from the Main Document part or Glossary Document part of relationship type `http://schemas.openxmlformats.org/wordprocessingml/2006/numbering` and has a content type of `vnd-openxmlformats.officedocument.wordprocessingml-numbering+xml`.

2.10.2 Numbering Definitions

The specification of a specific set of numbering information is called a *numbering definition*. Numbering definitions are stored in two components:

- Abstract numbering definitions
- Numbering definition instances

As shown below, their relationship is (essentially) that of an abstract and an inherited class.

2.10.3 Abstract Numbering Definitions

An *abstract numbering definition* is the basis for all numbering information in a WordprocessingML document, as it defines the appearance and behavior of a specific set of numbered paragraphs in a document, and is defined using the abstractNum element. Although abstract numbering definitions contain all of the numbering information for one type of numbering, they cannot be directly referenced by content (hence their abstract designation), they must be inherited by a numbering definition instance, which itself can be referenced by content. A specific abstract numbering definition in WordprocessingML can be divided into two parts:

- Common numbering properties
- Numbering levels

The complete definition of all abstract numbering properties can be found in the reference for the abstractNum element.

Common numbering properties refer to the properties that can be specified by all abstract numbering definitions regardless of their contents. Examples of common numbering properties include: a numbering ID (which uniquely identifies a numbering definition), the numbering definition type (single level, multi-level, multi-level hybrid), the numbering name, and optional numbering style references, as discussed in detail later in this subclause.

Consider the following example of an abstract numbering definition in a WordprocessingML document:

```
<w:abstractNum w:abstractNumId="4">
  <w:nsid w:val="1DE04504" />
  <w:multiLevelType w:val="hybridMultilevel" />
  <w:lvl w:ilvl="0" w:tplc="0409000F">
    …
  </w:lvl>
```

```
    <w:lvl w:ilvl="1" w:tplc="04090019">
      …
    </w:lvl>
    <w:lvl w:ilvl="2" w:tplc="04090019">
      …
    </w:lvl>
    <w:lvl w:ilvl="3" w:tplc="0409000F">
      …
    </w:lvl>
    …
  </w:abstractNum>
```

This numbering definition specifies two common properties: a numbering ID (using the nsid element) of 1DE04504, and a list type (using the multiLevelType element) of hybridMultilevel, which specifies that this abstract numbering definition is more than one level and contains multiple numbering formats.

The other part of an abstract numbering definition is the specification of one or more numbering levels, each of which defines a unique set of formatting properties for one level in this numbering definition.

Consider three numbered paragraphs that reference the same numbering definition, but each, in turn, reference a different level within that list:

1. One
    a. Two
        i. Three

Although the paragraphs each reference the same abstract numbering definition (which is discussed later), each refers to a separate level within that abstract numbering definition, and therefore each has a unique set of paragraph and numbering properties.

It is important to note that the concept of levels in an abstract numbering definition refers to the levels as defined in the file format, and in no way the logical indentation of numbered paragraphs within a WordprocessingML document.

Consider another set of numbered paragraphs in WordprocessingML, where each subsequent paragraph is a different level but references the same abstract numbering definition:

1) Level one
    a) Level two
i) Level three
    (1) Level four
        (a) Level five

In this example, the properties of each level of the numbering definition is such that the paragraphs for each level are indented arbitrarily. However, this is still a completely valid numbering definition, and the paragraphs each represent subsequent levels of the same numbering definition.

Within an abstract numbering level definition, each numbering level is represented by an lvl element that defines a single level of numbering information. Numbering levels specify the following properties: starting number value, a number format presentation code (e.g., 1 vs. the string literal One), an associated paragraph style, which previous level should cause this numbering level to restart, the numbering text, number justification, a paragraph properties indentation for this level, etc. The complete definition of all numbering level properties can be found on the reference for the lvl element.

Consider the following numbering level definition in WordprocessingML:

```
<w:lvl w:ilvl="1">
  <w:start w:val="4"/>
  <w:nfc w:val="3"/>
  <w:pStyle w:val="Heading1"/>
  <w:lvlText w:val="BEFORE %2 AFTER %1 END"/>
  <w:lvlJc w:val="left"/>
  <w:pPr>
    <w:tabs>
      <w:tab w:val="num" w:pos="2880"/>
    </w:tabs>
    <w:ind w:left="288" w:firstLine="1152"/>
  </w:pPr>
</w:lvl>
```

This particular numbering level defines the following information:

- This is level 1 (the second level) for this numbering definition
- Start at number 4
- Use number format 3 (which translates to 1, 2, 3, and so on)
- When this level is used, apply the Heading1 style
- Use the following level text for the number: BEFORE %2 AFTER %1 END
- Left justify the number
- Set a left indent of 288 twentieths of a point, and a first line indent of 1152 twentieths of a point

This information is used to display the number for paragraphs of level 1 the reference this numbering definition.

Of particular significance is the lvlText element, which defines the content of the number text for each numbering level. Its syntax allows any string literal to be placed in the number (e.g., the ARTICLE in ARTICLE I, ARTICLE II, ARTICLE III, and so on), as well as the current value of the number for this or any previous level in the list.

Referring to the numbering level definition above, the lvlText is defined as follows:

```
<w:lvlText w:val="BEFORE %2 AFTER %1 END"/>
```

This level text specifies three literal strings (BEFORE, AFTER, END) mixed with the current numbering value from level 1 and level 0 in the document. Therefore, assuming level 0 is just a simple number, when inserted it would read:

```
1
  BEFORE 1 AFTER 1 END
  BEFORE 2 AFTER 1 END
  BEFORE 3 AFTER 1 END
2
  BEFORE 1 AFTER 2 END
  BEFORE 2 AFTER 2 END
  …
```

The %1 and %2 values correspond to the value for level 0 and 1 of this list, respectively.

2.10.4 Numbering Definition Instances

A numbering definition instance is a specific instantiation of numbering information that can be referenced by zero or more paragraphs within the document. A numbering definition instance is defined using the num element. A specific numbering definition instance in WordprocessingML can be divided into two parts:

- An abstract numbering reference
- (Optional) level overrides

The definition of all numbering definition instance properties can be found on the reference for the num element.

The required piece of information in a numbering definition instance, the instance must reference an abstract numbering definition using the abstractNumId element. This element specifies the value of the abstractNumId attribute for the inherited abstract numbering definition information.

Consider the WordprocessingML for a document with four numbering definition instances, two of which reference the same underlying abstract numbering definition:

```
<w:numbering>
  ...
  <w:num w:numId="2">
    <w:abstractNumId w:val="0" />
  </w:num>
  <w:num w:numId="3">
    <w:abstractNumId w:val="1" />
  </w:num>
```

```
    <w:num w:numId="4">
      <w:abstractNumId w:val="4" />
    </w:num>
    <w:num w:numId="5">
      <w:abstractNumId w:val="4" />
    </w:num>
  </w:numbering>
```

As shown above, the first two numbering definition instances reference abstractNumId values of 0 and 1 respectively, and the last two both reference the abstract numbering definition with an abstractNumId of 4.

The second (and optional) piece of information for a numbering definition instance is one or more numbering level overrides using the lvlOverride element. This element specifies a set of optional overrides applied to zero or more levels from the abstract numbering definition inherited by this instance.

Consider a numbering definition instance that inherits its information from the abstract numbering definition with abstractNumId of 4, but wishes to use a different set of properties for level 0 of the numbering definition. The resulting WordprocessingML would look like:

```
  <w:num w:numId="6">
    <w:abstractNumId w:val="4" />
    <w:lvlOverride w:ilvl="0">
      <w:lvl w:ilvl="0">
        <w:start w:val="4" />
        <w:lvlText w:val="%1)" />
        <w:lvlJc w:val="left" />
        <w:pPr>
          <w:ind w:left="360" w:hanging="360" />
        </w:pPr>
      </w:lvl>
    </w:lvlOverride>
  </w:num>
```

This level overrides level 0 of the list with the specified set of numbering properties, replacing those in the abtract numbering definition.

2.10.5 Applying Numbering to Paragraphs

Once numbering information is defined in the numbering part, this information must be associated with paragraphs within the document in order to display numbering on one or more paragraphs of content.

To accomplish this, numbered paragraphs are identified by the numPr element within the paragraph's properties element (the pPr element). The numbering properties within a paragraph are specified using two specific elements that specify the numbering definition information to use:

- A numbering definition instance reference

- A numbering level reference

The numbering definition instance reference is specified using the numId element. This element contains a reference to the numId attribute in a specific numbering definition instance within the numbering part, which links this paragraph to that numbering definition instance.

The numbering level reference is specified using the ilvl element. This element contains a reference to the ilvl attribute in the specified numbering definition instance's level information, which specifies the numbering level within the referenced numbering definition instance to be used by this numbered paragraph.

Consider the following numbered paragraphs in a WordprocessingML document:

1. Level one item one
    a. Level two item one
2. Level one item two
    a. Level two item one

These four numbered paragraphs, all referencing the same numbering definition, produce the following WordprocessingML:

```
<w:p>
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="0" />
      <w:numId w:val="5" />
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>Level one item one</w:t>
  </w:r>
</w:p>
```

```
<w:p>
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="1" />
      <w:numId w:val="5" />
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>Level two item one</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="0" />
      <w:numId w:val="5" />
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>Level one item two</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="1" />
      <w:numId w:val="5" />
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>Level two item one</w:t>
  </w:r>
</w:p>
```

In these numbered paragraphs, level 0 and 1 of the numbering definition are referenced through the ilvl element with a val attribute of `0` or `1`, respectively, however, the numId element always references the numbering definition instance with a val of `5`.

The numbering at any particular numbering level is restarted when a paragraph in the current document from the same numbering definition uses the level specified in the lvlRestart element for this numbering level.

Consider a set of numbered paragraphs in a WordprocessingML document where:

- Level 1 is set to restart after each level 0 (lvlRestart of 1)

- Level 2 is set to never restart (lvlRestart of 0)

1) Level one
    a) Level two – restarts after each level one
        i) Level three – never restarts
    b) Level two – restarts after each level one
        ii) Level three – never restarts
2) Level one
    a) Level two – restarts after each level one
        iii) Level three – never restarts
    b) Level two – restarts after each level one
        iv) Level three – never restarts

As the example shows, the numbering at level 1 (a, b, c, and so on) restarts after each level 0 is used, but level 2 (i, ii, iii, and so on) never restarts.

2.10.6 The Complete Story

To summarize the use of numbering information in a document, the paragraph properties specify a numPr element, which references a numbering definition instance via the numId element. The numbering definition instance specifies an inherited abstract numbering definition via the abstractNumId element. The paragraph then also specifies the list level from the numbering definition instance using the ilvl element.

Consider the following WordprocessingML for a numbered paragraph:

```
<w:p>
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="0" />
      <w:numId w:val="5" />
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>Numbered paragraph</w:t>
  </w:r>
</w:p>
```

Based on the numId of 5, the paragraph uses the numbering definition instance with a numId of 5:

```
<w:numbering>
  ...
  <w:num w:numId="5">
    <w:abstractNumId w:val="4" />
  </w:num>
</w:numbering>
```

Based on the abstractNumId of 4, this instance inherits the abstract numbering definition with an abstractNumId of 4:

```
<w:numbering>
  <w:abstractNum w:abstractNumId="4">
    <w:nsid w:val="FFFFFF7F" />
    <w:multiLevelType w:val="singleLevel" />
    <w:lvl w:ilvl="0">
      <w:start w:val="1" />
      <w:lvlText w:val="%1." />
      <w:lvlJc w:val="left" />
      <w:pPr>
        <w:tabs>
          <w:tab w:val="num" w:pos="720" />
        </w:tabs>
        <w:ind w:left="720" w:hanging="360" />
      </w:pPr>
    </w:lvl>
  </w:abstractNum>
  …
</w:numbering>
```

Since the numbering definition instance does not specify an override for ilvl 0, the definition for the corresponding level from the abstract numbering definition is applied to the text.

2.10.7 Numbering Styles

As stated earlier in the styles subclause (§2.8), numbering styles are style definitions which specify common formatting properties for a multi-level numbering format within a document. This means that a numbering style definition in the styles part defines only a single property: a reference to a numbering definition instance stored in the document's numbering part, using the numId element within the numPr element. That numbering definition instance specifies an abstract numbering style, which contains the numbering level information for the numbering style. It also specifies that it is the basis for the numbering style by back-referencing the numbering style's styleId attribute via the styleLink element.

Unlike paragraph and character styles, numbering styles are never directly referenced by content in the document—instead, an abstract numbering definition specifies that it contains the underlying numbering

information for a numbering style, and one or more numbering definition instances reference that abstract numbering definition.

2.10.8 Referencing Numbering Styles

To use a numbering style in a document, the paragraph properties for one or more paragraphs again specify a numPr element, which references a numbering definition instance via the numId element. The numbering definition instance itself again specifies an inherited abstract numbering definition via the abstractNumId element.

At this stage, the abstract numbering definition specifies that it is based on a numbering style via either of the following:

- The abstract numbering style contains no level data, and simply specifies a reference to the numbering style's styleId attribute via the numStyleLink element.
- The abstract numbering style contains the numbering level information for the numbering style, and specifies that it is the basis for the numbering style by referencing the numbering style's styleId attribute via the styleLink element.

Although the result of each method is identical, the following two examples illustrate each of the syntaxes:

Consider the first numbering style syntax, in which the numbering on a paragraph is based on an abstract numbering definition which simply references the numbering style via numStyleLink. The contents of the paragraph would consist of the following:

```
<w:p>
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="0" />
      <w:numId w:val="6" />
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>This paragraph references a numbering style via numStyleLink.</w:t>
  </w:r>
</w:p>
```

The numId element references a numbering definition instance with a value of 6, located in the numbering part:

```
<w:num w:numId="6">
  <w:abstractNumId w:val="0" />
</w:num>
```

Based on the abstractNumId of 0, this instance inherits the abstract numbering definition with an abstractNumId of 0:

```
<w:abstractNum w:abstractNumId="0">
  <w:nsid w:val="38901FA4" />
  <w:multiLevelType w:val="multilevel" />
  <w:numStyleLink w:val="TestNumberingStyle" />
</w:abstractNum>
```

This abstract numbering definition contains no numbering information - it simply notes that it inherits the numbering information from the numbering style `TestNumberingStyle` by referencing the styleId attribute on that style:

```
<w:style w:type="numbering" w:styleId="TestNumberingStyle">
  <w:name w:val="Test Numbering Style" />
  <w:uiPriority w:val="99" />
  <w:rsid w:val="00DB3C4B" />
  <w:pPr>
    <w:numPr>
      <w:numId w:val="4" />
    </w:numPr>
  </w:pPr>
</w:style>
```

The style references a numbering definition instance, again via the numId element:

```
<w:num w:numId="4">
  <w:abstractNumId w:val="2" />
</w:num>
```

Based on the abstractNumId of 2, this instance inherits the abstract numbering definition with an abstractNumId of 2:

```
<w:abstractNum w:abstractNumId="2">
  <w:nsid w:val="46364EB7" />
  <w:multiLevelType w:val="multilevel" />
  <w:styleLink w:val="TestNumberingStyle" />
  <w:lvl w:ilvl="0">
    <w:lvlText w:val="%1 %1 %1" />
    <w:lvlJc w:val="left" />
    <w:pPr>
      <w:tabs>
        <w:tab w:val="num" w:pos="360" />
      </w:tabs>
      <w:ind w:left="0" w:firstLine="0" />
    </w:pPr>
  </w:lvl>
  …
</w:abstractNum>
```

This abstract numbering definition defines the properties for each level of the numbering format (levels 1 through 9 omitted for brevity). Since neither of the numbering definition instances specified overrides for level 0, the properties from abstract numbering format 2 are applied to level 0 in the resulting numbering definition instance and are applied to the text via the ilvl element.

Consider the second numbering style syntax, in which the numbering on a paragraph is based on an abstract numbering definition which defines the numbering information and references the numbering style via styleLink. The contents of the paragraph would consist of the following:

```
<w:p>
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="0" />
      <w:numId w:val="4" />
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>This paragraph references a numbering style via styleLink.</w:t>
  </w:r>
</w:p>
```

The numId element references a numbering definition instance with a value of 4, located in the numbering part:

```
<w:num w:numId="4">
  <w:abstractNumId w:val="2" />
</w:num>
```

Based on the abstractNumId of 2, this instance inherits the abstract numbering definition with an abstractNumId of 2:

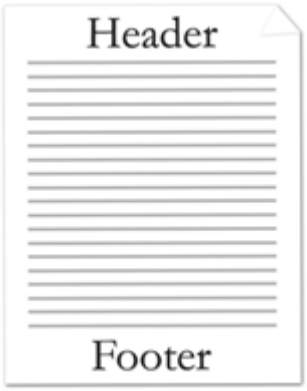
```
<w:abstractNum w:abstractNumId="2">
  <w:nsid w:val="46364EB7" />
  <w:multiLevelType w:val="multilevel" />
  <w:styleLink w:val="TestNumberingStyle" />
  <w:lvl w:ilvl="0">
    <w:lvlText w:val="%1 %1 %1" />
    <w:lvlJc w:val="left" />
    <w:pPr>
      <w:tabs>
        <w:tab w:val="num" w:pos="360" />
      </w:tabs>
      <w:ind w:left="0" w:firstLine="0" />
    </w:pPr>
  </w:lvl>
  …
</w:abstractNum>
```

This abstract numbering definition defines the properties for each level of the numbering format (levels 1 through 9 omitted for brevity) and specifies that it is the underlying numbering information for a numbering format by referencing the styleId of that numbering style via the styleLink element. Since the numbering definition instances specified no override for level 0, the properties from abstract numbering format 2 are applied to level 0 in the resulting numbering definition instance and are applied to the text via the ilvl element.

2.11 Headers and Footers

Headers and *footers* refer to text, graphics, or data (such as page number, date, document title, and so on) that can appear at the top or bottom of each page in a WordprocessingML document.

A *header* appears in the top margin (above the main document content on the page), while a *footer* appears in the bottom margin of a document page (below the main document content on the page); for example:



Since WordprocessingML is a flow-based format, headers and footers are applied by specifying the headers and footers for all pages in a particular section of a document.

2.11.1 Header Part

Header information in a WordprocessingML document is stored in a header part within the package, which is stored via an implicit relationship from the Main Document part or the Glossary Document part of relationship type `http://schemas.openxmlformats.org/wordprocessingml/2006/header` and has a content type of `vnd-openxmlformats.officedocument.wordprocessingml-header+xml`.

2.11.2 Footer Part

Footer information in a WordprocessingML document is stored in a footer part within the package, which is stored via an implicit relationship from the Main Document part or the Glossary Document part of relationship type `http://schemas.openxmlformats.org/wordprocessingml/2006/footer` and has a content type of `vnd-openxmlformats.officedocument.wordprocessingml-footer+xml`.

2.11.3 Headers and Footers

As described above, header and footer information is stored in one or more header or footer parts within the package.

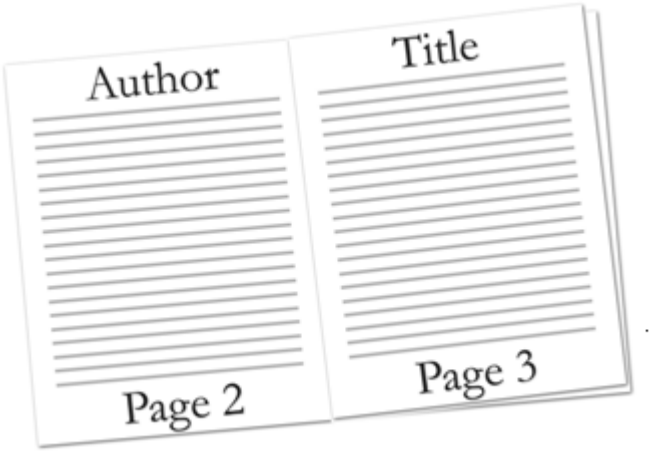
The hdr element defines a single header for the document, while the ftr element defines a single footer for the document. Headers and footers are just another document story in WordprocessingML. Within the root element of the header or footer, the content of the element is similar to the content of the body element, and contains what is referred to as *block-level markup* —markup that can exist as a sibling element to paragraphs in a WordprocessingML document.

Within each section of a document there can be up to three different types of headers and footers:

- First page header/footer
- Odd page header/footer
- Even page header/footer

First page headers and *footers* specify a unique header or footer that shall appear on the first page of a section. *Odd page headers* and *footers* specify a unique header and footer that shall appear on all odd numbered pages for a given section. *Even page headers* and *footers* specify a unique header and footer that shall appear on all even numbered pages in a given section.

Different headers or footers can be useful for bounded documents like books, as shown in the figure below.



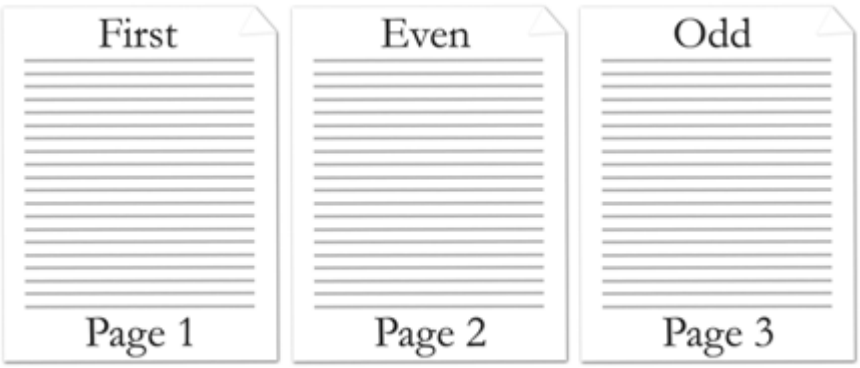
Consider the following simple one-page document with one header:

This document defines one header with the text `Header`. The header's content is stored in a unique Header part. The resulting header is represented by the following WordprocessingML:

```
<w:hdr>
  <w:p>
    <w:r>
      <w:t>Header</w:t>
    </w:r>
  </w:p>
</w:hdr>
```

Since headers are containers of block level contents, all block level contents can be used within them. In this particular example, the content is a single paragraph.

Consider a more complex three-page document with different first, odd, and even page headers defined:



This document defines three headers stored in three different header parts. The resulting headers are represented by the following WordprocessingML:

First page header part:

```
<w:hdr>
  <w:p>
    <w:r>
      <w:t>First</w:t>
    </w:r>
  </w:p>
</w:hdr>
```

Even page header part:

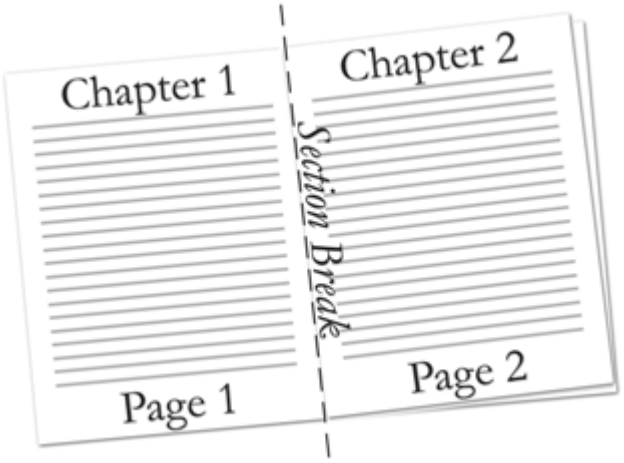
```
<w:hdr>
  <w:p>
    <w:r>
      <w:t>Even</w:t>
    </w:r>
  </w:p>
</w:hdr>
```

Odd page header part:

```
<w:hdr>
  <w:p>
    <w:r>
      <w:t>Odd</w:t>
    </w:r>
  </w:p>
</w:hdr>
```

2.11.4 Multiple Sections

Documents are capable of having multiple sections, where each section can define up to three headers and footers. By default, sections other than the first section inherit the previous header and footer references, unless that section specifies header and footer references.



Consider a two-page, two-section document with only the first section header defined. This document defines one header that is referenced in the first section. The document is represented by the following WordprocessingML:

```
<w:body>
  …
  <w:p>
    <w:pPr>
      <w:sectPr>
        <w:headerReference r:id="rId6" />
        …
      </w:sectPr>
    </w:pPr>
    …
  </w:p>
  …
  <w:sectPr>
    …
  </w:sectPr>
</w:body>
```

The second section does not explicitly reference a header. Instead, the second section inherits the header from the previous section.

2.11.5 Empty Header or Footer

Not specifying a header and footer reference in a section, other than the first section, causes the document to inherit the previous section's header and footer references. In order to declare an empty header or footer, a header or footer reference must be made to a null header or footer relationship, as follows:

```
<Relationship Id="rId2" Type="http:// …/header" Target="null" />
```

The `null` attribute value specifies that the header or footer shall not be inherited from the previous section, and a blank header or footer shall explicitly be used.

2.12 Footnotes and Endnotes

Footnotes and *endnotes* are separate text stories used in documents and books to show the source of borrowed material or to enter explanatory or supplementary information that does not interrupt the normal reading flow of the document.

Footnotes are typically located at the bottom of a page or beneath text being referenced, and *endnotes* are typically placed at the end of a document or at the end of a section. If document has been divided up into one or more sections, each section of a document can contain endnotes.

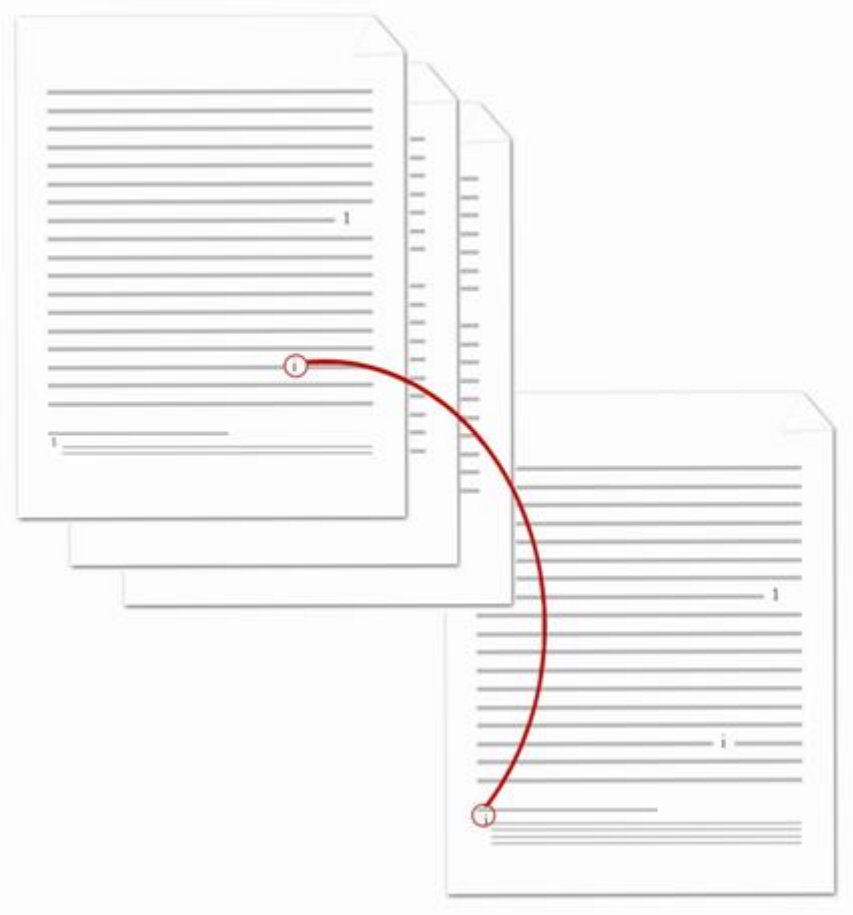
Both footnotes and endnotes consist of two parts:

- A note reference mark in the body text to indicate that additional information is in a footnote or endnote, with a numbering system used for each to tell readers whether to look for the note at the end of the page or the end of the document or section.
- The actual footnote or endnote story content.

Here's an example of a footnote applied to text in a document:

The note reference mark follows the noted text and specifies that there is associated footnote information; the footnote itself is at the bottom of the current page.

Consider the following example of an endnote applied to text in a document:



The note reference mark follows the noted text and specifies that there is associated endnote information; the endnote itself is at the end of the current section.

2.12.1 Footnote Part

Footnote information in a WordprocessingML document is stored in the footnotes part within the package, which is stored via an implicit relationship from the Main Document part or Glossary Document part of relationship type `http://schemas.openxmlformats.org/wordprocessingml/2006/footnotes` and has a content type of `vnd-openxmlformats.officedocument.wordprocessingml-footnotes+xml`.

2.12.2 Endnote Part

Endnote information in a WordprocessingML document is stored in the Endnotes part within the package, which is stored via an implicit relationship from the Main Document part or Glossary Document part of

relationship type `http://schemas.openxmlformats.org/wordprocessingml/2006/endnotes` and has a content type of `vnd-openxmlformats.officedocument.wordprocessingml-endnotes+xml`.

2.12.3 Footnotes and Endnotes

As described above, footnote and endnote information is stored in the corresponding footnotes and endnotes part within the package. The footnotes element below specifies three or more footnotes, each identified by the footnote element, for the document. The endnotes element specifies three or more endnotes, each identified by the endnote element, for the document. Each footnote or endnote element is associated with a unique ID, specified by the attribute id.

Consider three different types of footnotes, each identified by a footnote element, defined in the Footnotes part. The use of each type of footnote is defined in the next subclause:

```
<w:footnotes ...>
  <w:footnote w:type="separator" w:id="0">
    …
  </w:footnote>
  <w:footnote w:type="continuationSeparator" w:id="1">
    …
  </w:footnote>
  <w:footnote w:id="2">
    …
  </w:footnote>
</w:footnotes>
```

Similarly consider three different types of endnotes, each identified by an endnote element, defined in the Endnotes part. The use of each type of endnote is defined in the next subclause:

```
<w:endnotes ...>
  <w:endnote w:type="separator" w:id="0">
    …
  </w:endnote>
  <w:endnote w:type="continuationSeparator" w:id="1">
    …
  </w:endnote>
  <w:endnote w:id="2">
    …
  </w:endnote>
</w:endnotes>
```

Footnotes and endnotes are just another kind of paragraph in WordprocessingML. Within the footnote or endnote element, the footnote or endnote may contain any valid block-level content.

2.12.4 Footnote and Endnote Types

There are four different types of footnotes and endnotes:

- Normal – contain the text of any footnote (or endnote) in the document.
- Separator – define the separator used to separate the footnote (or endnote) from the document text.
- Continuation separator – define the separator used to separate the footnote (or endnote) from the document text when the footnote or endnote is a continuation from the previous page.
- Continuation notice – define the notice text to let readers know that the footnote (or endnote) has continued on the next page.

The attribute type specifies the type of footnote or endnote. Normal footnotes or endnotes are specified by a type of `normal` or by omitting type. In conjunction to a `normal` type, a footnote reference mark, specified by footnoteRef element, or endnote reference mark, specified by endnoteRef element, must be present within the footnote or endnote definition.

Consider the following page in a document, where some text is referenced by a footnote at the end of a page:

The footnote text at the bottom of the page is a normal type footnote represented by the following WordprocessingML:

```
<w:footnote w:id="2">
  <w:p>
    <w:pPr>
      <w:pStyle w:val="FootnoteText" />
    </w:pPr>
```

```
      <w:r>
        <w:rPr>
          <w:rStyle w:val="FootnoteReference" />
        </w:rPr>
        <w:footnoteRef />
      </w:r>
      <w:r>
        <w:t>Cool reference</w:t>
      </w:r>
    </w:p>
  </w:footnote>
```

Not specifying any type attribute in the footnote element defaults to being a `normal` type of footnote. In this example, the footnote has a unique ID of 2. The text of the footnote is contained in the text run. Like any paragraph, footnotes can be associated with a particular style, and, in this example, the paragraph uses the `FootnoteText` paragraph style. Similarly, like any run, footnotes can be associated with a particular style, and, in this example, the run uses the `FootnoteReference` run style.

Separator footnotes or endnotes are specified by `separator`. These types of footnotes or endnotes define the look of the separator used to separate document text from footnotes or endnotes. In conjunction to `separator` type, a footnote or endnote separator reference mark, specified by a separator element must be present within the footnote or endnote definition.

Consider the following page in a document, where some text is referenced by a footnote at the end of a page:

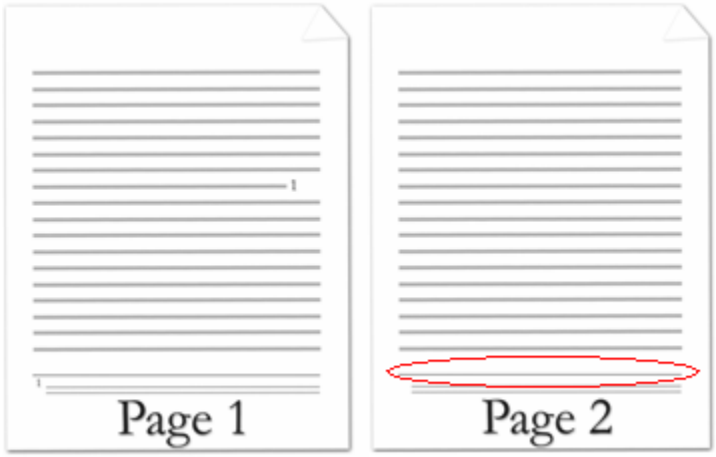
The line separating the document text from the footnote is represented by the following WordprocessingML:

```
<w:footnote w:type="separator" w:id="0">
  <w:p>
    <w:pPr>
      <w:spacing w:after="0" w:line="240" w:lineRule="auto" />
    </w:pPr>
    <w:r>
      <w:separator />
    </w:r>
  </w:p>
</w:footnote>
```

In this example, the footnote has a unique ID of `0`. The vertical spacing after the line separator is 0 twentieths of a point. The vertical spacing between the line separator and text is 240 twentieths of a point.

Continuation separator footnotes or endnotes are specified by `continuationSeparator`. These types of footnotes or endnotes define the look of the separator used to separate document text from footnotes or endnotes when the footnote or endnote continues the next page. In conjunction to a `continuationSeparator` type, a footnote or endnote continuation separator reference mark, specified by continuationSeparator element must be present within the footnote or endnote definition.

Consider the following two pages in a document, where some text is referenced by a footnote that extends to the next page:



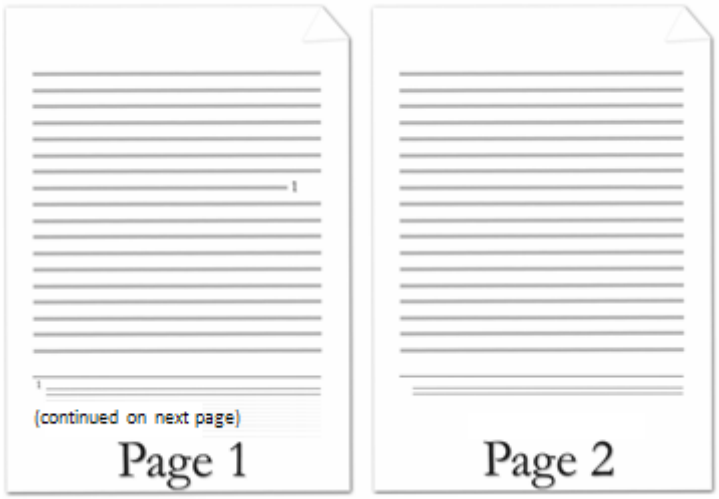
The line separating the document text from the footnote that is continued on the next page (circled in red in the image above) is the continuation separator footnote, and is represented by the following WordprocessingML:

```
<w:footnote w:type="continuationSeparator" w:id="1">
  <w:p >
    <w:pPr>
      <w:spacing w:after="0" w:line="240" w:lineRule="auto" />
    </w:pPr>
    <w:r>
      <w:continuationSeparator />
    </w:r>
  </w:p>
</w:footnote>
```

In this example, the footnote has a unique ID of 1. The vertical spacing after the line separator is 0 twentieths of a point. The vertical spacing between the line separator and text is 240 twentieths of a point.

Continuation notice footnotes or endnotes are specified by `continuationNotice`. These types of footnotes or endnotes specify the text to let readers know that the footnote or endnote is continued on the next page.

Consider the following two pages in a document, where some text is referenced by a footnote that extends to the next page. A continuation notice is given to readers to indicate that the footnote extends to the next page:



The continuation notice text is at the bottom of the footnote indicating that the footnote is continued to the next page (which reads `continued on next page above`) and is represented by the following WordprocessingML:

```
<w:footnote w:type="continuationNotice" w:id="3">
  <w:p >
    <w:pPr>
      <w:spacing w:after="0" w:line="240" w:lineRule="auto" />
    </w:pPr>
```

```
        <w:r>
          <w:t>(continued on next page)</w:t>
        </w:r>
      </w:p>
    </w:footnote>
```

In this example, the footnote has a unique ID of 3. The text that shows up after the footnote text is `(continued on next page)`.

2.12.5 Footnote and Endnote Reference

Once footnote or endnote information is defined in the footnotes or endnotes part, this information must be associated with document text within the document in order to display the footnotes or endnotes. Each footnote or endnote is identified by a unique ID that references footnote or endnote definitions specified in the footnotes or endnotes part. Footnote or endnote references are identified by the footnoteReference or endnoteReference element within the text run's element (the r element). The footnoteReference or endnoteReference element points to the footnote or endnote ID defined in the footnotes or endnotes part.

Consider the following one-page document, where some text is referenced by a footnote at the end of the document page:

The footnote references text and is represented by the following WordprocessingML:

```
    <w:p>
      <w:r>
        <w:t>Some referenced text</w:t>
      </w:r>
```

```
    <w:r>
      <w:rPr>
        <w:rStyle w:val="FootnoteReference" />
      </w:rPr>
      <w:footnoteReference w:id="2" />
    </w:r>
  </w:p>
```

The footnote references the footnote in the footnotes part with ID equals to 2. Like any run, footnotes can be associated with a particular style, and, in this example, the run uses the `FootnoteReference` run style. The style of the footnote defines the look and numbering of the footnote.

2.13 Glossary Document

The introduction to a WordprocessingML document formally introduced the concept of stories, individual ranges of a word-processing document containing block-level content like paragraphs and tables. Some examples of stories in a WordprocessingML document include the following: the main document, headers, footers, comments, footnotes, and endnotes.

At that time, a story was defined by two characteristics:

- It is a unique region containing block-level content
- All document stories shared the same set of properties (e.g., style definitions, numbering definitions, and settings)

The glossary document, although it follows the first rule, actually defies the second.

Within a WordprocessingML file, the *glossary document* is a supplemental storage location for additional document content which shall travel with the document, but which shall not be displayed for printed as part of the main document until it is explicitly added to that document by deliberate action.

The glossary document shall also be afforded a separate instance of all of the relationships that are provided on the main document part - this means that the glossary document shall have its own style definitions, numbering definitions, comments, headers, footers, etc. within the WordprocessingML document.

Consider a document that shall include ten optional clauses that may be inserted through a user interface. It is clearly not desirable to have these ten clauses appear in the main document story's contents before they are explicitly inserted, therefore each of them may be stored in the glossary document and inserted via the user interface as needed.

Within the glossary document, each distinct region of document content is referred to as a *glossary document entry*, and is defined via the docPart element. These document parts may contain any block-level WordprocessingML element, and may also have a set of classifications and behaviors applied to them via the glossary document entry's properties.

Consider the following definition for the contents of a glossary document part within a WordprocessingML document:

```
<w:glossaryDocument>
  <w:docParts>
    <w:docPart>
      <w:docPartPr>
        …
      </w:docPartPr>
      <w:docPartBody>
        <w:p>
          <w:r>
            <w:t>Sample entry.</w:t>
          </w:r>
        </w:p>
      </w:docPartBody>
    </w:docPart>
    <w:docPart>
      …
    </w:docPart>
  </w:docParts>
</w:glossaryDocument>
```

The glossaryDocument element defines the contents of the glossary document part. Within the glossary document, each docPart element contains the definition for one glossary document entry: in this case, there are two entries in the glossary document, the first of which contains a single paragraph with a single run of text.

Each glossary document entry consists of two components:

- The entry's properties, specified using the docPartPr element
- The entry's contents, specified using the docPartBody element

The first specifies information about the entry (e.g., its classification) for when it is inserted, the latter stores the block level content which constitutes the entry.

2.14 Annotations

2.14.1 Introduction

An *annotation* is one of various kinds of supplementary markup, which may be stored inside or around a region of text within the document's contents. The kinds of supplementary information stored within a document can include comments (§2.14.5), revisions (§2.14.7), spelling and/or grammatical errors (§2.14.10), bookmark information (§2.14.8), and optional editing permissions (§2.14.9).

Within a document's contents, annotations are stored in one of three different forms:

- Inline
- Cross-Structure
- Properties

These three forms are needed in order to maintain compatibility with both the legacy annotations functionality of current word-processing applications and the requirements of an XML-based format (i.e., well-formedness of the resulting XML markup). These three forms are referenced within the individual annotation types described in subclauses §2.14.2 through §2.14.4.

2.14.2 Inline Annotations

An *inline annotation* is a form of annotation that does not require special handling in order to maintain the XML well-formedness requirements of the resulting WordprocessingML output. In these cases, a single XML element shall encapsulate the entire contents of the document content which is being annotated.

Consider the following WordprocessingML markup for a paragraph that reads `The quick brown fox jumps over the jet lagged dog.`, where `jet lagged` replaced the previous text `lazy` when the editing application was tracking revisions:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">The quick brown fox jumps over the </w:t>
  </w:r>
  <w:del … >
    <w:r>
      <w:delText>lazy</w:delText>
    </w:r>
  </w:del>
  <w:ins … >
    <w:r>
      <w:t>jet lagged</w:t>
    </w:r>
  </w:ins>
  <w:r>
    <w:t xml:space="preserve"> dog.</w:t>
  </w:r>
</w:p>
```

The del and ins elements each fully encapsulate the extent of their respective annotations (a marked deletion and insertion, respectively), as they are inline annotations.

2.14.3 Cross-Structure Annotations

A *cross-structure annotation* is a form of annotation that can span portions of WordprocessingML markup. (Cross-structure annotations may span parts of multiple paragraphs, one half of a custom XML markup element's contents, and so on.) In these cases, the annotation's region is delimited by two elements, a start

element and an end element, which mark the start and end points of the annotated content, respectively, but do not contain it. Matching start and end markers have the same id attribute value.

Consider the following WordprocessingML markup for two paragraphs, each reading `Example Text`, where a bookmark has been added spanning the second word in paragraph one, and the first word in paragraph two:

```
<w:p>
  <w:r>
    <w:t>Example</w:t>
  </w:r>
  <w:bookmarkStart w:id="0" w:name="sampleBookmark" />
  <w:r>
    <w:t xml:space="preserve"> text.</w:t>
  </w:r>
</w:p>
<w:p>
  <w:r>
    <w:t>Example</w:t>
  </w:r>
  <w:bookmarkEnd w:id="0" />
  <w:r>
    <w:t xml:space="preserve"> text.</w:t>
  </w:r>
</w:p>
```

The bookmarkStart and bookmarkEnd elements specify the location where the bookmark starts and ends, but cannot contain that bookmark because it spans parts of two paragraphs. They are part of one group because the id attribute value specifies `0` for both.

2.14.4 Property Annotations

A *property annotation* is a form of annotation that is stored as a property on an object (Property annotations may appear on paragraph properties, run properties, table rows, and so on.) In these cases, the annotation's semantics are defined by the property, as they can affect content and/or formatting.

Consider the following WordprocessingML markup for a paragraph reading `Example Text`, where the first word had the bold property applied when the editing application was tracking revisions:

```
<w:p>
  <w:r>
    <w:rPr>
      <w:b/>
      <w:rPrChange … >
        <w:rPr/>
      </w:rPrChange>
    </w:rPr>
    <w:t>Example</w:t>
  </w:r>
  <w:r>
    <w:t xml:space="preserve"> text.</w:t>
  </w:r>
</w:p>
```

The rPrChange element contains the set of previously applied revision properties associated with a particular author at a particular time. It is stored itself as a property on the parent run which was modified.

2.14.5 Comments

A *comment* is an annotation that is anchored to a region of document content, but which contains an arbitrary amount of block-level content stored in its own separate document story. Within a WordprocessingML document, comments are stored in a separate comments part within the document package.

A comment in a WordprocessingML document is divided into two components:

- The comment anchor (the text to which the comment applies)
- The comment content (the contents of the comment)

The *comment anchor* is the cross-structure annotation that defines the region of text on which the comment in anchored. The *comment content* is the text of the comment.

Consider a paragraph in a WordprocessingML document whose second word is annotated with a comment:

The first component to this comment is the document content, which defines the extents of the comment and references the specific comment in the comments part:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">Some </w:t>
  </w:r>
```

```
    <w:commentRangeStart w:id="0" />
    <w:r>
      <w:t>text.</w:t>
    </w:r>
    <w:commentRangeEnd w:id="0" />
    <w:r>
      <w:commentReference w:id="0" />
    </w:r>
  </w:p>
```

The commentRangeStart and commentRangeEnd elements delimit the run content to which the comment with an id of `0` applies (in this case, the single run of text). The commentReference element that follows links the preceding run content with a comment in the comments part having an id of `0`. Without all three of these elements, the range and comment cannot be linked (although the first two elements are optional, in which case the comment shall be anchored at the comment reference mark)

The second component to this comment is the comment content, which defines the text in the comment:

```
  <w:comment w:id="0" w:author="Joe Smith"
    w:date="2006-04-06T13:50:00Z" w:initials="User">
    <w:p>
      <w:pPr>
        <w:pStyle w:val="CommentText" />
      </w:pPr>
      <w:r>
        <w:rPr>
          <w:rStyle w:val="CommentReference" />
        </w:rPr>
        <w:annotationRef />
      </w:r>
      <w:r>
        <w:t>comment</w:t>
      </w:r>
    </w:p>
  </w:comment>
```

In this example, the comment specifies that it was inserted by author `Joe Smith` with the initials `User` via the author and date attributes. It is linked to the run content via the id attribute, which matches the value of `0` specified using the commentReference element above. The block-level content of the comment specifies that its text is comment and the style of the comment content is based off of the character style with the name `CommentReference`.

2.14.6 Comments Part

Comment information in a WordprocessingML document is stored in the Comments part within the package, which is stored via an implicit relationship from the Main Document or Glossary Document part of relationship type `http://…/comments` and has a content type of `vnd-openxmlformats.officedocument.wordprocessingml-comments+xml`.

2.14.7 Revisions

A *revision* provides a mechanism for storing information about the evolution of the document (i.e., the set of modifications made to a document by one of more authors). When an application adds revisions to the content of a WordprocessingML document, depending on the revision type they are specifying this by storing either:

- The current state of the document (a deletion stores the current state of the text as deleted, and implies that its original state was the content that used to exist)
- The initial state of the document (a run's initial properties are explicitly stored in a previous run properties block, as the current run properties are always those that are the child of the rPr element

A revision consists of two required pieces of information:

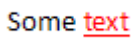
- The revision type (specified via the name of the revision element)
- A unique revision identifier (used to uniquely identify revisions)

As well as optional information:

- The author of the revision
- The date and time of the revision

A revision is stored using the inline annotation format or the property annotation format.

Consider a paragraph of text in a WordprocessingML document in which one word has been inserted, as follows:



This paragraph has the word `text` marked inserted as a revision, and is represented as the following WordprocessingML:

```
<w:p>
  <w:r>
    <w:t>Some</w:t>
  </w:r>
  <w:ins w:id="0" w:author="Joe Smith" w:date="2006-03-31T12:50:00Z">
    <w:r>
      <w:t>text</w:t>
```

```
      </w:r>
    </w:ins>
  </w:p>
```

The ins element contains all of the content that shall be treated as revision marked as inserted (i.e., the word `text`).

This means that it contains both required pieces of information: the revision type, specified by the name of the revision element (ins); and a unique revision identifier of `0`.

The element also stores the optional information about the revision: the word `text` was inserted by `Joe Smith` on March 31, 2006 at 12:50 pm.

Within a WordprocessingML document, the following types of revisions can be used to track the changes to a document (each annotation's form in parentheses):

- Insertions (inline annotations for run content, property annotations for tables and paragraphs)
- Deletions (inline annotations for run content, property annotations for tables and paragraphs)
- Moves (inline annotations)
- Changes to run/paragraph/table/numbering/section properties (property annotations)
- Changes to custom XML markup (property annotations)

2.14.8 Bookmarks

A *bookmark* refers to an arbitrary region of content that is bounded and has a unique name associated with it.

Because bookmarks are a legacy word-processing function that predates the concepts of XML and well-formedness, they can start and end at any location within a document's contents and, therefore, must use the cross-structure annotation format described in §2.14.3.

Consider the following WordprocessingML markup for two paragraphs, each reading `Example Text`, where a bookmark has been added spanning the second word in paragraph one and the first word in paragraph two:

```
  <w:p>
    <w:r>
      <w:t>Example</w:t>
    </w:r>
    <w:bookmarkStart w:id="0" w:name="sampleBookmark" />
    <w:r>
      <w:t xml:space="preserve"> text.</w:t>
    </w:r>
  </w:p>
```

```
<w:p>
  <w:r>
    <w:t>Example</w:t>
  </w:r>
  <w:bookmarkEnd w:id="0" />
  <w:r>
    <w:t xml:space="preserve"> text.</w:t>
  </w:r>
</w:p>
```

The bookmarkStart and bookmarkEnd elements specify the location where the bookmark starts and ends, but cannot contain it using a single tag because it spans parts of two paragraphs. However, the two tags are part of one group because the id attribute value specifies `0` for both.

2.14.9 Range Permissions

A *range permission* refers to a special type of bookmark used to control which subset(s) of users may edit a particular region of a document. Range permissions specify the user, or set of users, that are allowed to edit all content between them whenever the document protection specified by the documentProtection element is enabled and set to `readOnly` or `comments`.

Like bookmarks, range permissions are a legacy word-processing function that predates the concepts of XML and well-formedness, so they can start and end at any location within a document's contents and, therefore, must use the cross-structure annotation format described in §2.14.3.

Consider the following WordprocessingML markup for a single paragraph, where a range permission has been added spanning the words `range permission`:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">This is a </w:t>
  </w:r>
  <w:permStart w:id="0" w:edGrp="everyone"/>
  <w:r>
    <w:t>range permission</w:t>
  </w:r>
  <w:permEnd w:id="0"/>
  <w:r>
    <w:t>.</w:t>
  </w:r>
</w:p>
```

The permStart and permEnd elements specify the location where the range permission starts and ends. The two tags are part of one group because the id attribute value specifies `0` for both.

If document protection was enabled, then no content in this document shall be editable except for this range permission, which is editable by all users that open the document (specified using an editor group of everyone).

2.14.10 Spelling and Grammar

A *spelling and grammar error* is an annotation used to specify the locations of an existing spelling and/or grammatical error within the contents of a document. Spelling and grammar errors use the cross-structure annotation format.

Rationale: When a WordprocessingML document is saved, applications may choose to save currently flagged spelling and grammar errors, for two reasons:

- In order to increase the performance subsequent loads of the document (as those load operations can rely on the persisted proofing state of the document)
- In order to store words which shall not be marked as proofing errors regardless of how they would normally be flagged by the proofing tools engine (i.e., to store spelling and grammar exceptions).

Consider the following paragraph consisting of two misspelled words, where the second word has been explicitly flagged as not being a spelling error. This paragraph would consist of the following WordprocessingML markup:

```
<w:p>
  <w:proofErr w:val="spellStart"/>
  <w:r>
    <w:t>erqwt</w:t>
  </w:r>
  <w:proofErr w:val="spellEnd"/>
  <w:r>
    <w:t xml:space="preserve"> werewr</w:t>
  </w:r>
</w:p>
```

The proofErr elements, with a val attribute value of `spellStart` and `spellEnd`, respectively, delimit the start and end of the content in this paragraph that is stored as a spelling error. Since the second word is not included in that range, it is not stored as a spelling error.

2.15 Mail Merge

Mail merge refers to a process by which a WordprocessingML document is connected to and populated with external data by a conforming hosting application and/or data source access application. A WordprocessingML document that contains the necessary data to connect to an external data source during a Mail Merge is known as a source document. In other words, a source document is a WordprocessingML document containing the elements and attributes necessary to enable the document to connect to an external data source, but not yet merged with any data.

Applications leverage source documents to generate new documents containing the static content contained in the merged document as well as data from the specified external data source. The documents that result from importing external data into a source document are known as merged documents. How source documents and merged documents are specified is explained in the following sections.

2.15.1 Mail Merge, WordprocessingML, and Hosting Applications

The two key parts of the mail merge process are:

1. Connecting to an external data source
2. Populating mail merge fields with external data

It is important to note that aspects of the mail merge process outside of connecting to an external data source and populating mail merge fields with external data, are at the discretion of the hosting application.

As an additional example, WordprocessingML provides an element to be used as a flag by hosting applications to specify action to be taken on the merged documents that are generated by a mail merge. In other words, performing actions such as:

- creating a new document for each merged document
- generating and sending emails containing merged document
- printing merged documents

may be specified through WordprocessingML, but what if any specific action is taken on merged documents is determined by the application.

2.15.2 Connecting Documents to an External Data Source

As mentioned, a source document is the single WordprocessingML document that contains the data necessary to be connected to an external data source by a conforming hosting application and/or data source access application. The presence and parameters of this connection are specified within the mailMerge element. This element enables WordprocessingML documents to be connected to an external data source by specifying the following data:

- Where the external data is located (e.g., file path)
- What type of data the external data source contains (e.g., database and spreadsheet)
- How the data will be accessed

Consider a document containing static WordprocessingML constructs such as paragraphs in addition to two WordprocessingML mail merge fields calling for `Courtesy Title` and `Last Name` data.

Dear {MERGEFIELD "Courtesy Title" \m} {MERGEFIELD "Last Name" \m},

Sample text. Sample text.

Sincerely,

If the following WordprocessingML was added to this document, this document would become a source document rather than just a standard WordprocessingML document, as the mailMerge element specifies the elements and attributes necessary to enable the hosting applications and/or data source access applications to connect the document to an external data source.

```
<w:mailMerge>
  …
  <w:dataType w:val="database" />
  <w:query w:val="SELECT * FROM Table1" />
  <w:dataSource r:id="rId1" />
  …
</w:mailMerge>
```

Here, the dataType and dataSource elements specify that the given document shall be connected to the external data source referenced by the r:id attribute's value of `rId1`. While connected to the external data source, the merged document together with the hosting application and/or data source access application may extract data from the external data source as specified by the connectString and query elements.

2.15.3 Populating Merged Documents with External Data

Before the hosting application can populate merged documents with external data, mail merge fields must be inserted into the *merged* document and mapped to the external data. How external data is mapped to given mail merge fields is determined by the WordprocessingML element fieldMapData.

Consider the example merged document from the previous example which contained the two mail merge fields calling for `Courtesy Title` and `Last Name`. The WordprocessingML below demonstrates how mapping of the external data to the merged document's mail merge fields occurs:

```
<w:fieldMapData>
  <w:type w:val="dbColumn" />
  <w:name w:val="Customer Title" />
  <w:mappedName w:val="Courtesy Title" />
  <w:column w:val="9" />
</w:fieldMapData>
<w:fieldMapData>
  <w:type w:val="dbColumn" />
  <w:name w:val="Customer Last Name" />
  <w:mappedName w:val="Last Name" />
  <w:column w:val="10" />
</w:fieldMapData>
```

Within the first fieldMapData element, the child elements column, name, type, and mappedName specify that the data contained within tenth column titled 'Customer Title', in the specified external database, is to be mapped to the mail merge field calling for 'Courtesy Title' data, respectively. Within the second fieldMapData element, the child elements column, name, type, and mappedName specify that the data contained within eleventh column in the specified external database is to be mapped to the merge field titled `Customer Last Name` or the predefined merge field name `Last Name`.

Once a merged document's mail merge fields have been mapped to external data, the hosting application and/or data source access application may populate the respective fields with applicable external data.

Consider a conforming hosting application and/or data source access application that wishes to populate the mail merge fields within the merged document from the previous example with applicable external data. In addition, consider that the specified external data source contains two records--one for Mr. John Doe and one for Ms. Jane Smith. With external data from the `Customer Title` column mapped to the Mail Merge field calling for `Courtesy Title` data, and the `Customer Last Name` column mapped to the Mail Merge field calling for `Last Name` data to populate the fields within this merged document with external data.

The mail merge process will then run through the specified external database and populate the mail merge fields with the data from the `Customer Title` and `Customer Last Name` columns in the specified database, and generate two of merged documents containing the specified external data as well as the static contents of the source document (illustrated in the table below):

Source Document	Merged document populated with first external data source entry	Merged document populated with second external data source entry

Source Document	Merged document populated with first external data source entry	Merged document populated with second external data source entry
Dear{MERGEFIELD "Courtesy Title" \m} {MERGEFIELD "Last Name" \m}, Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sincerely,	Dear Mr. Doe: Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sincerely,	Dear Ms. Smith: Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sincerely,

2.16 Settings

A *setting* specifies a stored preference that shall be used when processing the contents of the document. In other words, settings refer to specified behaviors that shall be applied to WordprocessingML documents on a document by document basis. Just like paragraphs and text runs have properties specified that apply to their contents, entire WordprocessingML documents leverage settings to specify properties and behaviors that apply to the entire document.

These settings are typically divided into three categories:

- *Document Settings* — Settings that influence the appearance and behavior of the current document, as well as storing document-level state.
- *Compatibility Settings* — Settings that tell applications to perform behaviors which are designed to maintain visual output of previous word-processing applications.
- *Web Settings* — Settings that affect how a document shall be handled when it is saved as HTML.

2.16.1 Document Settings

A *document setting* specifies a document-level property that affects the handling of a given document, and influences the appearance and behavior of the current document, as well as the stored document-level state. All document settings are found in the Document Settings part.

Consider a document in which the document setting doNotHyphenateCaps is applied. As a document setting this element specifies whether words comprised of all capitalized letters shall be hyphenated or not throughout the given document.. Specifically, if words in ALL CAPITAL LETTERS shall not be hyphenated, this requirement would be specified by adding the following WordprocessingML to the settings part:

```
<w:doNotHyphenateCaps w:val="true"/>
```

Specifying that words comprised of ALL CAPITAL LETTERS shall be hyphenated, as illustrated below:

THIS IS A HYPHENATION EXAMPLE. THIS IS A SHORT HYPHENATION EXAMPLE. THIS IS A SHORT HYPHENATION EXAMPLE.

If this element is omitted, then words in ALL CAPITAL LETTERS shall be hyphenated when the document is hyphenated, as illustrated below:

THIS IS A HYPHENATION EXAMPLE. THIS IS A SHORT HYPHENATION EXAMPLE. THIS IS A SHORT HYPHE-NATION EXAMPLE.

2.16.2 Compatibility Settings

A *compatibility setting* is an optional setting used to mimic behavior of documents created in earlier word-processing applications. It is recommended that new WordprocessingML documents contain no compatibility settings. If compatibility settings are needed, they are stored in the Document Settings part (§2.16.1).

Consider a document in which the compatibility setting ww11IndentRules is applied. As a compatibility setting, this element specifies an indentation behavior to be applied throughout the given document to preserve visual fidelity with an earlier word processing application. Specifically, if the indentation applied to numbering when positioned next to a wrapped object shall not be suppressed, this requirement would be specified by adding the following WordprocessingML to the settings part

```
<w:compat>
  <w:ww11IndentRules />
</w:compat>
```

Specifying that indentation applied to numbering when positioned next to a wrapped object shall not be suppressed, as illustrated below:

1. Example

2. Example
3. Example
4. Example
5. Example

If this element is omitted, then indentation applied to numbering when positioned next to a wrapped object shall be suppressed, as illustrated below:

1. Example

2. Example
3. Example
4. Example
5. Example

2.16.3 Web Settings

A *web setting* is a setting used to specify a document-level property that is applicable when saving a web page as a WordprocessingML document, or when saving a WordprocessingML document as a webpage. Thus, if a given WordprocessingML document was not created from a web page, and will never become a web page, no web settings are needed within the document. If they are needed, web settings are stored in the Web Settings part.

Consider a document in which the web setting allowPNG is applied. As a web setting this element specifies if the PNG graphics format will be used for persisting images when saving the document as a web page. Specifically, if the PNG graphics format will be used when saving a document as a web page, this requirement would be specified by adding the following WordprocessingML to the settings part:

```
<w:webSettings>
  <w:allowPNG />
</w:webSettings>
```

If this element is omitted, then the JPEG graphics format will be used for persisting images when saving the document as a web page.

2.17 Fields and Hyperlinks

2.17.1 Fields

Most text in a word processing document is static; that is, unless it is directly changed as the result of editing, its contents remain the same, no matter how the rest of the document might change. However, certain useful pieces of information can change value over the life of a document. Consider the case of a reference to a page number, as in "For more information on this topic, see page 56." Clearly, hard coding the page number as 56 means that that number will need to be manually replaced as the document's size or layout is changed. Even a simple change to any margin, line spacing, or font size can invalid such references.

Fields provide a mechanism for placeholders, such as page reference numbers, that can be added to a document such that those placeholders are replaced by their corresponding values when the document is rendered for display or print. Other applications for fields include, but are not limited to, automatic numbering of tables and figures, document creation and current date and time, document author information, and the computation of totals for a table column.

A *field* is a set of codes that instructs a WordprocessingML consumer to insert text, graphics, page numbers, and other material into a document automatically. (The DATE field causes the current date to be inserted.) The text or graphics inserted into a document when a consumer carries out a field's codes is referred to as the *field result* for that field. The act of carrying out a field's codes is referred to as a *field update*. As to how or when any field is updated is outside the scope of this standard.

2.17.2 Hyperlinks

As well as allowing for dynamic run content using fields, a WordprocessingML document may contain one or more *hyperlinks*, which allow for the linking of two disparate regions of WordprocessingML content (analogous to hyperlinks in HTML pages). WordprocessingML hyperlinks can be any of the following:

- Intradocument: A hyperlink can target any bookmark contained within the current WordprocessingML document.
- Interdocument: A hyperlink can target another WordprocessingML package, as well as specify a bookmark within that package.
- Other destinations: A hyperlink can target any other valid URI location.

2.18 Miscellaneous Topics

2.18.1 Text Boxes

All VML-based drawing objects (except for connectors) support the addition of rich WordprocessingML content within their extents. When WordprocessingML contents have been added to a VML drawing object, the resulting text is contained within a *text box*.

When WordprocessingML content is contained within a text box, it is represented within the object by specifying the VML textbox element, which contains within it a single txbxContent element that contains all of the desired WordprocessingML content. Text box content cannot contain references to other document stories, nor can it contain other txbxContent elements. That is, nested shapes cannot have rich content.

2.18.2 Subdocuments

Within a WordprocessingML document, it is sometimes necessary to break a large document into two or more separate WordprocessingML document files, allowing each of these files to be distributed, edited, and handled independently.

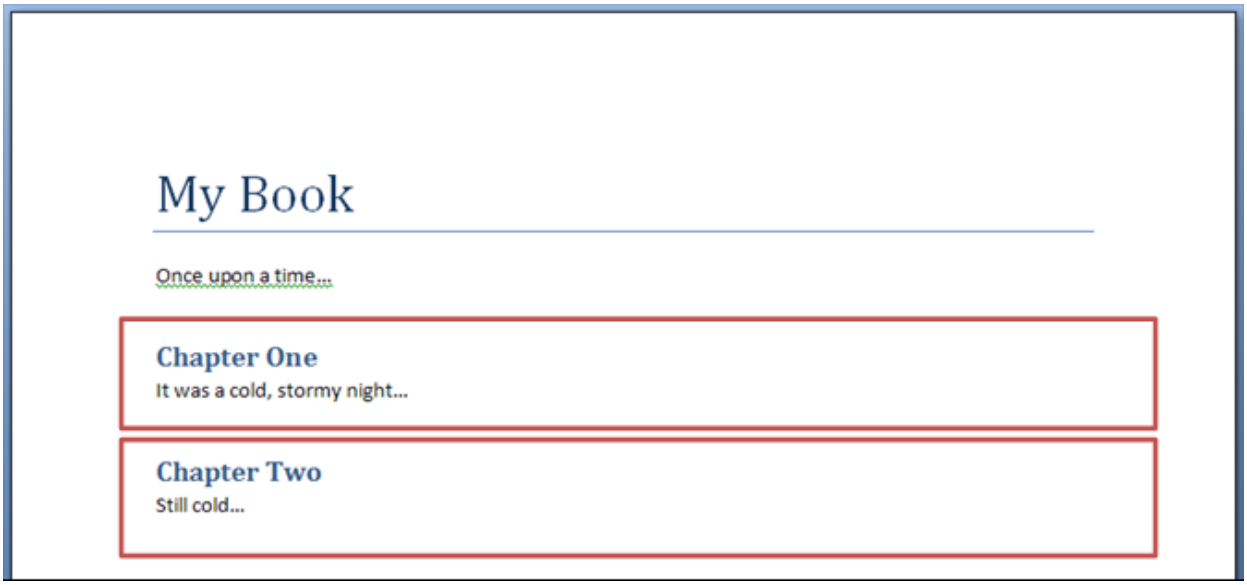
A book might consist of five chapters, each edited by a separate author. The editor for the book would therefore desire to create six WordprocessingML documents - one for each author to work on their chapter, and a main document which collates the content of the five chapters appropriately.

When a WordprocessingML document is composed of other WordprocessingML documents in this way, the resulting documents are a master document and its subdocuments.

- A *master document* is a document which incorporates one or more subdocuments (as well as optional WordprocessingML content) to create a larger document
- A *subdocument* is a WordprocessingML document—there is no specific information in a document which classifies it as such

Consider a WordprocessingML document, which is being used to write a book:

To allow this document to be written by multiple authors, each chapter in the book is placed in a separate file (the sections highlighted in red below):



The result is three WordprocessingML documents:

- A master document (containing the title of the book, the first paragraph, and references to the subdocuments for each chapter)
- Two subdocuments (one for each chapter)

2.18.3 Importing External Content

When generating WordprocessingML documents, it is sometimes necessary to include existing document content (henceforth called *external content*) within the document. External content in a document is typically

included because it was stored in a format other than the WordprocessingML format defined by this Office Open XML specification.

In order to facilitate the inclusion of such content without requiring its conversion as a prerequisite to its inclusion in a document, WordprocessingML includes the facility for applications to implement the import of external content in any format as part of a WordprocessingML document. This functionality, called *external content import*, allows the inclusion of content of an arbitrary content type within the WordprocessingML package, which can then be opened and merged into the main document when the package is consumed by applications which understand that content type.

Consider a WordprocessingML document which is being created based on the following existing HTML content:

```
<html … >
  <body style="margin-left:200px;margin-top:50px">
    <p>Paragraph one.</p>
    <blockquote style="border:5px solid #00FFFF">Paragraph in a
blockquote.</blockquote>
    <p>Paragraph two.</p>
  </body>
</html>
```

This content can be converted to its WordprocessingML equivalents using the XML syntax defined by this Office Open XML specification, or a more basic tool can use the external content import to include the HTML document within a WordprocessingML package, allowing a subsequent consumer of that content to import the resulting HTML. When the resulting WordprocessingML package is opened, the HTML document it could be read (if it is an alternate format understood by the consuming application) and migrated into the appropriate location in the main WordprocessingML document.

2.18.4 Roundtripping Alternate Content

Office Open XML defines a mechanism for the storage of content which is not defined by this Office Open XML specification, for example extensions developed by future software applications which leverage the Open XML formats. This mechanism allows for the storage of a series of alternative representations of content, of which the consuming application may use the first alternative whose requirements are met.

Consider an application which creates a new paragraph property intended to make the colors of its text change randomly when it is displayed. This functionality is not defined in this Office Open XML specification, and so the application might choose to create an alternative representation setting a different manual color on each character for clients which do not understand this extension using an AlternateContent block as follows:

```
<ve:AlternateContent xmlns:ve="…">
  <ve:Choice Requires="colors" xmlns:colors="urn:randomTextColors">
    <w:p>
      <w:pPr>
        <colors:random colors:val="true" />
      </w:pPr>
      <w:r>
        <w:t>Random colors!</w:t>
      </w:r>
    </w:p>
  </ve:Choice>
  <ve:Fallback>
    <w:p>
      <w:r>
        <w:rPr>
          <w:color w:val="FF0000" />
        </w:rPr>
        <w:t>R</w:t>
      </w:r>
      <w:r>
        <w:rPr>
          <w:color w:val="00FF00" />
        </w:rPr>
        <w:t>a</w:t>
      </w:r>
      </w:r>
      …
    </w:p>
  </ve:Fallback>
</ve:AlternateContent>
```

The Choice element that requires the new color extensions uses the random element in its namespace, and the Fallback element allows clients that do not support this namespace to see an appropriate alternative representation.

These alternate content blocks may occur at any location within a WordprocessingML document, and applications shall handle and process them appropriately (taking the appropriate choice).

However, WordprocessingML does not explicitly define a set of locations where applications shall attempt to store and roundtrip all non-taken choices whenever possible.

If an application does not understand the colors extension, the resulting file (if alternate choices are to be preserved would appear as follows:

```
<ve:AlternateContent xmlns:ve="…">
  <ve:Choice Requires="colors" xmlns:colors="urn:randomTextColors">
    …
  </ve:Choice>
  <ve:Fallback>
    …
  </ve:Fallback>
</ve:AlternateContent>
```

The file would then appear as follows after the choice is processed:

```
<w:p>
  <w:r>
    <w:rPr>
      <w:color w:val="FF0000" />
    </w:rPr>
    <w:t>R</w:t>
  </w:r>
  <w:r>
    <w:rPr>
      <w:color w:val="00FF00" />
    </w:rPr>
    <w:t>a</w:t>
  </w:r>
  …
</w:p>
```

End of informative text.

3. Introduction to SpreadsheetML

This clause is informative.

This clause contains a detailed introduction to the structure of a SpreadsheetML document.

3.1 Workbook

3.1.1 Overview

A *workbook* is composed of book-level properties and a collection of one or more *sheet*s. The sheets are the central working surface for a spreadsheet application. The workbook part and corresponding properties comprise data used to set application- and workbook-level operational state. The workbook also serves to bind all the sheets and child objects into an organized single file. The workbook properties include information about what application last saved the file, where and how the windows of the workbook were positioned, and an enumeration of the worksheets in the workbook.

3.1.2 Minimum Workbook Scenario

For the sake of simplicity, it is important to minimize the required set of workbook properties that must be present to compose a valid workbook. The smallest possible (blank) workbook must contain the following:

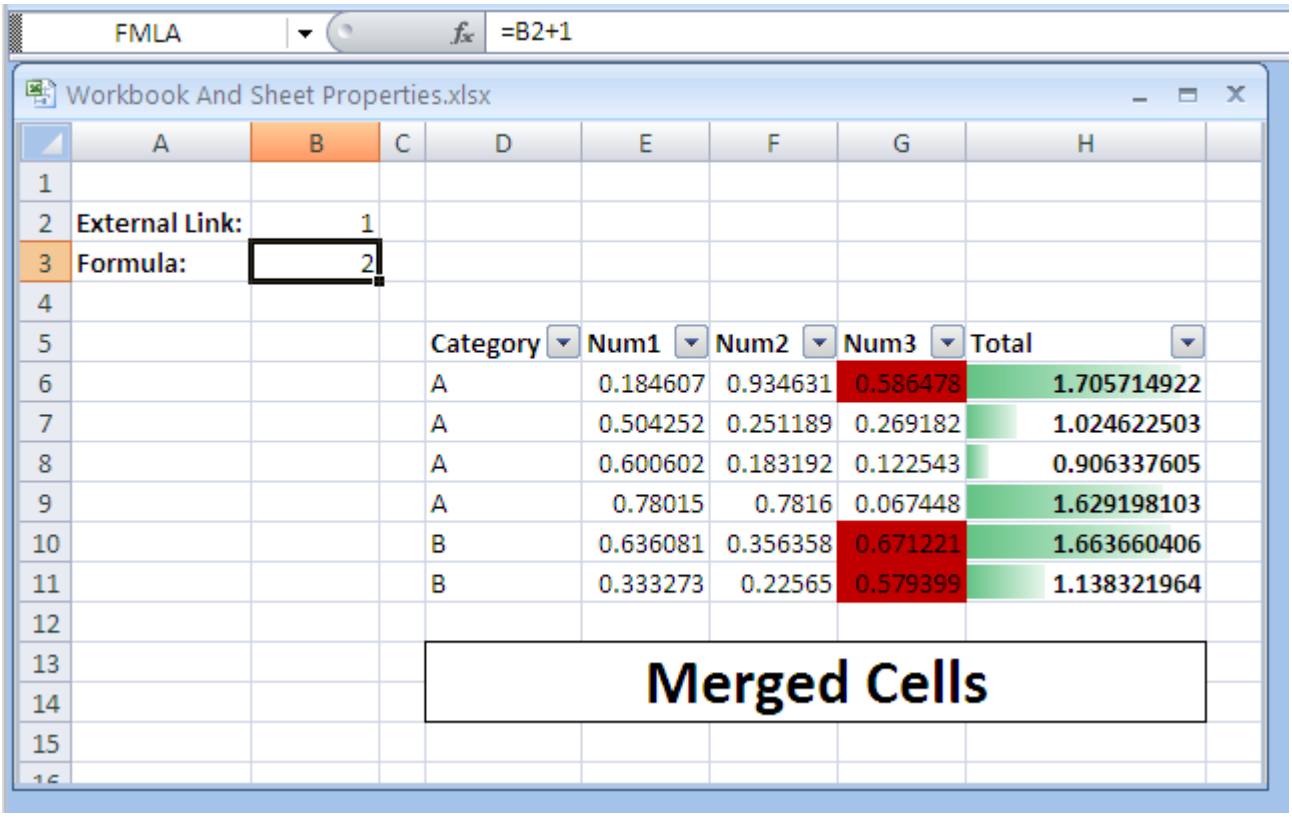
- A single sheet
- A sheet ID
- A relationship Id that points to the location of the sheet definition

For example:

```
<workbook>
  <sheets>
    <sheet name="Sheet1" sheetId="1" r:id="rId1"/>
  </sheets>
</workbook>
```

3.1.3 Example Workbook Properties

Consider the following graphical representation of a workbook:



The above example will have the following workbook properties definition:

```
<workbook>
  <fileVersion lastEdited="4" lowestEdited="4" rupBuild="3814"/>
  <workbookPr backupFile="1" saveExternalLinkValues="0" updateLinks="never"/>
  <calcPr calcId="122211" calcMode="manual" iterate="1"/>
  <bookViews>
    <workbookView showHorizontalScroll="0" showVerticalScroll="0"
      showSheetTabs="0" xWindow="45" yWindow="15" windowWidth="9420"
      windowHeight="5460" tabRatio="701"/>
  </bookViews>
  <sheets>
    <sheet name="Sheet1" sheetId="1" sh:id="rId1"/>
    <sheet name="Sheet2" sheetId="2" sh:id="rId2"/>
    <sheet name="Sheet3" sheetId="3" sh:id="rId3"/>
  </sheets>
</workbook>
```

The elements and attributes used here are discussed in more detail in the following subclauses.

3.1.4 fileVersion

This contains file versioning properties, as follows.

- lastEdited – The version of the application that last saved the file.
- lowestEdited – The earliest version of the application that saved the file. This value is reset any time an application that understands all data in the file saves the file.
- rupBuild – An incremental public release of the application (e.g., RTM version or SP1 version).
- workbookPr – A group of various workbook properties.
- backupFile – A flag that indicates whether the application should create a backup of the file in question during a save operation.
- saveExternalLinkValues – A flag that indicates whether the application should cache values retrieved from other workbooks via an externally linking formula during save. If `yes`, a supporting part is written out containing a cached cell table from the external workbook.
- updateLinks – A flag that dictates how external links are handled upon opening the file. In this example, `never` means don't ask the user if they want to refresh the cached values from an external workbook, and in fact, don't ever do it until the user initiates the action.
- calcPr – Various calculation properties grouped together.
- calcId – The version of the calculation engine used to calculate values in the workbook. When a newer version of the application opens a file with an older calcId value, the application performs a full calculation of all formulas immediately after opening the workbook, to ensure proper calculation results.
- calcMode – A flag that indicates when the application should calculate formulas:
  - `Manual` means to wait for the user to initate the action.
  - `Automatic` means to perform only the needed calculations whenever a cell value changes.
- Iterate – When formula references are circular (i.e., they refer back on themselves for required input), the `iterate` flag specifies that this is an intended and valid state. Further properties not discussed here control the number of iterative calculations to perform before stopping calculation.
- bookViews – A collection of views.

3.1.5 workbookView

A single view definition represented using the following flags:

- showHorizontalScroll – Controls visibility of the horizontal scroll bar of the application. In the example above, it is set to not being visible.
- showVerticalScroll – Controls visibility of the vertical scroll bar of the application. In the example above, it is set to not being visible.
- showSheetTabs – Controls visibility of the worksheet tabs in the application. In the example above, they are set to not being visible.
- xWindow – Specifies the x coordinate (in twips) of the upper right corner of the workbook window.
- yWindow – Specifies the y coordinate (in twips) of the upper right corner of the workbook window.

- windowWidth – Specifies the width of the workbook window.
- windowHeight – Specifies the height of the workbook window.
- tabRatio – Specifies the ratio between the workbook tabs bar and the horizontal scroll bar.
- Sheets – A collection of worksheets in the workbook.
- Sheet – A single sheet definition (book-level).
- Name – The name of the worksheet. These must be unique within the workbook.
- sheetId – The internal Id of the sheet. These must be unique within the workbook.
- Id - The relationship Id that points to the sheet part definition.

3.2 Sheets

*Sheet*s are the central structures within a workbook, and are where a user does most of his spreadsheet work. The most common type of sheet is the *worksheet*, which is represented as a grid of cells. Worksheet cells can contain text, numbers, dates, and formulas. Cells can also be formatted. A workbook usually contains more than one sheet. To aid in the analysis of data and the making of informed decisions, spreadsheet applications often implement features and objects which help calculate, sort, filter, organize, and graphically display information. Since these features are often connected very tightly with the spreadsheet grid, these are also included in the sheet definition on disk.

Other types of sheets include *chart sheet*s and *dialog sheet*s.

3.2.1 Minimum Worksheet Scenario

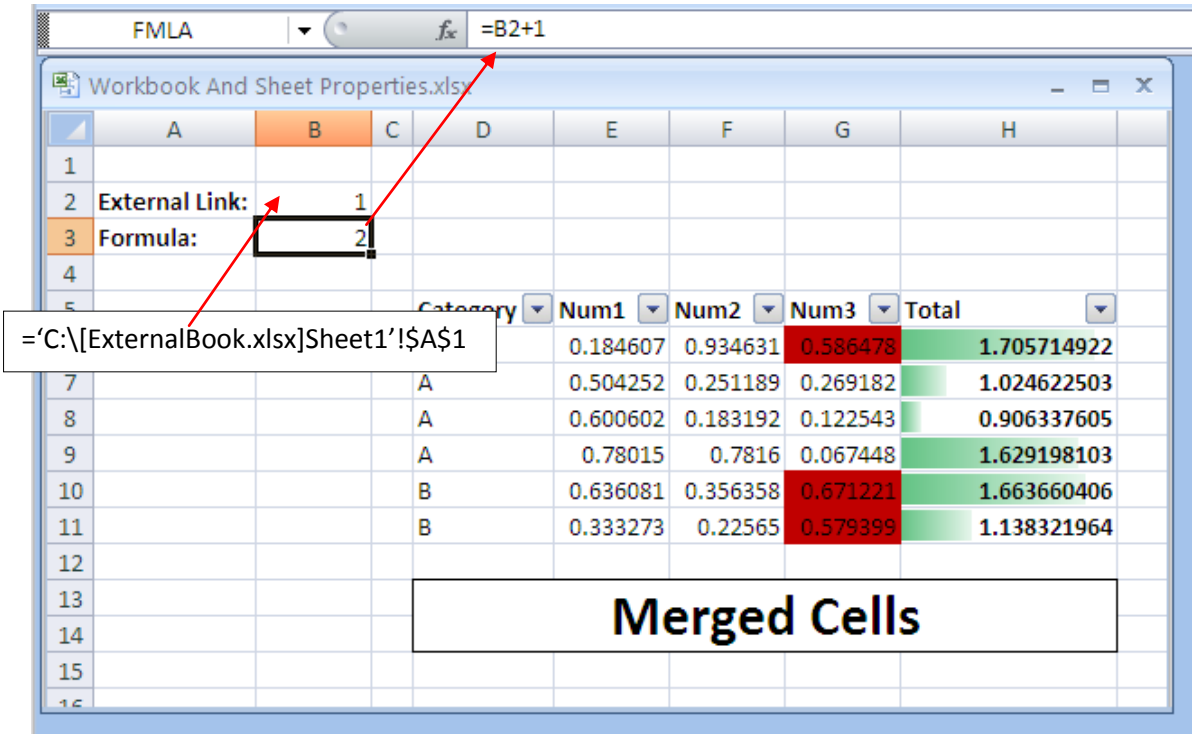
The smallest possible (blank) sheet is as follows:

```
<worksheet>
  <sheetData/>
</worksheet>
```

The empty sheetData collection represents an empty grid; this element is required. As defined in the schema, some optional sheet property collections can appear before sheetData, and some can appear after. To simplify the logic required to insert a new sheetData collection into an existing (but empty) sheet, the sheetData collection is required, even when empty.

3.2.2 Example Sheet

Consider the following graphical representation of a worksheet:



Notice that cells A2 and A3 contain text. Cell B1 contains a formula linking to another workbook, whose value is 1. Cell B2 contains a formula as well; this formula appears in the formula bar (top of picture) because it is the active cell. Cells D5:H5 contain bold-faced text that serves as headers for the table of data residing in D6:H11. The table of data has a filter feature applied to it (evidenced by drop down arrows in the header row), and columns G and H have different types of conditional formatting applied. Finally, cells D13:H14 are part of a merged cell feature, where a series of cells behave together as a single, larger cell.

When saved, the above example will have the syntax below written out in the corresponding sheet part. Sheet information is organized into three main sections:

1. Top-level sheet properties (everything before sheetData)
2. The cell table (sheetData)
3. Supporting sheet features (everything after sheetData)

Therefore, the XML for the above example would look like this, broken into three sections:

3.2.3 Sheet Properties

```
<worksheet>
  <sheetPr filterMode="1"/>
  <dimension ref="A2:H14"/>
```

```
    <sheetViews>
      <sheetView tabSelected="1" workbookViewId="0">
        <selection activeCell="B3" sqref="B3"/>
      </sheetView>
    </sheetViews>
    <sheetFormatPr defaultRowHeight="15"/>
    <cols>
      <col min="1" max="1" width="12.85546875" bestFit="1" customWidth="1"/>
      <col min="3" max="3" width="3.28515625" customWidth="1"/>
      <col min="4" max="4" width="11.140625" bestFit="1" customWidth="1"/>
      <col min="8" max="8" width="17.140625" style="1" customWidth="1"/>
    </cols>
```

3.2.4 Sheet Data

sheetData, which represents the cell table, directly after the cols collection:

```
  <sheetData>
    <row r="2" spans="1:2" customFormat="1">
      <c r="A2" s="1" t="s">
        <v>0</v>
      </c>
      <c r="B2">
        <f>[1]Sheet1!$A$1</f>
        <v>1</v>
      </c>
    </row>
    <row r="3" spans="1:8" customFormat="1">
      <c r="A3" s="1" t="s">
        <v>1</v>
      </c>
      <c r="B3">
        <f>B2+1</f>
        <v>2</v>
      </c>
      <c r="H3" s="1"/>
    </row>
    <row r="4" spans="1:8">
      <c r="H4"/>
    </row>
    <row r="5" spans="4:8">
      <c r="D5" s="1" t="s">
        <v>4</v>
      </c>
```

```
        <c r="E5" s="1" t="s">
          <v>5</v>
        </c>
        <c r="F5" s="1" t="s">
          <v>6</v>
        </c>
        <c r="G5" s="1" t="s">
          <v>7</v>
        </c>
        <c r="H5" s="1" t="s">
          <v>8</v>
        </c>
      </row>
      <row r="6" spans="4:8">
        <c r="D6" t="s">
          <v>2</v>
        </c>
        <c r="E6">
          <v>0.18460660235998017</v>
        </c>
        <c r="F6">
          <v>0.93463071023892952</v>
        </c>
        <c r="G6">
          <v>0.58647760893211043</v>
        </c>
        <c r="H6" s="1">
          <f ce="1">SUM(E6:G6)</f>
          <v>1.7057149215310201</v>
        </c>
      </row>
      <row r="7" spans="4:8">
        <c r="D7" t="s">
          <v>2</v>
        </c>
        <c r="E7">
          <v>0.50425224796279555</v>
        </c>
        <c r="F7">
          <v>0.25118866081991786</v>
        </c>
```

```
        <c r="G7">
          <v>0.26918159410869791</v>
        </c>
        <c r="H7" s="1">
          <f t="shared" ref="H7:H11" ce="1" si="0">SUM(E7:G7)</f>
          <v>1.0246225028914113</v>
        </c>
      </row>
      <row r="8" spans="4:8">
        <c r="D8" t="s">
          <v>2</v>
        </c>
        <c r="E8">
          <v>0.6006019062877066</v>
        </c>
        <c r="F8">
          <v>0.18319235857964333</v>
        </c>
        <c r="G8">
          <v>0.12254334000604317</v>
        </c>
        <c r="H8" s="1">
          <f t="shared" ce="1" si="0">SUM(E8:G8)</f>
          <v>0.9063376048733931</v>
        </c>
      </row>
      <row r="9" spans="4:8" hidden="1">
        <c r="D9" t="s">
          <v>2</v>
        </c>
        <c r="E9">
          <v>0.78015011938458589</v>
        </c>
        <c r="F9">
          <v>0.78159963723670689</v>
        </c>
        <c r="G9">
          <v>6.7448346870105036E-2</v>
        </c>
```

```
        <c r="H9" s="1">
          <f t="shared" ce="1" si="0">SUM(E9:G9)</f>
          <v>1.6291981034913978</v>
        </c>
      </row>
      <row r="10" spans="4:8" hidden="1">
        <c r="D10" t="s">
          <v>3</v>
        </c>
        <c r="E10">
          <v>0.63608141933645479</v>
        </c>
        <c r="F10">
          <v>0.35635845012920608</v>
        </c>
        <c r="G10">
          <v>0.67122053637107193</v>
        </c>
        <c r="H10" s="1">
          <f t="shared" ce="1" si="0">SUM(E10:G10)</f>
          <v>1.6636604058367328</v>
        </c>
      </row>
      <row r="11" spans="4:8" hidden="1">
        <c r="D11" t="s">
          <v>3</v>
        </c>
        <c r="E11">
          <v>0.33327331908137214</v>
        </c>
        <c r="F11">
          <v>0.2256497329592122</v>
        </c>
        <c r="G11">
          <v>0.5793989116090501</v>
        </c>
        <c r="H11" s="1">
          <f t="shared" ce="1" si="0">SUM(E11:G11)</f>
          <v>1.1383219636496344</v>
        </c>
      </row>
```

```
      <row r="13" spans="4:8">
        <c r="D13" s="2" t="s">
          <v>9</v>
        </c>
        <c r="E13" s="3"/>
        <c r="F13" s="3"/>
        <c r="G13" s="3"/>
        <c r="H13" s="4"/>
      </row>
      <row r="14" spans="4:8">
        <c r="D14" s="5"/>
        <c r="E14" s="6"/>
        <c r="F14" s="6"/>
        <c r="G14" s="6"/>
        <c r="H14" s="7"/>
      </row>
    </sheetData>
```

3.2.5 Supporting Features

The supporting feature definitions follow the cell table data:

```
    <sheetProtection objects="0" scenarios="0"/>
    <autoFilter ref="D5:H11">
      <filterColumn colId="0">
        <filters>
          <filter val="A"/>
        </filters>
      </filterColumn>
      <filterColumn colId="1">
        <customFilters and="1">
          <customFilter operator="greaterThan" val="0"/>
          <customFilter operator="lessThan" val="0.7"/>
        </customFilters>
      </filterColumn>
    </autoFilter>
    <mergeCells>
      <mergeCell ref="D13:H14"/>
    </mergeCells>
    <conditionalFormatting sqref="H6:H11">
      <cfRule type="dataBar" priority="3" stopIfTrue="0">
        <formula>MAX(IF(ISBLANK(H6:H11), "", IF(ISERROR(H6:H11), "",
          H6:H11)))</formula>
        <formula>MIN(IF(ISBLANK(H6:H11), "", IF(ISERROR(H6:H11), "",
          H6:H11)))</formula>
```

```
        <dataBar minLength="10" maxLength="90" showValue="1">
          <cfvo type="min" val="0"/>
          <cfvo type="max" val="0"/>
          <color rgb="FF63C384"/>
        </dataBar>
      </cfRule>
    </conditionalFormatting>
    <conditionalFormatting sqref="G6:G11">
      <cfRule type="cellIs" dxfId="0" priority="1" stopIfTrue="0"
        operator="greaterThan">
        <formula>0.5</formula>
      </cfRule>
    </conditionalFormatting>
    <printOptions/>
    <pageMargins left="0.7" right="0.7" top="0.75" bottom="0.75"
      header="0.3" footer="0.3"/>
    <pageSetup orientation="portrait" horizontalDpi="300" verticalDpi="300"/>
    <headerFooter/>
```

These elements are discussed in more detail in the following subclauses.

3.2.6 Sheet Properties

Referring back to §3.2.3, note that several sheet-level properties are expressed before the sheetData cell table is encountered.

sheetPr indicates that an AutoFilter has been applied on this sheet. Dimension indicates the used range on this sheet. There should be no data or formulas outside this range. The sheetViews collection indicates which cell is active on the sheet, and indicates whether this particular sheet is the active sheet in the workbook.

A collection of column-level settings appears in the cols collection.

Finally, within sheetFormatPr, a default row height is set.

3.2.7 sheetData Cell Table

The cell table is the core structure of a worksheet. It consists of all the text, numbers, and formulas in the grid.

3.2.8 Row

```
    <row r="2" spans="1:2" customFormat="1">
      <c r="A2" s="1" t="s">
        <v>0</v>
      </c>
```

```
    <c r="B2">
      <f>[1]Sheet1!$A$1</f>
      <v>1</v>
    </c>
  </row>
```

The cells in the cell table are organized by row. Each row has an index (attribute `r`) so that empty rows need not be written out. Each row indicates the number of cells defined for it, as well as their relative position in the sheet. In this example, the first row of data is row 2.

3.2.9 Cell

```
  <c r="B3">
    <f>B2+1</f>
    <v>2</v>
  </c>
```

The cell itself is expressed by the `c` collection. Each cell indicates it's location in the grid using A1-style reference notation. A cell can also indicate a style identifier (attribute `s`) and a data type (attribute `t`). The cell types include string, number, and Boolean. In order to optimize load/save operations, default data values are not written out.

3.2.9.1 Cell Values

Cells contain values, whether the values were directly typed in (e.g., cell A2 in our example has the value `External Link:`) or are the result of a calculation (e.g., cell B3 in our example has the formula `B2+1`).

String values in a cell are not stored in the cell table unless they are the result of a calculation. Therefore, instead of seeing `External Link:` as the content of the cell's `v` node, instead you see a zero-based index into the shared string table where that string is stored uniquely. This is done to optimize load/save performance and to reduce duplication of information. To determine whether the `0` in `v` is a number or an index to a string, the cell's data type must be examined. When the data type indicates string, then it is an index and not a numeric value.

3.2.9.2 Formulas

Cells can contain formulas, which calculate results. Formulas are expressed in the file the same way the user sees them at runtime of the application. This is specifically a design choice meant to aid in creation and processing of workbook contents.

A formula can have attributes on it indicating how to handle calculation of the cell.

3.2.9.2.1 Shared Formulas

```
<row r="7" spans="4:8">
  <c r="H7" s="1">
    <f t="shared" ref="H7:H11" ce="1" si="0">SUM(E7:G7)</f>
    <v>1.0246225028914113</v>
  </c>
</row>
<row r="8" spans="4:8">
  <c r="H8" s="1">
    <f t="shared" ce="1" si="0">SUM(E8:G8)</f>
    <v>0.9063376048733931</v>
  </c>
</row>
```

Just as strings in cells can be extremely pervasive and redundant in a sheet (and therefore must be optimized), formulas are also extremely pervasive in a sheet, and often can be optimized. Consider the table in the above example, where column H contains a formula that sums the numbers in columns E through G, for each row. The only difference between the formulas in H6:H12 is that the reference increases by 1 row from one row to the next. Therefore, an optimization is created where only the formula in H6 needs to be written out, with some additional information indicating how far to propagate the formula once loaded. This enables the loading application to load and parse only the first of the shared formulas, and then more quickly apply the necessary transforms to produce the additional related formulas in subsequent cells.

Note that while formulas can be shared, it is desirable to enable easy access to the contents of a cell. Therefore, it is allowed that all formulas may be written out, but only the primary formula in a shared formula need be loaded and parsed.

3.2.9.2.2 External Referencing Formulas

```
<c r="B2">
  <f>[1]Sheet1!$A$1</f>
  <v>1</v>
</c>
```

In the above example, cell B2 contains a formula that references a cell in another workbook, namely `'C:\[ExternalBook.xlsx]Sheet1'!$A$1`. This formula is referencing `ExternalBook.xlsx` located at `c:\`. Furthermore, the formula is requesting the value of cell A1 on `Sheet1` of that particular workbook.

Instead of writing `'C:\[ExternalBook.xlsx]Sheet1'!$A$1` directly in the formula, it is desirable to make all external references much more accessible, especially given the potentially enormous size of a cell table. Therefore, the URL and file location is persisted using the relationships semantic, in a relationship file, and then referenced inline with the formula: `[1]Sheet1!$A$1`. In this way, external resource files can more easily be determined and updated if needed.

Note that whenever a workbook contains a formula referencing another workbook, some values from that external workbook are also cached with the referencing workbook. This is done so that if a recalculation of the workbook is needed and the workbook isn't accessible, a cached value may be used to complete the calculation.

3.2.10 Supporting Sheet Features

3.2.11 Defined Names

```
<definedNames>
  <definedName name="FMLA">Sheet1!$B$3</definedName>
  <definedName name="SheetLevelName" comment="This name is scoped to Sheet1"
    localSheetId="0">Sheet1!$B$3</definedName>
</definedNames>
```

Defined names can be used in place of cell references in formulas. For example, instead of using B3+1 to add 1 to the value that's in B3, one could define a name, as in FMLA, and assign it to B3. Then FMLA+1 can be used to perform the calculation.

Names can be defined and assigned to a cell location or range or to a formula or constant value. Names can be referenced in formulas. Names can be scoped to either the entire workbook (default) or just the local sheet. Names scoped to the local sheet cannot be referenced from other sheets. Names scoped to the workbook can be referenced from any sheet.

Defined names are actually stored in the workbook part, but are discussed here in the context of the sheet because they are so closely related to cells and formulas.

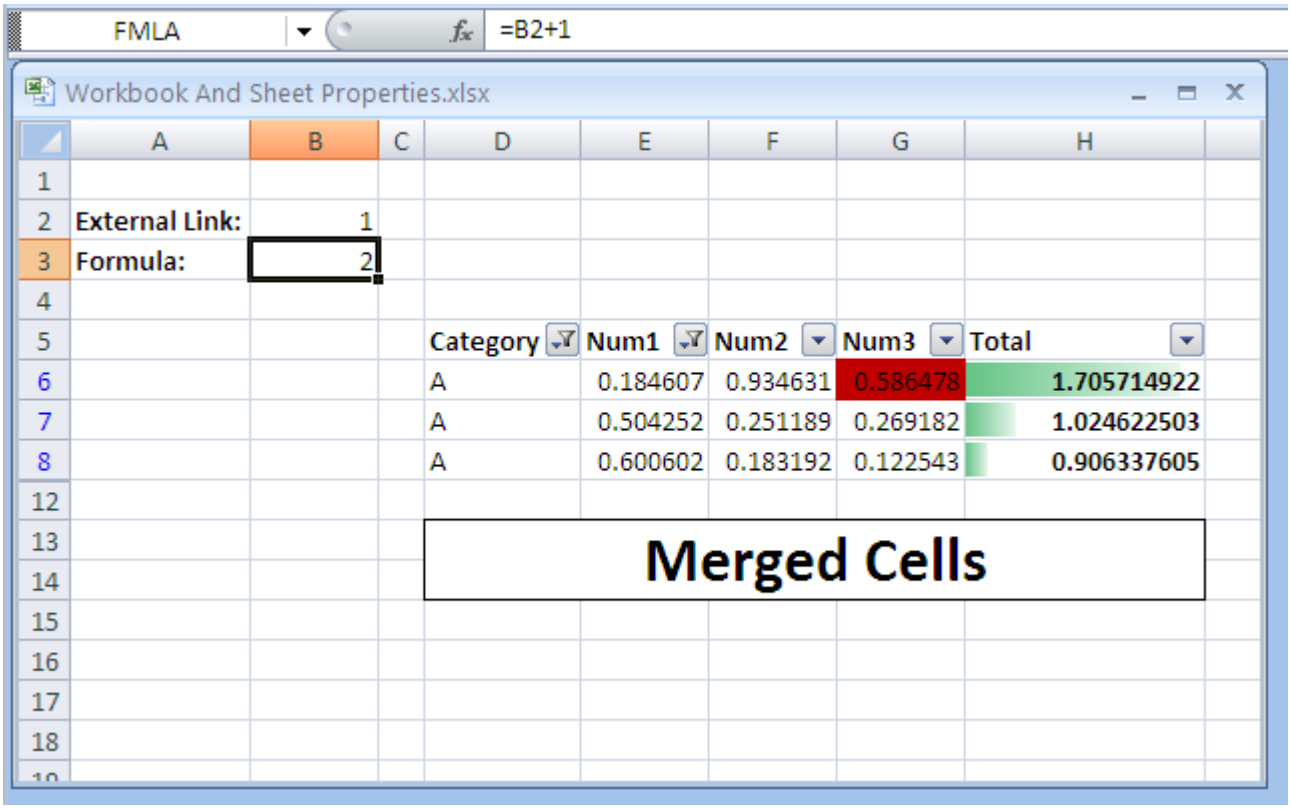
3.2.12 AutoFilter

```
<autoFilter ref="D5:H11">
  <filterColumn colId="0">
    <filters>
      <filter val="A"/>
    </filters>
  </filterColumn>
  <filterColumn colId="1">
    <customFilters and="1">
      <customFilter operator="greaterThan" val="0"/>
      <customFilter operator="lessThan" val="0.7"/>
    </customFilters>
  </filterColumn>
</autoFilter>
```

AutoFilters specify criteria for which cells in a table should be displayed. In this example, the first column (zero-based index colId) in the table (cells D5:D11), has a criteria specifying that only rows in the table whose value in column D are equal to A will be shown. The rest of the rows are *hidden*.

A second criterion is specified as well, on the 2nd column, E: only rows whose values in column E are greater than 0 and less than 0.7.

The resulting grid could be rendered like this:



3.2.13 Merged Cells

```
<mergeCells>
  <mergeCell ref="D13:H14"/>
</mergeCells>
```

In the example, cells D13:H14 have been merged into a single, larger cell. Note that the cell table itself doesn't reflect this merge, but it does reflect the data content and formatting. Specifically, the top-left cell in a merged collection of cells contains the value, and all the cells reflect the various border formatting.

3.2.14 Conditional Formatting

```
<conditionalFormatting sqref="H6:H11">
  <cfRule type="dataBar" priority="3" stopIfTrue="0">
    <formula>MAX(IF(ISBLANK(H6:H11), "", IF(ISERROR(H6:H11), "",
      H6:H11)))</formula>
    <formula>MIN(IF(ISBLANK(H6:H11), "", IF(ISERROR(H6:H11), "",
      H6:H11)))</formula>
```

```
      <dataBar minLength="10" maxLength="90" showValue="1">
        <cfvo type="min" val="0"/>
        <cfvo type="max" val="0"/>
        <color rgb="FF63C384"/>
      </dataBar>
    </cfRule>
  </conditionalFormatting>
  <conditionalFormatting sqref="G6:G11">
    <cfRule type="cellIs" dxfId="0" priority="1" stopIfTrue="0"
  operator="greaterThan">
      <formula>0.5</formula>
    </cfRule>
  </conditionalFormatting>
```

There are two conditional formats applied: one to the table of data in column H and the other to the table of data in column G.

In column G, a red fill is applied to any cell whose value is greater than 0.5. Notice that sqref specifies the range to which the rule applies. The formatting is specified by dxfId, which is a reference to a formatting expression in the central styles part.

In column H there is a dataBar formatting rule, which applies a variable length bar to the cell background, where the length of the bar depends on the relative value of the cell.

3.3 Shared String Table

3.3.1 Overview

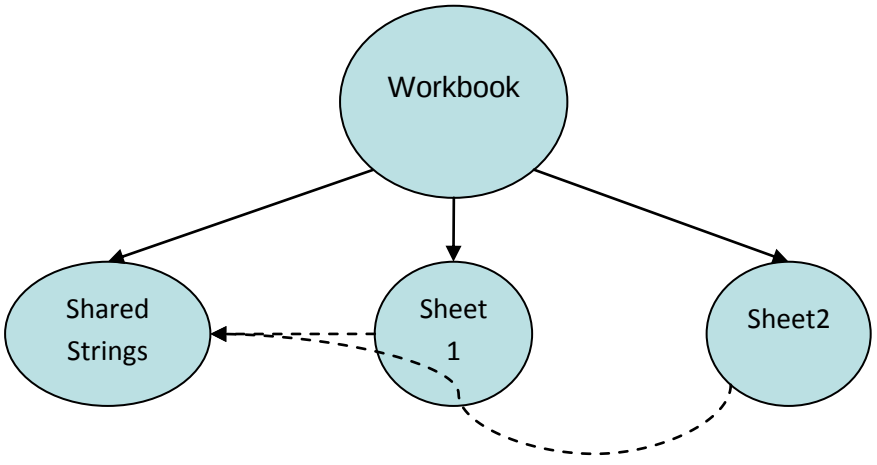
A workbook may contain thousands of cells containing string (non-numeric) data. Furthermore, this data is very likely to be repeated across many rows or columns. The goal of implementing a single string table that is shared across the workbook is to improve performance in opening and saving the file by only reading and writing the repetitive information once.

For example, consider a workbook summarizing information for cities within various countries. There may be a column for the name of the country, a column for the name of each city in that country, and a column containing the data for each city:

	A	B	C	D
1				
2		**Country**	**City**	**Data**
3		United States	Seattle	469.83
4		United States	Denver	36.16
5		United States	New York	114.38
6		United States	Philadelphia	540.95
7		United States	Houton	649.07
8		United States	San Diego	258.4
9		United States	San Francisco	686.05
10		Argentina	Buenos Aires	14.08
11		Argentina	San Juan	28.17
12		Argentina	Salta	757.31
13		Argentina	Rosario	246.23
14		Argentina	La Plata	947.63
15		Japan	Tokyo	597.55
16		Japan	Nagoya	619.01
17		Japan	Yokohama	525.14
18		Japan	Sendai	611.74
19				

In this case, the country name is repetitive, being duplicated in many cells. In many cases, the repetition is extensive, and a tremendous savings is realized by making use of a shared string table when saving the workbook.

3.3.2 File Architecture



There is a single shared strings part for all the strings in a workbook. This part is related to the workbook. Each cell (in sheet1.xml, for example) containing a string value refers by index to a string expressed in the shared

strings part. The solid arrows represent relationships among the parts and the dotted arrows represent references by index to a string in the shared strings part.

3.3.3 Example: Plain Text

This first example demonstrates plain text in cells. Note that in this example, some of the cells are formatted (e.g., the column headers "Country", "City", and "Data" are bold faced). Since the formatting is applied at the cell level, cell styles and formatting is used to describe the formatting, rather than using text formatting on the text itself.

A later example demonstrates how to handle a variety of text formatting (rich text) within a single cell.

3.3.4 Illustration

Consider the example in the introduction:

	A	B	C	D
1				
2		**Country**	**City**	**Data**
3		United States	Seattle	469.83
4		United States	Denver	36.16
5		United States	New York	114.38
6		United States	Philadelphia	540.95
7		United States	Houton	649.07
8		United States	San Diego	258.4
9		United States	San Francisco	686.05
10		Argentina	Buenos Aires	14.08
11		Argentina	San Juan	28.17
12		Argentina	Salta	757.31
13		Argentina	Rosario	246.23
14		Argentina	La Plata	947.63
15		Japan	Tokyo	597.55
16		Japan	Nagoya	619.01
17		Japan	Yokohama	525.14
18		Japan	Sendai	611.74
19				

In this example, the country names in the column titled 'Country'—"United States", "Argentina", and "Japan"—would appear a single time in the shared strings part. Additionally, all the city names (in the column titled 'City') would appear a single time in the shared strings part, as would the column titles themselves in B2:D2. The numeric values in the 'Data' column would be expressed inline with the cell table definition (e.g., in Sheet1.xml).

3.3.5 The XML

The shared string table XML for this example looks like this:

```
<sst … count="35" uniqueCount="22">
  <si>
    <t>United States</t>
  </si>
  <si>
    <t>Seattle</t>
  </si>
  <si>
    <t>Denver</t>
  </si>
  <si>
    <t>New York</t>
  </si>
  <si>
    <t>Philadelphia</t>
  </si>
  <si>
   <t>Houton</t>
  </si>
  <si>
    <t>San Diego</t>
  </si>
  <si>
    <t>San Francisco</t>
  </si>
  <si>
    <t>Argentina</t>
  </si>
  <si>
    <t>Buenos Aires</t>
  </si>
  <si>
   <t>San Juan</t>
  </si>
  <si>
    <t>Salta</t>
  </si>
  <si>
    <t>Rosario</t>
  </si>
  <si>
    <t>La Plata</t>
  </si>
```

```
    <si>
      <t>Japan</t>
    </si>
    <si>
      <t>Tokyo</t>
    </si>
    <si>
      <t>Nagoya</t>
    </si>
    <si>
      <t>Yokohama</t>
    </si>
    <si>
      <t>Sendai</t>
    </si>
    <si>
      <t>Country</t>
    </si>
    <si>
      <t>City</t>
    </si>
    <si>
      <t>Data</t>
    </si>
  </sst>
```

The cell table for this example looks like this:

```
  <sheetData>
    <row r="2" spans="2:4" customFormat="1">
      <c r="B2" s="1" t="s">
        <v>19</v>
      </c>
      <c r="C2" s="1" t="s">
        <v>20</v>
      </c>
      <c r="D2" s="1" t="s">
        <v>21</v>
      </c>
    </row>
    <row r="3" spans="2:4" customFormat="1">
      <c r="B3" t="s">
        <v>0</v>
      </c>
```

```
        <c r="C3" t="s">
          <v>1</v>
        </c>
        <c r="D3">
          <f t="shared" ref="D3:D18" ca="1" si="0">ROUND(RAND()*1000,2)</f>
          <v>374.9</v>
        </c>
      </row>
      <row r="4" spans="2:4" customFormat="1">
        <c r="B4" t="s">
          <v>0</v>
        </c>
        <c r="C4" t="s">
          <v>2</v>
        </c>
        <c r="D4">
          <f t="shared" ca="1" si="0"/>
          <v>452.82</v>
        </c>
      </row>
      <row r="5" spans="2:4" customFormat="1">
        <c r="B5" t="s">
          <v>0</v>
        </c>
        <c r="C5" t="s">
          <v>3</v>
        </c>
        <c r="D5">
          <f t="shared" ca="1" si="0"/>
          <v>632.1</v>
        </c>
      </row>
      <row r="6" spans="2:4" customFormat="1">
        <c r="B6" t="s">
          <v>0</v>
        </c>
        <c r="C6" t="s">
          <v>4</v>
        </c>
        <c r="D6">
          <f t="shared" ca="1" si="0"/>
          <v>374.9</v>
        </c>
      </row>
      <row r="7" spans="2:4" customFormat="1">
        <c r="B7" t="s">
          <v>0</v>
        </c>
        <c r="C7" t="s">
          <v>5</v>
        </c>
        <c r="D7">
          <f t="shared" ca="1" si="0"/>
          <v>452.82</v>
        </c>
      </row>
      <row r="8" spans="2:4" customFormat="1">
        <c r="B8" t="s">
          <v>0</v>
        </c>
        <c r="C8" t="s">
          <v>6</v>
        </c>
        <c r="D8">
          <f t="shared" ca="1" si="0"/>
          <v>632.1</v>
        </c>
      </row>
      <row r="9" spans="2:4" customFormat="1">
        <c r="B9" t="s">
          <v>7</v>
        </c>
        <c r="C9" t="s">
          <v>1</v>
        </c>
        <c r="D9">
          <f t="shared" ca="1" si="0"/>
          <v>374.9</v>
        </c>
      </row>
      <row r="10" spans="2:4" customFormat="1">
        <c r="B10" t="s">
          <v>7</v>
        </c>
        <c r="C10" t="s">
          <v>2</v>
        </c>
        <c r="D10">
          <f t="shared" ca="1" si="0"/>
          <v>452.82</v>
        </c>
      </row>
      <row r="11" spans="2:4" customFormat="1">
        <c r="B11" t="s">
          <v>7</v>
        </c>
        <c r="C11" t="s">
          <v>3</v>
        </c>
        <c r="D11">
          <f t="shared" ca="1" si="0"/>
          <v>632.1</v>
        </c>
      </row>
      <row r="12" spans="2:4" customFormat="1">
        <c r="B12" t="s">
          <v>7</v>
        </c>
        <c r="C12" t="s">
          <v>4</v>
        </c>
        <c r="D12">
          <f t="shared" ca="1" si="0"/>
          <v>374.9</v>
        </c>
      </row>
      <row r="13" spans="2:4" customFormat="1">
        <c r="B13" t="s">
          <v>7</v>
        </c>
        <c r="C13" t="s">
          <v>5</v>
        </c>
        <c r="D13">
          <f t="shared" ca="1" si="0"/>
          <v>452.82</v>
        </c>
      </row>
      <row r="14" spans="2:4" customFormat="1">
        <c r="B14" t="s">
          <v>7</v>
        </c>
        <c r="C14" t="s">
          <v>6</v>
        </c>
        <c r="D14">
          <f t="shared" ca="1" si="0"/>
          <v>632.1</v>
        </c>
      </row>
      <row r="15" spans="2:4" customFormat="1">
        <c r="B15" t="s">
          <v>8</v>
        </c>
        <c r="C15" t="s">
          <v>1</v>
        </c>
        <c r="D15">
          <f t="shared" ca="1" si="0"/>
          <v>374.9</v>
        </c>
      </row>
      <row r="16" spans="2:4" customFormat="1">
        <c r="B16" t="s">
          <v>8</v>
        </c>
        <c r="C16" t="s">
          <v>2</v>
        </c>
        <c r="D16">
          <f t="shared" ca="1" si="0"/>
          <v>452.82</v>
        </c>
      </row>
      <row r="17" spans="2:4" customFormat="1">
        <c r="B17" t="s">
          <v>8</v>
        </c>
        <c r="C17" t="s">
          <v>3</v>
        </c>
        <c r="D17">
          <f t="shared" ca="1" si="0"/>
          <v>632.1</v>
        </c>
      </row>
      <row r="18" spans="2:4" customFormat="1">
        <c r="B18" t="s">
          <v>8</v>
        </c>
        <c r="C18" t="s">
          <v>4</v>
        </c>
        <c r="D18">
          <f t="shared" ca="1" si="0"/>
          <v>374.9</v>
        </c>
      </row>
      <row r="19" spans="2:4" customFormat="1">
        <c r="B19" t="s">
          <v>8</v>
        </c>
        <c r="C19" t="s">
          <v>5</v>
        </c>
        <c r="D19">
          <f t="shared" ca="1" si="0"/>
          <v>452.82</v>
        </c>
      </row>
      <row r="20" spans="2:4" customFormat="1">
        <c r="B20" t="s">
          <v>8</v>
        </c>
        <c r="C20" t="s">
          <v>6</v>
        </c>
        <c r="D20">
          <f t="shared" ca="1" si="0"/>
          <v>632.1</v>
        </c>
      </row>
    </sheetData>
    <mergeCells count="1">
      <mergeCell ref="B2:D2"/>
    </mergeCells>
    <phoneticPr fontId="1" type="noConversion"/>
    <pageMargins left="0.75" right="0.75" top="1" bottom="1" header="0.5" footer="0.5"/>
    <pageSetup paperSize="9" orientation="portrait" horizontalDpi="4294967292" verticalDpi="4294967292"/>
    <drawing r:id="rId1"/>
  </worksheet>
</file>
```

```
        <c r="D6">
          <f t="shared" ca="1" si="0"/>
          <v>886.37</v>
        </c>
      </row>
      <row r="7" spans="2:4" customFormat="1">
        <c r="B7" t="s">
          <v>0</v>
        </c>
        <c r="C7" t="s">
          <v>5</v>
        </c>
        <c r="D7">
          <f t="shared" ca="1" si="0"/>
          <v>291.14</v>
        </c>
      </row>
      <row r="8" spans="2:4" customFormat="1">
        <c r="B8" t="s">
          <v>0</v>
        </c>
        <c r="C8" t="s">
          <v>6</v>
        </c>
        <c r="D8">
          <f t="shared" ca="1" si="0"/>
          <v>114.97</v>
        </c>
      </row>
      <row r="9" spans="2:4" customFormat="1">
        <c r="B9" t="s">
          <v>0</v>
        </c>
        <c r="C9" t="s">
          <v>7</v>
        </c>
        <c r="D9">
          <f t="shared" ca="1" si="0"/>
          <v>291.99</v>
        </c>
      </row>
```

```
      <row r="10" spans="2:4" customFormat="1">
        <c r="B10" t="s">
          <v>8</v>
        </c>
        <c r="C10" t="s">
          <v>9</v>
        </c>
        <c r="D10">
          <f t="shared" ca="1" si="0"/>
          <v>335.42</v>
        </c>
      </row>
      <row r="11" spans="2:4" customFormat="1">
        <c r="B11" t="s">
          <v>8</v>
        </c>
        <c r="C11" t="s">
          <v>10</v>
        </c>
        <c r="D11">
          <f t="shared" ca="1" si="0"/>
          <v>664.72</v>
        </c>
      </row>
      <row r="12" spans="2:4" customFormat="1">
        <c r="B12" t="s">
          <v>8</v>
        </c>
        <c r="C12" t="s">
          <v>11</v>
        </c>
        <c r="D12">
          <f t="shared" ca="1" si="0"/>
          <v>992.62</v>
        </c>
      </row>
      <row r="13" spans="2:4" customFormat="1">
        <c r="B13" t="s">
          <v>8</v>
        </c>
        <c r="C13" t="s">
          <v>12</v>
        </c>
```

```
      <c r="D13">
        <f t="shared" ca="1" si="0"/>
        <v>148.5</v>
      </c>
    </row>
    <row r="14" spans="2:4" customFormat="1">
      <c r="B14" t="s">
        <v>8</v>
      </c>
      <c r="C14" t="s">
        <v>13</v>
      </c>
      <c r="D14">
        <f t="shared" ca="1" si="0"/>
        <v>193.53</v>
      </c>
    </row>
    <row r="15" spans="2:4" customFormat="1">
      <c r="B15" t="s">
        <v>14</v>
      </c>
      <c r="C15" t="s">
        <v>15</v>
      </c>
      <c r="D15">
        <f t="shared" ca="1" si="0"/>
        <v>849.36</v>
      </c>
    </row>
    <row r="16" spans="2:4" customFormat="1">
      <c r="B16" t="s">
      <v>14</v>
      <c r="C16" t="s">
        <v>16</v>
      </c>
      <c r="D16">
        <f t="shared" ca="1" si="0"/>
        <v>765.46</v>
      </c>
    </row>
```

```
    <row r="17" spans="2:4" customFormat="1">
      <c r="B17" t="s">
        <v>14</v>
      </c>
      <c r="C17" t="s">
        <v>17</v>
      </c>
      <c r="D17">
        <f t="shared" ca="1" si="0"/>
        <v>350.26</v>
      </c>
    </row>
    <row r="18" spans="2:4" customFormat="1">
      <c r="B18" t="s">
        <v>14</v>
      </c>
      <c r="C18" t="s">
        <v>18</v>
      </c>
      <c r="D18">
        <f t="shared" ca="1" si="0"/>
        <v>979.22</v>
      </c>
    </row>
  </sheetData>
```

3.3.6 Shared String Table

```
  <sst … count="35" uniqueCount="22">
    <si>
      <t>United States</t>
    </si>
    <si>
      <t>Seattle</t>
    </si>
    <si>
      <t>Denver</t>
    </si>
```

Examining the XML for the shared string part, it can be found that the first entry in the string table is "United States", residing in position 0. The value "Seattle" can be found in position 1 and "Denver" can be found in position 2.

3.3.7 Cell Table

```
<row r="2" spans="2:4" customFormat="1">
  <c r="B2" s="1" t="s">
    <v>19</v>
  </c>
  <c r="C2" s="1" t="s">
    <v>20</v>
  </c>
  <c r="D2" s="1" t="s">
    <v>21</v>
  </c>
</row>
```

The first cell in our spreadsheet that contains data is B2. The XML indicates that it is of type 'string' (`t="s"`). This indicates that the numeric value found inside the `<v>` element is an index to a string in the string table rather than an actual number in the spreadsheet. The value for cell B2 is '19'. The 19th entry in the shared string table (counting the first entry as 0) has a value of "Country". Therefore, cell B2 contains the word "Country".

```
<row r="3" spans="2:4" customFormat="1">
  <c r="B3" t="s">
    <v>0</v>
  </c>
  <c r="C3" t="s">
    <v>1</v>
  </c>
  <c r="D3">
    <f t="shared" ref="D3:D18" ca="1" si="0">ROUND(RAND()*1000,2)</f>
    <v>374.9</v>
  </c>
</row>
```

Cell B3 (`<c @r="B3"…>`) is also of type string, and the '0' inside the `v` element refers to the 0th item in the string table, which corresponds to the string value "United States". Cell C3 is a string type of cell and references the shared string found in position 1 in the string table, corresponding to the value "Seattle". Cell D3 contains an `f` element, indicating a formula.

```
<row r="4" spans="2:4" customFormat="1">
  <c r="B4" t="s">
    <v>0</v>
  </c>
  <c r="C4" t="s">
    <v>2</v>
  </c>
```

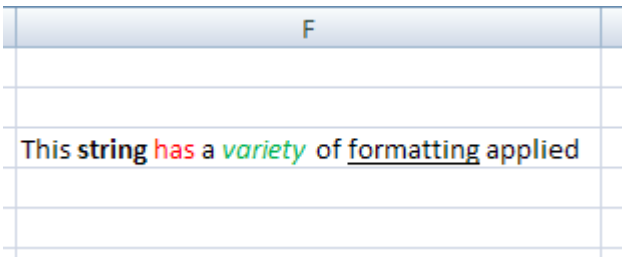
```
    <c r="D4">
      <f t="shared" ca="1" si="0"/>
      <v>452.82</v>
    </c>
  </row>
```

Examining the cell table entries for the data in row 4 of the spreadsheet, we see that cell B4 also contains the string value "United States". This is the 2nd occurrence of the value "United States" in this example. Since this value only occurs once in the string table, again the cell is using an index of 0 to reference the string item in the string table. Cell C4 is of type string and an index value of '2' indicates that "Denver" is the value of this cell.

3.3.8 Example: Rich Text

In this example, a single string cell value has multiple types of text formatting applied to various parts of the text.

3.3.9 Illustration



3.3.10 Shared String Table

The main difference between plain text and rich text is seen in the string table itself. The si element is capable of containing rich text expressions:

```
  <si>
    <r>
      <t xml:space="preserve">This </t>
    </r>
    <r>
      <rPr>
        <b/>
        <sz val="11"/>
        <color theme="1"/>
        <rFont val="Calibri"/>
        <family val="2"/>
        <scheme val="minor"/>
      </rPr>
      <t xml:space="preserve">string </t>
    </r>
```

```
      <r>
        <rPr>
          <sz val="11"/>
          <color rgb="FFFF0000"/>
          <rFont val="Calibri"/>
          <family val="2"/>
          <scheme val="minor"/>
        </rPr>
        <t>has</t>
      </r>
      <r>
        <rPr>
          <sz val="11"/>
          <color theme="1"/>
          <rFont val="Calibri"/>
          <family val="2"/>
          <scheme val="minor"/>
        </rPr>
        <t xml:space="preserve"> a </t>
      </r>
      <r>
        <rPr>
          <i/>
          <sz val="11"/>
          <color rgb="FF00B050"/>
          <rFont val="Calibri"/>
          <family val="2"/>
          <scheme val="minor"/>
        </rPr>
        <t>variety</t>
      </r>
      <r>
        <rPr>
          <sz val="11"/>
          <color theme="1"/>
          <rFont val="Calibri"/>
          <family val="2"/>
          <scheme val="minor"/>
        </rPr>
        <t xml:space="preserve"> of </t>
      </r>
      <r>
```

```
        <rPr>
          <u/>
          <sz val="11"/>
          <color theme="1"/>
          <rFont val="Calibri"/>
          <family val="2"/>
          <scheme val="minor"/>
        </rPr>
        <t>formatting</t>
      </r>
      <r>
        <rPr>
          <sz val="11"/>
          <color theme="1"/>
          <rFont val="Calibri"/>
          <family val="2"/>
          <scheme val="minor"/>
        </rPr>
        <t xml:space="preserve"> applied</t>
      </r>
    </si>
```

Reading the string from left to right as it appears in the cell, each word represents a change in formatting. This change in formatting corresponds to separate run elements r to separate the text with different formatting. Every word is expressed using a run element r, which expresses the properties of the text rPr and the text itself t.

Since there are no properties associated with the first word "This", the text inherits the default formatting for the cell.

The rich text expression for the second string "string" contains a bold faced font element indicator b in the run properties rPr, therefore this text will have bold face applied. While other text formatting properties are expressed, they are the same as the cell formatting. This additional information is expressed for the sake of clarity and completeness of expression.

The rich text expression for the third string "has" contains a color element indicator color in the run properties rPr. Therefore, the color of the text associated with this set of run properties will be red, according to the color value expressed.

The formatting for the remaining words in this rich text string can be deduced in a similar manner, such that "a" has default formatting applied, "variety" is both italicized and green, "of" has default formatting applied, "formatting" is underlined, and "applied" has default formatting applied.

3.4 Tables

3.4.1 Overview

A table helps organize and provide structure to lists of information in a worksheet. Tables have clearly labeled columns, rows, and data regions. Tables enable users to sort, analyze, format, manage, add, and delete information. Here's an example of what a table can look like:

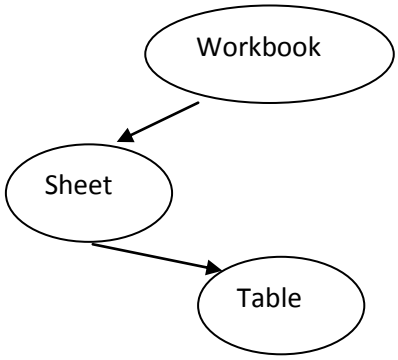
	A	B	C	D
1				
2		State	City	Zipcode
3		WA	Seattle	98101
4		WA	Seattle	98102
5		WA	Tacoma	98467
6		OR	Portland	97204
7		OR	Portland	97205
8		ID	Post Falls	83854
9		Count		6

Notice that this table has column headings "State", "City", and "Zipcode". There is a row summarizing the data, in this case a count of zip codes. The formatting helps make clear where the column headings are (Bold faced, bordered on top and bottom all the way across), where the data region is (banded row stripes), and where the totals row is (double border separating data from totals, bold face totals label).

Because the table feature has been applied to this data, special behaviors can be applied which help the user perform useful actions. For example, if the user types additional data in row 10, the table can expand and automatically add that data to the data region of the table. Similarly, adding a column is as easy as typing a new column heading to the right or left of the current column headings. Filter and sort abilities are automatically surfaced to the user via the drop down arrows. Special calculated columns can be created which summarize or calculate data in the table. These columns have the ability to expand and shrink according to size of the table, and maintain proper formula referencing.

Tables can be made from data already present in the worksheet. Tables can also be the result of an external data query. Finally, tables can be the result of mapping a collection of repeating XML elements to a worksheet range.

3.4.2 File Architecture



Each table is referred to by a relationship from a sheet to the table. The relationship is found in the sheet's _rels directory. The sheet XML also references the ID of this relationship, because there can be more than one table in a sheet.

The sheet XML stores the numeric and textual data. The table XML records the various attributes for the particular table object.

3.4.3 Example: Table

This example demonstrates a table created from data that was previously entered in the sheet. (See §3.15 for a discussion of Tables with XML data bindings.)

3.4.4 Illustration

Consider the example provided in §3.4.1 above:

	A	B	C	D
1				
2				
3		State	City	Zipcode
4		WA	Seattle	98101
5		WA	Seattle	98102
6		WA	Tacoma	98467
7		OR	Portland	97204
8		OR	Portland	97205
9		ID	Post Falls	83854
10		**Count**		6
11				

Notice that this table has column headings "State", "City", and "Zipcode". There is a row summarizing the data, in this case a count of Zip codes. In the "State" column abbreviations of United States State names are listed. In the City column are listed names of Cities within those states. Finally, within the Zipcode column are postal codes residing within those cities.

The table has a style applied, which provides unique formatting for:

- The column heading area, with bold facing and top and bottom borders
- The data area, with banded striping
- The total row area, with a top double border and bold facing
- The last column area, with a solid background fill of blue.

3.4.5 The Sheet XML

The sheet XML for this example references the table definition part:

```
      …
      <tableParts count="1">
        <tablePart r:id="rId1"/>
      </tableParts>
  </worksheet>
```

3.4.6 The Table XML

The tableParts collection appears after the sheetData section of the sheet. This sheet references a table whose relationship Id r:id value is `rId1`.

The Table definition XML for this example:

```
  <table xmlns= … id="8" name="Table19" displayName="Table19" ref="B3:D10"
    totalsRowCount="1">
    <autoFilter ref="B3:D10"/>
    <tableColumns count="3">
      <tableColumn id="1" name="State" totalsRowLabel="Count"
        totalsRowDxfId="0"/>
      <tableColumn id="2" name="City"/>
      <tableColumn id="3" name="Zipcode" totalsRowFunction="count"/>
    </tableColumns>
    <tableStyleInfo name="TableStyleMedium16" showFirstColumn="0"
      showLastColumn="1"
      showRowStripes="1" showColumnStripes="0"/>
  </table>
```

name indicates that the Table's name is `Table19`, and the ref value indicates that it occupies the range `B3:D10` on the relevant sheet. totalsRowCount value of `1` indicates that this Table's total row is visible.

The autoFilter element indicates that the autoFilter feature is applied to the range `B3:D10`. The tableColumns collection indicates there are 3 columns in the table, whose names are "State", "City", and "Zipcode". Furthermore, the column titled "State" has a label in the total row, whose caption is "Count", and the column titled "Zipcode" has a total row function applied, whose function is "count".

The tableStyleInfo element indicates various attributes of this Table's style and formatting. In this example, name indicates that the Table style named "TableStyleMedium16" has been applied. Additionally, even though formatting has been defined by this table style to indicate uniquely the first column of the table, since showFirstColumn is set to `0` (false), this first column formatting will not be applied to the table. The same is true for column stripes. Since showColumnStripes is set to `0` (false), even though formatting for column stripes is defined by the table style, it is not applied to this table. However, both row striping and last column formatting is set to be applied to this table, as indicated by showRowStripes and showLastColumn.

3.5 Calculation Chain

3.5.1 Overview

The Calculation Chain part specifies the order in which cells in the workbook were last calculated. It only records information about cells containing formulas. It does *not* include any information about the formula-dependency calculation tree. In other words, the Calculation Chain part does not indicate the dependencies that formulas have on other cell values; it only indicates the order in which the cells were last calculated.

Any particular calculation event can cause the calculation chain order to be rearranged or altered. For example, adding more formulas to the workbook will add references in the Calculation Chain part.

Another example of how the calculation order can be updated involves the idea of partial calculation. *Partial calculation* is an optimization a spreadsheet application can implement to calculate only those cells that are dependent on other cells whose values have changed, and to ignore other formulas in the workbook. This helps to avoid redundantly recalculating results that are already known. Therefore, if a set of formulas that were previously ignored during a calculation become required for calculation (due to a cell's value changing), then these formulas will move to "first" on the calculation chain so they can be evaluated.

While calculation chain information can be loaded by a spreadsheet application, it is not required. A calculation chain can be constructed in memory at load-time based on the formulas and their interdependence, if the spreadsheet application finds this information useful. The order expressed in the Calculation Chain part does not force or dictate to the implementing application the order in which calculations must be performed at runtime.

3.5.2 Example

Consider the following set of formulas in a workbook:

	A		B		C		D	
1	1	1	11	11				
2	=A1+1	2	=B1+1	12				
3	=A2+1	3	=B2+1	13				
4	=A3+1	4	=B3+1	14				
5	=A4+1	5	=B4+1	15				
6	=A5+1	6	=B5+1	16				
7	=A6+1	7	=B6+1	17				
8	=A7+1	8	=B7+1	18				
9	=A8+1	9	=B8+1	19				
10	=A9+1	10	=B9+1	20	=B10+A10	30	=C10+10	40
11								

Note that the content of each cell is displayed on the left side of the cell, and the evaluated value is superimposed on the right side of each cell.

Cell A1 contains the numeric constant 1. Cell A2 contains the formula =A1+1, and this formula is filled down to A10. Cell B1 contains the numeric constant 11. Cell B2 contains the formula =B1+1, and this formula is filled down to B10. C10 contains the formula =B10+A10, whose current value is 30. D10 contains the formula =C10+10, whose current value is 40.

Because dependencies among formulas do affect calculation order, dependencies will be discussed briefly here. The formula in D10 depends on the result from C10. The formula in C10 depends on the results from both A10 and B10. The formulas in column A each depend on the cell above them, ultimately depending on the constant value in A1. The formulas in column B each depend on the cell above them, ultimately depending on the constant value in B1.

This example was created by first entering the values in A1 then B1. Next, typing the formula in A2, and filling that across to B2. Then the formulas in A2 and B2 were concurrently filled down to A10:B10. Next, the formula was typed into C10, and finally the formula in D10 was entered. The application was in automatic/partial calculation mode when this information was entered.

3.5.2.1 Partial Calculation

The calculation chain might be saved after initially entering the data and saving the workbook, as follows:

```
<calcChain xmlns="…">
  <c r="D10" i="1"/>
  <c r="C10"/>
  <c r="A3"/>
  <c r="B3"/>
  <c r="A4"/>
  <c r="B4"/>
  <c r="A5"/>
  <c r="B5"/>
  <c r="A6"/>
  <c r="B6"/>
  <c r="A7"/>
  <c r="B7"/>
  <c r="A8"/>
  <c r="B8"/>
  <c r="A9"/>
  <c r="B9"/>
  <c r="A10"/>
  <c r="B10"/>
  <c r="B2"/>
  <c r="A2"/>
</calcChain>
```

Every c element represents a cell containing a formula. The first cell calculated appears first (top-to-bottom), and so on. The reference attribute r indicates the cell's address in the sheet. The index attribute i indicates the

index of the sheet with which that cell is associated. The sub-chain attribute s (not present in this first example) indicates that this cell can be treated as a sub chain of the preceding cell. Sub-chains can be useful when calculation can be multi-threaded or calculated concurrently. Whenever a cell does not contain an i or s attribute, it is understood to inherit these values from the previous cell.

Because of the way in which the workbook was initially created and saved, cell D10 should be the first cell calculated. The reason for this, which cannot be determined from examining the XML, is that cell D10 is the only cell that needs calculating, due to the partial calculation optimization. Since the cells A2:B10 and C10 were previously calculated (as a result of entering formulas in those cells), when entering the formula in D10, D10 is the only cell that needs to be calculated.

This calculation chain indicates that after D10 is calculated, C10 can be evaluated. In looking at the dependencies, it is understood that during a full calculation, C10 would be evaluated before D10 can be evaluated. However, because of the partial calculation optimization, at the time C10 was entered, it was placed first on the calculation chain to be evaluated. Subsequent to that, D10 was entered, and so C10 was moved to second position in the calculation chain, and that is why it is currently in the second place.

Moving through the rest of the cells with this same logic, just before C10 was entered, A3, then B3, then A4, then B4, and so on up to A10 and B10 were added and then evaluated as part of the fill-down operation.

Finally, cells A2 and B2 were the first formulas to be added and calculated. All formulas in the workbook were added after A2 and B2 were evaluated. Since A2 and B2 didn't need to be re-evaluated (due to the partial calculation optimization) after that, they eventually settled to the end of the calculation chain.

3.5.2.2 First Full Calculation

Below is how the calculation chain will look after changing the values of A1 and B1, or after forcing the application to perform a full calculation on the entire set of formulas:

```
<calcChain xmlns="…">
  <c r="B2" i="1"/>
  <c r="B3" s="1"/>
  <c r="B4" s="1"/>
  <c r="B5" s="1"/>
  <c r="B6" s="1"/>
  <c r="B7" s="1"/>
  <c r="B8" s="1"/>
  <c r="B9" s="1"/>
  <c r="B10" s="1"/>
  <c r="C10" s="1"/>
  <c r="D10" s="1"/>
  <c r="A2"/>
  <c r="A3" s="1"/>
  <c r="A4" s="1"/>
```

```
    <c r="A5" s="1"/>
    <c r="A6" s="1"/>
    <c r="A7" s="1"/>
    <c r="A8" s="1"/>
    <c r="A9" s="1"/>
    <c r="A10" s="1"/>
  </calcChain>
```

Now the order of calculation seems more in line with the way in which the formulas depend on each other: cells B2:B10 are calculated in order, and cells A2:A10 are calculated in order.

Additionally, the application has discovered that the formulas in column B can be calculated in parallel with the formulas in column A (i.e., they don't depend on each other). This is evidenced by the presence of s="1" on the cell element for cells B2:B10, indicating that B2:10 are part of a "child-chain" starting with B2. Note also that C10 and B10 are included in that child chain, even though these formulas do, in fact, depend on calculated values from column A. This is due to the multi-threaded nature of the calculation engine. Currently the chain on which C10 and D10 reside can be calculated concurrently with the chain on which A2:A10 reside because by the time C10 and D10 need to be calculated (e.g., by CPU #1), A10 has already been calculated (e.g., by CPU #2). In some future calculation, the timing may be different, and at that time, the application will need to resort to moving C10 and D10 to a new calculation level (see §3.5.2.3).

3.5.2.3 Twentieth 20th Full Calculation

After several full calculation iterations, this particular calculation chain will settle into a stable state. For example:

```
  <calcChain xmlns="…">
    <c r="B2" i="1"/>
    <c r="B3" s="1"/>
    <c r="B4" s="1"/>
    <c r="B5" s="1"/>
    <c r="B6" s="1"/>
    <c r="B7" s="1"/>
    <c r="B8" s="1"/>
    <c r="B9" s="1"/>
    <c r="B10" s="1"/>
    <c r="A2"/>
    <c r="A3" s="1"/>
    <c r="A4" s="1"/>
    <c r="A5" s="1"/>
    <c r="A6" s="1"/>
```

```
      <c r="A7" s="1"/>
      <c r="A8" s="1"/>
      <c r="A9" s="1"/>
      <c r="A10" s="1"/>
      <c r="C10" l="1"/>
      <c r="D10" s="1"/>
    </calcChain>
```

The difference introduced here is the concept of a dependency-level attribute `l`. This flag indicates that all chain and child chain concurrent calculation must be completed (and all cells will have newly calculated values) before proceeding with calculation.

In this example, cells `C10` and `D10` are marked to exist in a new and separate dependency level from the cells `A2:A10` and `B2:B10`. This makes sense given how the dependencies for these formulas are set up: `A2:A10` can be calculated concurrently with `B2:B10` because they do not depend on each other. `A2:A10` exists as one calculation chain, and `B2:B10` exist as another parallel calculation chain. However, `C10` and `D10` are both dependent on calculated results from the two parallel chains, and so can only be calculated after the first set of parallel calculations are completed.

Dependency-Level `l` flags indicate where calculation must wait for all concurrent threads to complete before continuing with calculation

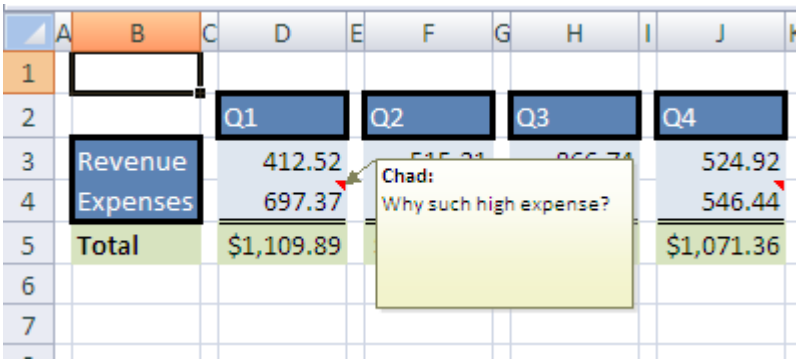
3.6 Comments

3.6.1 Overview

A *comment* is a rich text note that is attached to, and associated with, a cell, separate from other cell content. Comment content is stored separate from the cell, and is displayed in a drawing object (like a text box) that is separate from, but associated with, a cell. Comments are used as reminders, such as noting how a complex formula works, or to provide feedback to other users. Comments can also be used to explain assumptions made in a formula or to call out something special about the cell.

3.6.2 Example

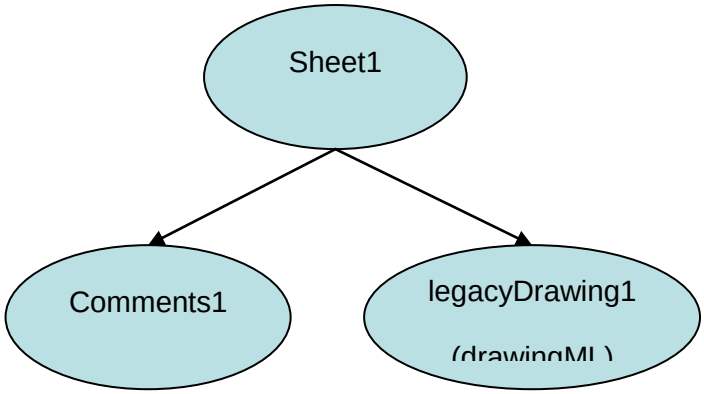
Consider the following graphic representation of a worksheet:



Note that cells D4 and J4 have comments.

3.6.3 File Architecture

Inside the file, the comment content ("Comments1") is expressed separately from the sheet information ("Sheet1"), and separately from the drawing information ("legacyDrawing1") for the containing object:



The parts are related using relationships from the sheet to the comments and drawings parts. Comments are stored together at the sheet-level. Therefore, if there are five worksheets in a workbook, and three of those contain cells with associated comments, there will be three comment parts in the file, one for each sheet.

3.6.4 The XML

```
<comments>
  <authors>
    <author>Chad</author>
    <author>CBR</author>
  </authors>
  <commentList>
    <comment ref="D4" authorId="0">
      <text>
        <r>
          <rPr>
            <b/>
            <sz val="8"/>
            <color indexed="81"/>
            <rFont val="Calibri"/>
            <charset val="1"/>
            <scheme val="minor"/>
          </rPr>
          <t>Chad:</t>
        </r>
```

```
            <r>
              <rPr>
                <sz val="8"/>
                <color indexed="81"/>
                <rFont val="Calibri"/>
                <charset val="1"/>
                <scheme val="minor"/>
              </rPr>
              <t xml:space="preserve">Why such high expense?</t>
            </r>
          </text>
        </comment>
        <comment ref="J4" authorId="1">
          <text>
            <r>
              <rPr>
                <b/>
                <sz val="8"/>
                <color indexed="81"/>
                <rFont val="Calibri"/>
                <charset val="1"/>
                <scheme val="minor"/>
              </rPr>
              <t>CBR:</t>
            </r>
            <r>
              <rPr>
                <sz val="8"/>
                <color indexed="81"/>
                <rFont val="Calibri"/>
                <charset val="1"/>
                <scheme val="minor"/>
              </rPr>
              <t xml:space="preserve">
                Pending a couple expenses in December.</t>
            </r>
          </text>
        </comment>
      </commentList>
    </comments>
```

3.6.5 Authors

```
<comments>
  <authors>
    <author>Chad</author>
    <author>CBR</author>
  </authors>
```

The authors collection is a unique list of author names for all comments on a particular sheet. In this example, there are two authors listed. Each comment definition references the authors collection by zero-based index.

3.6.6 Comments

```
<commentList>
  <comment ref="D4" authorId="0">
    <text>
      <r>
        <rPr>
          <b/>
          <sz val="8"/>
          <color indexed="81"/>
          <rFont val="Calibri"/>
          <charset val="1"/>
          <scheme val="minor"/>
        </rPr>
        <t>Chad:</t>
      </r>
      <r>
        <rPr>
          <sz val="8"/>
          <color indexed="81"/>
          <rFont val="Calibri"/>
          <charset val="1"/>
          <scheme val="minor"/>
        </rPr>
        <t xml:space="preserve">
          Why such high expense?</t>
      </r>
    </text>
  </comment>
```

commentList is a listing of all comments on a sheet. The first comment in this example has ref="D4" and authorId="0". This indicates that the comment is associated with cell D4 and is associated with the author "Chad".

The content of comment is rich text, following the rich text schematics, including the author name and actual comment.

3.7 Styles

3.7.1 Overview

There are several ways to express formatting applied to objects in a worksheet. SpreadsheetML supports the concepts of Styles, Themes, and Direct Formatting applied to cell ranges, Tables, PivotTables, Charts, and Shapes.

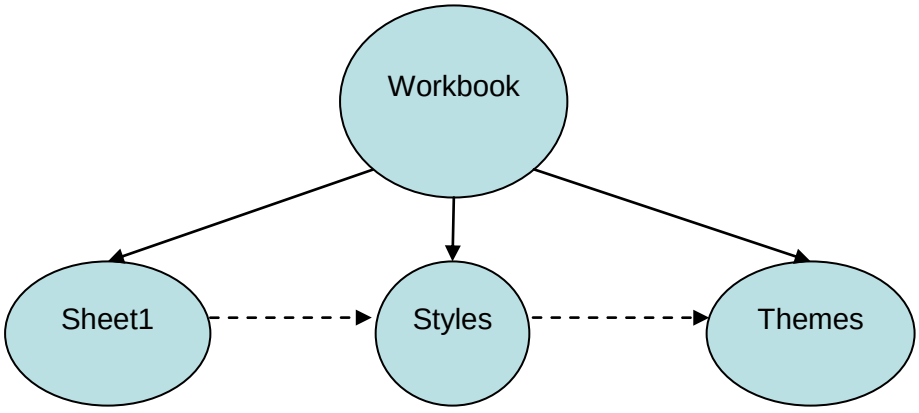
A *Style* is a named collection of formatting elements. A *cell style* can specify number format, cell alignment, font information, cell border specifications, colors, and background / foreground fills. *Table styles* specify formatting elements for the regions of a table (e.g. make the header row & totals bold face, and apply light gray fill to alternating rows in the data portion of the table to achieve striped or banded rows). *PivotTable styles* specify formatting elements for the regions of a PivotTable (e.g. 1st & 2nd level subtotals, row axis, column axis, and page fields).

A *Style* can specify color, fonts, and shape effects directly, or these elements can be referenced indirectly by referring to a *Theme* definition. Using *styles* allows for quicker application of formatting and more consistently stylized documents.

Themes define a set of colors, font information, and effects on shapes (including Charts). If a style or formatting element defines its color, font, or effect by referencing a theme, then picking a new theme switches all the colors, fonts, and effects for that formatting element.

Applying *Direct Formatting* means that particular elements of formatting (e.g. a bold font face or a number format) have been applied, but the elements of formatting have been chosen individually instead of collectively by choosing a named *Style*. Note that when applying direct formatting, themes can still be referenced, causing those elements to change as the theme is changed.

3.7.2 File Architecture



For a workbook, a single Styles part holds all its formatting definitions. Similarly, a single Themes part defines the theme information used in the workbook. These parts are referenced by relationship from the Workbook

part. Each of the formatted objects refers by index to a master formatting definition record expressed in the Styles part. This master formatting record references additional supporting formatting element collections in the Styles part. If the formatting element in the Styles part is defined in terms of a theme, then this formatting element will reference an index to a theme element defined in the Theme part. The solid arrows denote that there are relationships expressed from the Workbook part to each of the Sheet1, Styles, and Themes parts. The dotted arrow from the Sheet1 part to the Styles part indicates that there are references in the Sheet1 part's markup that refer, by index, to elements defined in the markup in the Styles part. Similarly, the dotted arrow from the Styles part to the Themes part indicates that there are references in the Styles part's markup that refer, by index, to elements defined in the markup in the Themes part.

3.7.3 Organization in the Styles Part

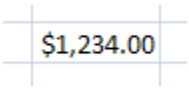
The Styles part is organized into element collections as described in the following subclauses. The element collections must appear in the order presented below. The element collections are siblings in the Styles part XML definition, whose parent, the root node of this part, is `<styleSheet>`.

Please refer to the reference material or schemas themselves for more precise descriptions on the required order of elements.

3.7.3.1 Number Format Expressions

This is where cell number formats used in this workbook are expressed. This collection never references a theme. In this collection the `numFmtId` attribute is an actual ID, unlike the other sibling collections. That is, instead of relying on the order in which a particular `numFmt` appears, it is referenced elsewhere by calling out the `numFmtId` value. Then the corresponding `numFmt` record can be found by finding the `numFmt` record with the matching `numFmtId` value. In the case of number formats, a set of `numFmtId` values are predefined and fixed by this specification. These values map to actual number formatting expressions.

The following XML would format a cell containing the value 1234 to look like this:




```
<numFmts count="1">
    <numFmt numFmtId="165" formatCode="&amp;quot;$&amp;quot;#,##0.00"/>
</numFmts>
```

A `<numFmt>` definition is referenced by ID (`numFmtId`) from either a `<cellXf>` or a `<cellStyleXf>`.

To read more about how to interpret number format codes like the value found in `formatCode` above, please read the reference section on `numFmt`, in the Styles section.

3.7.3.2 Font Definitions

This is where font definitions used in this workbook are expressed. Elements of the font definition may reference theme definitions.

```
<fonts count="1">
    <font>
          <b/>
          <sz val="11"/>
          <color theme="1"/>
          <name val="Calibri"/>
          <family val="2"/>
    </font>
</fonts>
```

This font definition specifies bold face, font size of "11", a font color specified in the Theme part, specifically the color whose index is "1" in the `<clrScheme>` collection, a font name of "Calibri", and whose font family value is "2" (for more explanation on font family, please refer to the Styles reference material). A `<font>` definition is referenced by index (`fontId`) from either a `<cellXf>` or a `<cellStyleXf>`.

A `font` record is referenced by zero-based index, meaning the numerical order in which the `font` appears under `fonts`.

3.7.3.3 Fill Definitions

This is where fills used in the workbook are expressed.

```
<fills count="1">
    <fill>
          <patternFill patternType="solid">
                <fgColor theme="4"/>
                <bgColor theme="4"/>
          </patternFill>
    </fill>
</fills>
```

This fill definition specifies a solid pattern fill, whose color uses a themed color, whose index is "4" in the `<clrScheme>` collection of the Theme part. A `<fill>` definition is referenced by index (`fillId`) from either a `<cellXf>` or a `<cellStyleXf>`.

A `fill` record is referenced by zero-based index, meaning the numerical order in which the `fill` appears under `fills`.

3.7.3.4 Borders Definitions

This is where border formats are specified.

```
<borders count="1">
    <border>
          <left/>
```

```
            <right/>
            <top/>
            <bottom/>
            <diagonal/>
        </border>
    </borders>
```

This example specifies a cell with left, right, top, and bottom borders. A `<border>` definition is referenced by index (`borderId`) from either a `<cellXf>` or a `<cellStyleXf>`.

A `border` record is referenced by zero-based index, meaning the numerical order in which the `border` appears under `borders`.

3.7.3.5 Master Records - Cell Styles

The 'master' cell style record (`<xf>`) ties together all the formatting (e.g. number format, font information, and fill) for a named cell style. An `<xf>` inside `<cellStyleXfs>` is referenced by zero-based index (not ID) (`xfId`) from a `<cellStyle>` definition, which names a particular cell style.

```
    <cellStyleXfs count="1">
        <xf numFmtId="0" fontId="0" fillId="0" borderId="0"/>
    </cellStyleXfs>
```

3.7.3.6 Master Records - Formatting

The 'master' cell style record (`<xf>`) ties together all the formatting (e.g. number format, font information, and fill) for a cell's direct formatting. An `<xf>` inside `<cellXfs>` is referenced by zero-based index (not ID) (`s`) from a cell definition (`<c>`) in one of the sheets.

```
    <cellXfs count="1">
        <xf numFmtId="0" fontId="0" fillId="0" borderId="0" xfId="0"/>
    </cellXfs>
```

3.7.3.7 Cell Styles

This is a collection of cell styles used in the workbook.

```
    <cellStyles count="1">
        <cellStyle name="Accent1" xfId="1" builtinId="29"/>
    </cellStyles>
```

3.7.3.8 Differential Formatting Records

"Differential formatting" enables subsets of formatting to be specified, without overriding other elements of formatting. For example, if it is desired to express "add bold face to whatever formatting is already there", then a `<dxf>` definition can be used. `<dxf>` definitions are used to express additional (or "differential") formatting that will be applied via Table styles or PivotTable styles. `<dxf>` definitions are referenced by index

(dxfId) from a <tableStyleElement>. The formatting elements used in a <dxf> definition are subsets of formatting collections described above.

A dxf record is referenced by zero-based index, meaning the numerical order in which the dxf appears under dxfs.

```
<dxfs count="1">
    <dxf>
          <font>
                <b/>
                <color theme="0"/>
          </font>
          <fill>
                <patternFill patternType="solid">
                      <fgColor theme="5"/>
                      <bgColor theme="5"/>
                </patternFill>
          </fill>
    </dxf>
</dxfs>
```

3.7.3.9 Custom Table Style Definitions

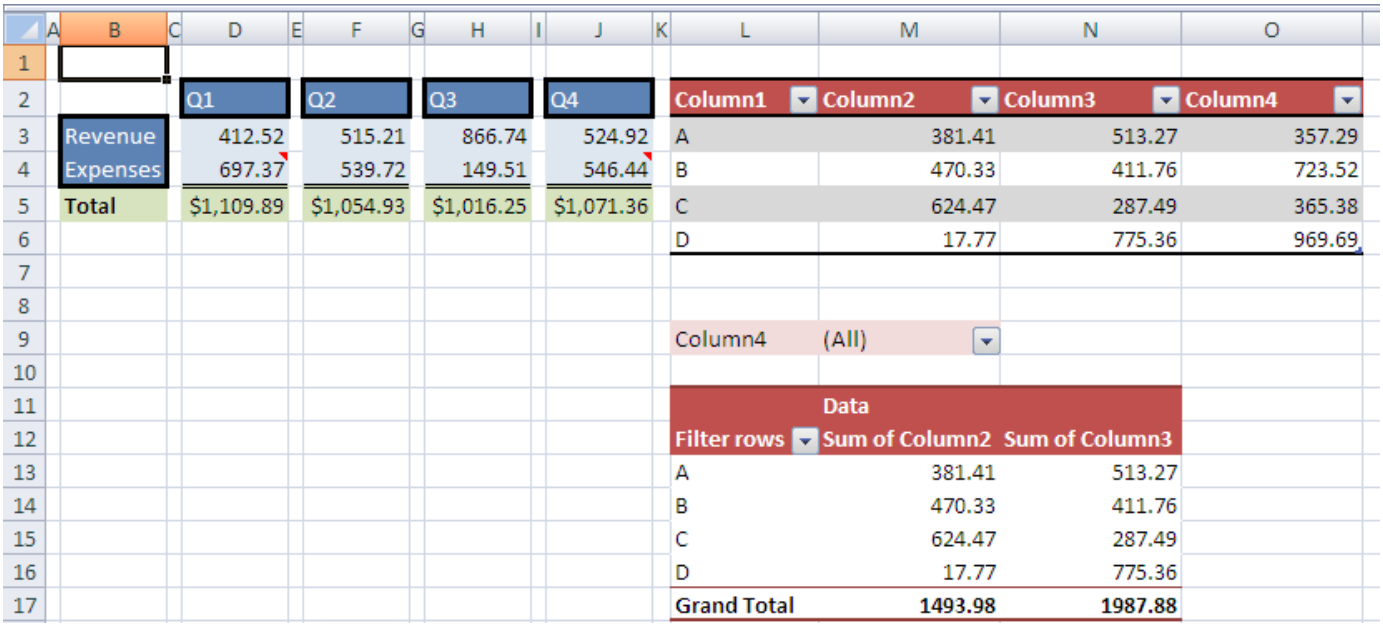
Built-in Table and PivotTable styles are not saved out, only custom-defined styles are saved out. In this example, a custom table style defines formatting for an element of a table, the "whole table" region.

```
<tableStyles count="1" defaultTableStyle="TableStyleMedium9"
defaultPivotStyle="PivotStyleLight16">
    <tableStyle name="TableStyleMedium10 - Custom" pivot="0" count="1">
          <tableStyleElement type="wholeTable" dxfId="6"/>
    </tableStyle>
</tableStyles>
```

3.7.4 Example

3.7.4.1 Illustration

For this example, consider this graphic representation of a worksheet:



		Q1	Q2	Q3	Q4
	Revenue	412.52	515.21	866.74	524.92
	Expenses	697.37	539.72	149.51	546.44
	Total	$1,109.89	$1,054.93	$1,016.25	$1,071.36

Column1	Column2	Column3	Column4
A	381.41	513.27	357.29
B	470.33	411.76	723.52
C	624.47	287.49	365.38
D	17.77	775.36	969.69

Column4	(All)

	Data	
Filter rows	Sum of Column2	Sum of Column3
A	381.41	513.27
B	470.33	411.76
C	624.47	287.49
D	17.77	775.36
Grand Total	1493.98	1987.88

Looking at the top left region of the illustration, cells D2, F2, H2, J2, and B3:B4 have the cell style "Accent1" applied to them. "Accent1" is a theme-driven style, and results in a blue cell fill and white / Calibri font formatting. Additionally these cells have direct border formatting applied which isn't specified as part of the "Accent1" cell style.

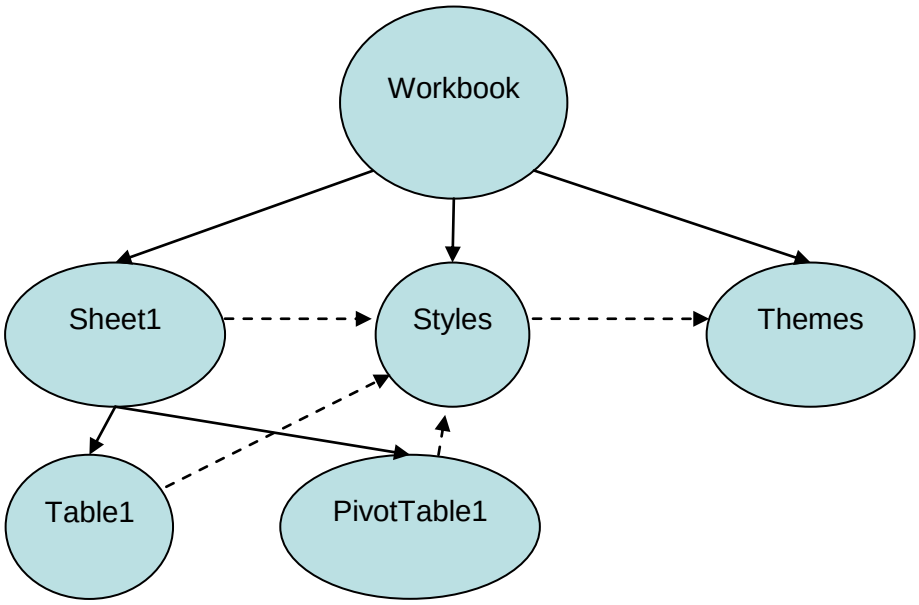
Cells D3:D4, F3:F4, H3:H4, and J3:J4 have a light blue cell fill applied. The light blue color is part of a themed color scheme, and will update when a new theme is selected.

Cells D5, F5, H5, and J5 have a currency number format applied as well as a green cell fill. While the cell fill is a themed color, the number format is fixed and will not vary or change if a new theme is selected.

The table in L2:O6 has a table style applied, called "TableStyleMedium10", which specifies formatting for the header row, row striping, and total row (even though the total row isn't shown in this example).

The PivotTable in L9:N17 has a PivotTable style applied, called "PivotStyleMedium10", which specifies formatting for the regions of a PivotTable, including the page field area in L9:M9, the header row area in L11:N12, the totals row in L17:N17, and the body of data in L13:N16.

3.7.4.2 File Architecture



All of the cells illustrated are defined in the "Sheet1" part in this example. The table is defined in the "Table1" part and the PivotTable is defined in the part named "PivotTable1". Each of the formatted objects refers to a set of formatting definitions which are expressed in the "Styles" part. If the formatting element is part of a themed set, the element will reference a theme element defined in the "Themes" part. The solid arrows represent relationships among the parts, the dotted arrows represent references by Id or index to various elements in the target part.

3.7.4.3 The XML For This Example

```
<styleSheet>
    <numFmts count="1">
          <numFmt numFmtId="164" formatCode="&quot;$&quot;#,##0.00"/>
    </numFmts>
    <fonts count="5">
          <font>
                <sz val="11"/>
                <color theme="1"/>
                <name val="Calibri"/>
                <scheme val="minor"/>
          </font>
          <font>
                <b/>
                <sz val="11"/>
                <color theme="1"/>
                <name val="Calibri"/>
                <family val="2"/>
          </font>
```

```
            <font>
                  <b/>
                  <sz val="8"/>
                  <color indexed="81"/>
                  <name val="Calibri"/>
                  <charset val="1"/>
                  <scheme val="minor"/>
            </font>
            <font>
                  <sz val="8"/>
                  <color indexed="81"/>
                  <name val="Calibri"/>
                  <charset val="1"/>
                  <scheme val="minor"/>
            </font>
            <font>
                  <sz val="11"/>
                  <color theme="0"/>
                  <name val="Calibri"/>
                  <scheme val="minor"/>
            </font>
      </fonts>
      <fills count="5">
            <fill>
                  <patternFill patternType="none"/>
            </fill>
            <fill>
                  <patternFill patternType="gray125"/>
            </fill>
            <fill>
                  <patternFill patternType="solid">
                        <fgColor theme="4"/>
                        <bgColor theme="4"/>
                  </patternFill>
            </fill>
            <fill>
                  <patternFill patternType="solid">
                        <fgColor theme="6" tint="0.59999389629810485"/>
                        <bgColor indexed="65"/>
                  </patternFill>
            </fill>
            <fill>
                  <patternFill patternType="solid">
```

```
                        <fgColor theme="4" tint="0.79998168889431442"/>
                        <bgColor indexed="65"/>
                  </patternFill>
            </fill>
      </fills>
      <borders count="5">
            <border>
                  <left/>
                  <right/>
                  <top/>
                  <bottom/>
                  <diagonal/>
            </border>
            <border>
                  <left/>
                  <right/>
                  <top/>
                  <bottom style="double">
                        <color indexed="64"/>
                  </bottom>
                  <diagonal/>
            </border>
            <border>
                  <left style="thick">
                        <color auto="1"/>
                  </left>
                  <right style="thick">
                        <color auto="1"/>
                  </right>
                  <top style="thick">
                        <color auto="1"/>
                  </top>
                  <bottom style="thick">
                        <color auto="1"/>
                  </bottom>
                  <diagonal/>
            </border>
            <border>
                  <left style="thick">
                        <color auto="1"/>
                  </left>
                  <right style="thick">
                        <color auto="1"/>
```

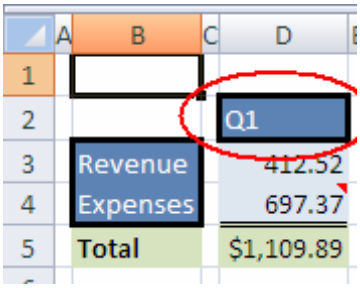
```
                </right>
                <top style="thick">
                      <color auto="1"/>
                </top>
                <bottom/>
                <diagonal/>
          </border>
          <border>
                <left style="thick">
                      <color auto="1"/>
                </left>
                <right style="thick">
                      <color auto="1"/>
                </right>
                <top/>
                <bottom style="thick">
                      <color auto="1"/>
                </bottom>
                <diagonal/>
          </border>
    </borders>
    <cellStyleXfs count="2">
          <xf numFmtId="0" fontId="0" fillId="0" borderId="0"/>
          <xf numFmtId="0" fontId="4" fillId="2" borderId="0"
applyNumberFormat="0" applyBorder="0" applyAlignment="0" applyProtection="0">
                <protection/>
          </xf>
    </cellStyleXfs>
    <cellXfs count="14">
          <xf numFmtId="0" fontId="0" fillId="0" borderId="0" xfId="0"/>
          <xf numFmtId="0" fontId="1" fillId="0" borderId="0" xfId="0"
applyFont="1"/>
          <xf numFmtId="4" fontId="0" fillId="0" borderId="0" xfId="0"
applyNumberFormat="1" applyBorder="1"/>
          <xf numFmtId="164" fontId="0" fillId="0" borderId="0" xfId="0"
applyNumberFormat="1" applyBorder="1"/>
          <xf numFmtId="0" fontId="0" fillId="0" borderId="0" xfId="0"
pivotButton="1"/>
          <xf numFmtId="0" fontId="0" fillId="0" borderId="0" xfId="0"
applyAlignment="1">
                <alignment horizontal="left"/>
          </xf>
```

```
            <xf numFmtId="0" fontId="0" fillId="0" borderId="0" xfId="0"
    applyNumberFormat="1"/>
            <xf numFmtId="0" fontId="4" fillId="2" borderId="2" xfId="1"
    applyBorder="1"/>
            <xf numFmtId="0" fontId="4" fillId="2" borderId="3" xfId="1"
    applyBorder="1"/>
            <xf numFmtId="0" fontId="4" fillId="2" borderId="4" xfId="1"
    applyBorder="1"/>
            <xf numFmtId="0" fontId="1" fillId="3" borderId="0" xfId="0"
    applyFont="1" applyFill="1"/>
            <xf numFmtId="164" fontId="0" fillId="3" borderId="0" xfId="0"
    applyNumberFormat="1" applyFill="1" applyBorder="1"/>
            <xf numFmtId="4" fontId="0" fillId="4" borderId="0" xfId="0"
    applyNumberFormat="1" applyFill="1" applyBorder="1"/>
            <xf numFmtId="4" fontId="0" fillId="4" borderId="1" xfId="0"
    applyNumberFormat="1" applyFill="1" applyBorder="1"/>
        </cellXfs>
        <cellStyles count="2">
            <cellStyle name="Accent1" xfId="1" builtinId="29"/>
            <cellStyle name="Normal" xfId="0" builtinId="0"/>
        </cellStyles>
        <dxfs count="7">
            <dxf>
                <fill>
                    <patternFill patternType="solid">
                        <fgColor theme="0" tint="-0.14999847407452621"/>
                        <bgColor theme="0" tint="-0.14999847407452621"/>
                    </patternFill>
                </fill>
            </dxf>
            <dxf>
                <fill>
                    <patternFill patternType="solid">
                        <fgColor theme="0" tint="-0.14999847407452621"/>
                        <bgColor theme="0" tint="-0.14999847407452621"/>
                    </patternFill>
                </fill>
            </dxf>
            <dxf>
                <font>
                    <b/>
                    <color theme="0"/>
                </font>
```

```
            <fill>
                  <patternFill patternType="solid">
                        <fgColor theme="5"/>
                        <bgColor theme="5"/>
                  </patternFill>
            </fill>
      </dxf>
      <dxf>
            <font>
                  <b/>
                  <color theme="0"/>
            </font>
            <fill>
                  <patternFill patternType="solid">
                        <fgColor theme="5"/>
                        <bgColor theme="5"/>
                  </patternFill>
            </fill>
      </dxf>
      <dxf>
            <border>
                  <top style="double">
                        <color theme="1"/>
                  </top>
            </border>
      </dxf>
      <dxf>
            <font>
                  <b/>
                  <color theme="0"/>
            </font>
            <fill>
                  <patternFill patternType="solid">
                        <fgColor theme="5"/>
                        <bgColor theme="5"/>
                  </patternFill>
            </fill>
            <border>
                  <bottom style="medium">
                        <color theme="1"/>
                  </bottom>
            </border>
      </dxf>
```

```
            <dxf>
                  <font>
                        <color theme="1"/>
                  </font>
                  <border>
                        <top style="medium">
                              <color theme="1"/>
                        </top>
                        <bottom style="medium">
                              <color theme="1"/>
                        </bottom>
                  </border>
            </dxf>
      </dxfs>
      <tableStyles count="1" defaultTableStyle="TableStyleMedium9"
   defaultPivotStyle="PivotStyleLight16">
            <tableStyle name="TableStyleMedium10 - Custom" pivot="0" count="7">
                  <tableStyleElement type="wholeTable" dxfId="6"/>
                  <tableStyleElement type="headerRow" dxfId="5"/>
                  <tableStyleElement type="totalRow" dxfId="4"/>
                  <tableStyleElement type="firstColumn" dxfId="3"/>
                  <tableStyleElement type="lastColumn" dxfId="2"/>
                  <tableStyleElement type="firstRowStripe" dxfId="1"/>
                  <tableStyleElement type="firstColumnStripe" dxfId="0"/>
            </tableStyle>
      </tableStyles>
      <colors/>
   </styleSheet>
```

3.7.4.4 Cell D2 Formatting



Cell D2 contains the text "Q1" and is defined in the cell table of sheet1 as:

```
   <c r="D2" s="7" t="s">
      <v>0</v>
   </c>
```

On this cell, the attribute value `s="7"` indicates that the 7th (zero-based) `<xf>` definition of `<cellXfs>` holds the formatting information for the cell. The 7th `<xf>` of `<cellXfs>` is defined as:

```
<xf numFmtId="0" fontId="4" fillId="2" borderId="2" xfId="1" applyBorder="1"/>
```

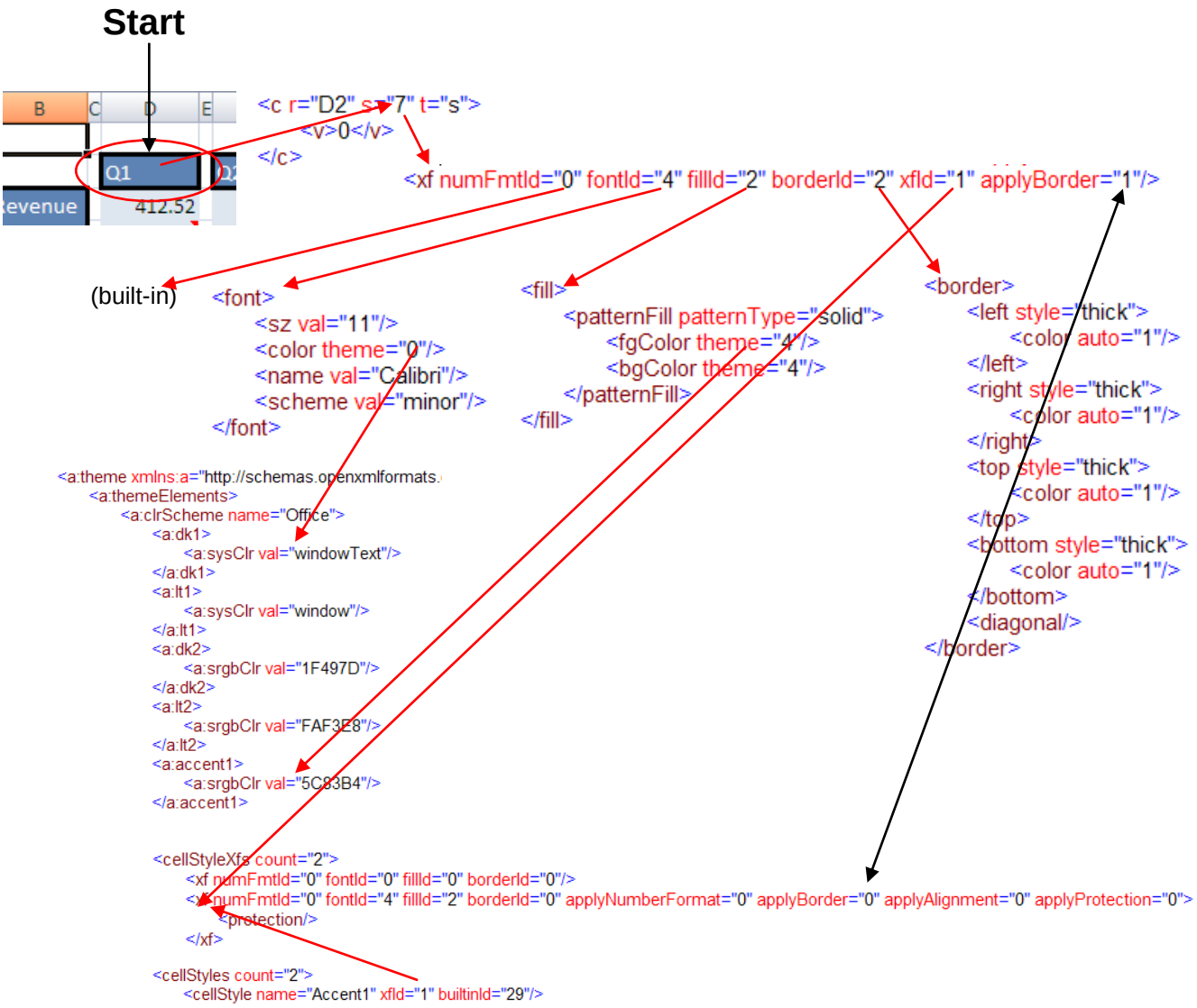
The number formatting information cannot be found in a `<numFmt>` definition because it is a built-in format; instead, it is implicitly understood to be the 0th built-in number format. Remembering that the indexes to other element collections are also zero-based, the font information can be found in the 4th `<font>` definition; the fill information in the 2nd `<fill>` definition; and the border information in the 2nd `<border>` definition. The cell uses a cell style which is defined in the 1st `<cellStyleXf>` definition and, finally, borders specified in this master formatting record should be applied.

Remember that these collections are zero-based.

Additionally the `<fill>` definition for D2 references a themed color, whose index is 4th in the `<clrScheme>` definition of the theme part:

```
<fill>
      <patternFill patternType="solid">
            <fgColor theme="4"/>
            <bgColor theme="4"/>
      </patternFill>
</fill>
```

Graphically, the index references can be shown like this:



3.7.4.5 Custom Table Style

L	M	N	O
Column1	Column2	Column3	Column4
A	381.41	513.27	357.29
B	470.33	411.76	723.52
C	624.47	287.49	365.38
D	17.77	775.36	969.69

This range of cells is a Table object with a custom Table style applied. The table definition in table1 specifies which table style is applied, and which aspects of the table style definition are 'turned on' and should be applied:

```
<table  id="2" name="Table11" displayName="Table11" ref="L20:O24"
totalsRowShown="0">
    <tableStyleInfo name="TableStyleMedium10 - Custom" showFirstColumn="0"
showLastColumn="0" showRowStripes="1" showColumnStripes="0"/>
</table>
```

The <tableStyleInfo> element indicates that this Table uses the "TableStyleMedium10 - Custom" style and that "first column", "last column", and "column stripes" formatting are OFF. It also indicates that "row stripes" formatting is ON.

Here is the "TableStyleMedium10 - Custom" definition in the Styles part:

```
    <tableStyles count="1" defaultTableStyle="TableStyleMedium9"
defaultPivotStyle="PivotStyleLight16">
        <tableStyle name="TableStyleMedium10 - Custom" pivot="0" count="7">
            <tableStyleElement type="wholeTable" dxfId="6"/>
            <tableStyleElement type="headerRow" dxfId="5"/>
            <tableStyleElement type="totalRow" dxfId="4"/>
            <tableStyleElement type="firstColumn" dxfId="3"/>
            <tableStyleElement type="lastColumn" dxfId="2"/>
            <tableStyleElement type="firstRowStripe" dxfId="1"/>
            <tableStyleElement type="firstColumnStripe" dxfId="0"/>
        </tableStyle>
    </tableStyles>
    <colors/>
</styleSheet>
```

Note that even though column stripes are defined for this table style, they are not used for this instance of the table.

The header row formatting for this table is defined by the 5th <dxf> definition:

```
        <dxf>
            <font>
                <b/>
                <color theme="0"/>
            </font>
            <fill>
                <patternFill patternType="solid">
                    <fgColor theme="5"/>
                    <bgColor theme="5"/>
                </patternFill>
            </fill>
            <border>
                <bottom style="medium">
```

```
                        <color theme="1"/>
                    </bottom>
                </border>
            </dxf>
```

This formatting indicates that for the header row of this table, the font is bold face and uses a themed color; the fill is solid and uses a themed color; and there is a bottom border on the cells.

3.8 Worksheet Metadata

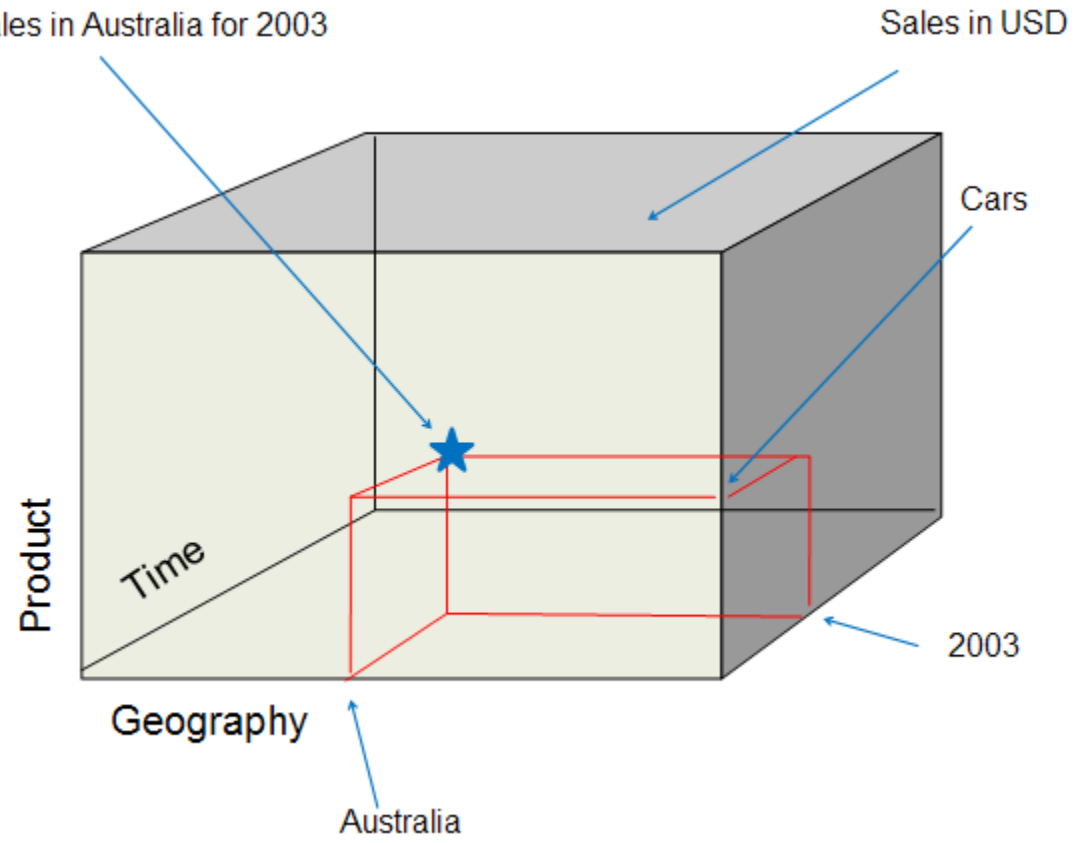
3.8.1 Overview

Value and cell metadata are additional properties that can be associated with a particular cell or value. Cell metadata properties can be carried along with the cell as it moves (e.g., via insert, shift, copy/paste, merge, or unmerge) and value metadata properties can be propagated along with the value as it is referenced in formulas.

All of this metadata is stored separately in the metadata.xml part in the workbook.

While the architecture of this feature allows for future extensions, only MDX metadata—metadata that is associated with a particular cube function and it's results—is currently defined. For example, if a CUBEMEMBER function call is used to identify a particular member in an OLAP cube, then the metadata would express the OLAP connection name, the mdx expression identifying that member, and various operational attributes of that metadata (e.g., whether it propagates through formula assignment, shifts with the cell when the cell moves locations in the grid, and so on).

3.8.1.1 OLAP Cube Review



Consider the 3-dimensional OLAP cube above. The three *dimension*s of the cube are "Product", "Geography", and "Time". "Sales Amount" is the *measure* being summarized, and is often considered an additional dimension of the cube. OLAP cubes can be N-dimensional, while this one has three dimensions.

Within each dimension are *hierarchies*, or ways of organizing the dimension into various levels of granularity. For example, within the Time dimension there can exist the Calendar Year / Quarter / Month / Day hierarchy. Likewise, the Time dimension can also have the Fiscal Year / Quarter / Month / Day hierarchy. Each of Year / Quarter / Month / Day represents *levels* in the hierarchy. '2003' is considered a *member* of the 'Year' level within the hierarchy. Likewise, the Product dimension can have multiple hierarchies. A hierarchy could be constructed based on the type of product while another hierarchy could be constructed based on the color of the product.

In the example above, picking a member from one dimension would be visualized as a slice through the cube. For example, picking 'Australia' from the Geography dimension could be a relatively thick slice of the cube, if there were many levels underneath 'Country', like 'State', 'City', and 'PostalCode'. Picking a member from Geography that is more granular than 'Australia" results in a thinner slice of the cube in the Geography dimension, because now some of Australia will have been omitted from the data.

A *tuple* is the intersection of two or more members from distinct dimensions. In the example above, three members from three dimensions are expressed:

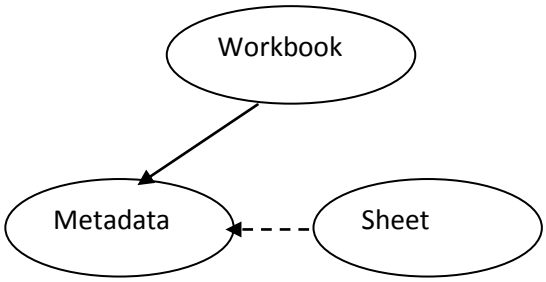
1. From Geography we have [Geography].[Country].&[Australia]
2. From Product we have [Product].[Cars].children
3. From Time we have [Time].[Calendar].[Calendar Year].&[2003]

3.8.1.2 OLAP Function Summary

There are seven recognized "CUBE" functions that can be used in cell formulas. These functions enable formulas in any cell to fetch data from Analysis Services. Specifically, the functions can fetch any member, set, aggregated value, property, or KPI from the OLAP cube. Because it is formula-driven, the layout of this data is as flexible as cells and formulas. The OLAP data can be placed anywhere on the spreadsheet, intermingled with other local calculations or within other formulas.

The function names are: CUBEKPIMEMBER, CUBEMEMBER, CUBEMEMBERPROPERTY, CUBERANKEDMEMBER, CUBESET, CUBESETCOUNT, and CUBEVALUE.

3.8.2 File Architecture – Relationships

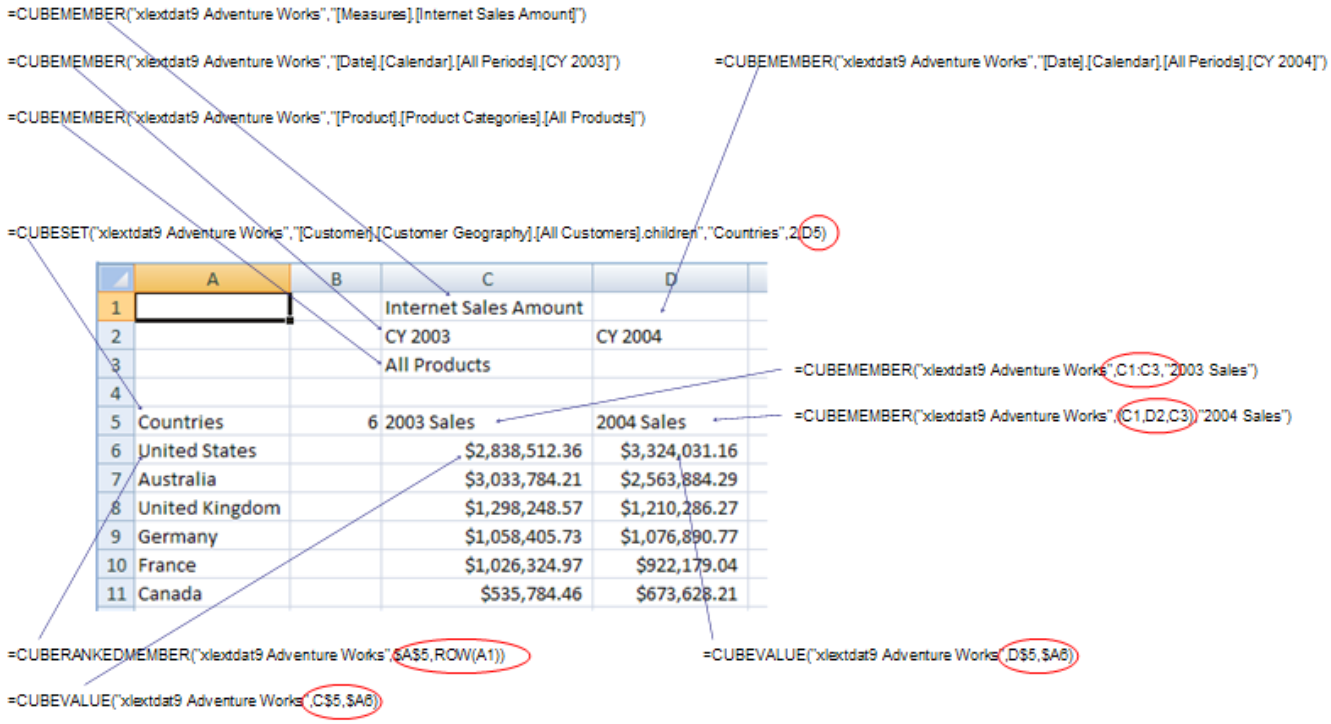


The workbook holds the relationship to the metadata part, and cells within a sheet reference the items in the metadata part.

3.8.3 Example

In the following example, Cube functions are used to build up a report of internet sales by country, for all products, for the calendar years 2003 and 2004.

3.8.3.1 Illustration



	A	B	C	D
1			Internet Sales Amount	
2			CY 2003	CY 2004
3			All Products	
4				
5	Countries	6	2003 Sales	2004 Sales
6	United States		$2,838,512.36	$3,324,031.16
7	Australia		$3,033,784.21	$2,563,884.29
8	United Kingdom		$1,298,248.57	$1,210,286.27
9	Germany		$1,058,405.73	$1,076,890.77
10	France		$1,026,324.97	$922,179.04
11	Canada		$535,784.46	$673,628.21

3.8.3.1.1 Function Summary

CUBEMEMBER(*connection*, *member-expression*, *caption*) returns a member in the cube (e.g., Bicycles, Cars, All Products, CY 2004) (member can be a tuple)

CUBESET(*connection*, *set-expression*, *caption*, *sort-order*, *sort-by*) returns a set of members (e.g., all Countries)

CUBERANKEDMEMBER(*connection*, *set-expression*, *rank*, *caption*) returns one member of the referenced set (e.g., Australia)

CUBEVALUE(*connection*, *member-expression-1*, *member-expression-2*, …) returns the aggregate summarized value for the intersection of members specified.

3.8.3.1.2 Walk Through

C1 contains a CUBEMEMBER function call specifying the "Internet Sales Amount" member from the Measures dimension. C2 contains a CUBEMEMBER function call specifying "CY 2003" from the Date dimension. D2 contains a similar function specifying "CY2004" from the Date dimension. C3 contains a CUBEMEMBER function call specifying "All Products" from the Product dimension. Each of these cells contain simple string values (e.g., "All Products" in C3), and each of these cells is associated with mdx metadata which specifies the mdx expression identifying a particular member of a particular dimension (e.g., [Measures].[Internet Sales Amount]).

A5 contains a CUBESET function call specifying a set of members. Additionally, the CUBESET function call allows for specifying a caption for the cell ("Countries"), a sort order for sorting the set (in this case, "2" corresponds to descending), and a sort by field (in this case the set will be sorted by the member as expressed in the mdx associated with cell D5, labeled "2004 Sales"). Finally, it should be noted that CUBESET returns a set of members, not just a single member.

Cells A6:A11 use the CUBERANKEDMEMBER function to return the individual members, by rank, returned from the CUBESET function call in A5. For example, A6 uses the "xlextdat9 Adventure Works" connection to connect to the OLAP cube, and addresses the first member (because "ROW(A1)" resolves to "1") in the set returned in A5.

Cell C6 uses the CUBEVALUE function to return measure data identified by intersecting the mdx expression found in A6 with the mdx expression found in C5 ("CY 2003 Internet Sales for All Products in the United States"). C7:C11 use similar CUBEVALUE function calls. D6:D11 involves similar functions as well, but using "CY 2004" instead.

The power of metadata in this example is that anytime a CUBE function argument referenced another cell, and that argument requires a set or member mdx expression, the mdx metadata for that referenced cell is returned to the calling function instead of the simple string value. For example, A6 contains a CUBERANKEDMEMBER function call, whose second argument is expecting a set of members. The reference for that argument is $A$5. Instead of using the A5's string value of "Countries" (which would result in a error), A5's mdx expression metadata is used instead, which returns a set. Similarly any of the CUBEVALUE function calls rely on cell references, where those cells contain mdx metadata used to pinpoint the measure data requested. Furthermore, each of the cells referenced by C6:D11, in turn reference other cells' mdx metadata. In this way, the mdx metadata is able to propagate through the formula calculation chain.

3.8.3.2 Worksheet Metadata XML

3.8.3.2.1 General Organization

```
<metadata>
  <metadataTypes/>
  <metadataStrings/>
  <mdxMetadata/>
  <valueMetadata/>
</metadata>
```

There are four general collections in the metadata part:

- metadataTypes - expresses various application runtime behaviors that apply to a set of metadata.
- metadataStrings - expresses supporting string resources for the metadata part. This includes the connection name to the OLAP cube as well as mdx expressions identifying members and sets.
- mdxMetadata - expresses the tuples in use in this workbook.
- valueMetadata - The block (bk) elements stored in valueMetadata are referenced from cells in the sheet definition (vm on the cell is an index (1-based) to a bk element). Each record in a block

references additional element collections in the metadata part to define fully the metadata associated with this particular record, and therefore the full metadata definition for a particular cell's value. The records in valueMetadata serve as the bridge between the metadata definitions and the cells or values in the sheet.

3.8.3.2.2 Metadata Behaviors

The metadata type expresses operations on cells that allow the metadata to remain associated with the cell. Operations not listed or set to '0' would cause the metadata to no longer be associated with the cell.

```
<metadataTypes count="1">
  <metadataType name="XLMDX" minSupportedVersion="120000" copy="1"
    pasteAll="1" pasteValues="1" merge="1" splitFirst="1"
    rowColShift="1" clearFormats="1" clearComments="1" assign="1"
    coerce="1"/>
</metadataTypes>
```

Regarding metadataTypes:

- count is the number of metadataType elements.
- type is a particular set of cell operations.

Regarding metadataType:

- name is the name of this particular metadata type.
- minSupportedVersion indicates the earliest version of the application which supports this metadata type.
- copy value of 1 indicates that this metadata will be copied to other cells when the cell is copied.
- pasteAll value of 1 indicates that this metadata will be pasted to another cell when 'paste all' is chosen during a copy/paste operation.
- pasteValues value of 1 indicates that this metadata will be pasted to another cell when only the values of the cell is pasted during a copy/paste operation.
- merge value of 1 indicates that when the cell is merged, the metadata associated with the cell remains.
- splitFirst value of 1 indicates that when a merged cell is split, the metadata associated with the merged cell is only applied to the first (from top left) cell resulting from the split.
- rowColShift value of 1 indicates that metadata associated with a cell remains after rows and columns are inserted, even when the cell is moved.
- clearFormats value of 1 indicates that the metadata remains after the cell has been cleared of all formatting.
- clearComments value of 1 indicates that the metadata remains after comments have been cleared from the cell.
- assign value of 1 indicates that the metadata propagates through formula assignment operations

- coerce value of 1 indicates that the metadata can be removed when the data type is coerced to another type.

3.8.3.2.3 Metadata Strings

This collection is a set of string resources for the metadata part. Most follow the format of an mdx expression. Connection names (to OLAP cubes) are also expressed here.

```
<metadataStrings count="12">
  <s v="xlextdat9 Adventure Works"/>
  <s v="[Measures].[Internet Sales Amount]"/>
  <s v="[Date].[Calendar].[Calendar Year].&amp;[2003]"/>
  <s v="[Date].[Calendar].[Calendar Year].&amp;[2004]"/>
  <s v="[Product].[Product Categories].[All Products]"/>
  <s v="[Customer].[Customer Geography].[All Customers].children"/>
  <s v="[Customer].[Customer Geography].[Country].&amp;[Australia]"/>
  <s v="[Customer].[Customer Geography].[Country].&amp;
    [United States]"/>
  <s v="[Customer].[Customer Geography].[Country].&amp;
    [United Kingdom]"/>
  <s v="[Customer].[Customer Geography].[Country].&amp;[Germany]"/>
  <s v="[Customer].[Customer Geography].[Country].&amp;[France]"/>
  <s v="[Customer].[Customer Geography].[Country].&amp;[Canada]"/>
</metadataStrings>
```

Regarding metadataStrings:

- count indicates the number of strings in the collection.
- s is the string container element
- v is the string value itself.

3.8.3.2.4 mdxMetadata

This collection expresses mdx metadata, and builds up the mdx members, sets, KPIs, and member properties.

valueMetadata records reference these records.

```
<mdxMetadata count="26">
  <mdx n="0" f="m">
    <t c="1">
      <n x="1"/>
    </t>
  </mdx>
```

```
      <mdx n="0" f="m">
        <t c="1">
          <n x="2"/>
        </t>
      </mdx>
      <mdx n="0" f="m">
        <t c="1">
          <n x="3"/>"
        </t>
      </mdx>
      <mdx n="0" f="m">
        <t c="1">
          <n x="4"/>
        </t>
      </mdx>
      <mdx n="0" f="m">
        <t c="3">
          <n x="1"/>
          <n x="2"/>
          <n x="4"/>
        </t>
      </mdx>
      <mdx n="0" f="m">
        <t c="3">
          <n x="1"/>
          <n x="3"/>
          <n x="4"/>
        </t>
      </mdx>
      <mdx n="0" f="r">
        <t c="1">
          <n x="6"/>
        </t>
      </mdx>
      <mdx n="0" f="r">
        <t c="1">
          <n x="7"/>
        </t>
      </mdx>
```

```
<mdx n="0" f="r">
  <t c="1">
    <n x="8"/>
  </t>
</mdx>
<mdx n="0" f="r">
  <t c="1">
    <n x="9"/>
  </t>
</mdx>
<mdx n="0" f="r">
  <t c="1">
    <n x="10"/>
  </t>
</mdx>
<mdx n="0" f="r">
  <t c="1">
    <n x="11"/>
  </t>
</mdx>
<mdx n="0" f="v">
  <t c="4" ct="en-US">
    <n x="1"/>
    <n x="2"/>
    <n x="4"/>
    <n x="6"/>
  </t>
</mdx>
<mdx n="0" f="v">
  <t c="4" ct="en-US">
    <n x="1"/>
    <n x="3"/>
    <n x="4"/>
    <n x="7"/>
  </t>
</mdx>
```

```
      <mdx n="0" f="v">
        <t c="4" ct="en-US">
          <n x="1"/>
          <n x="2"/>
          <n x="4"/>
          <n x="7"/>
        </t>
      </mdx>
      <mdx n="0" f="v">
        <t c="4" ct="en-US">
          <n x="1"/>
          <n x="3"/>
          <n x="4"/>
          <n x="8"/>
        </t>
      </mdx>
      <mdx n="0" f="v">
        <t c="4" ct="en-US">
          <n x="1"/>
          <n x="2"/>
          <n x="4"/>
          <n x="8"/>
        </t>
      </mdx>
      <mdx n="0" f="v">
        <t c="4" ct="en-US">
          <n x="1"/>
          <n x="3"/>
          <n x="4"/>
          <n x="9"/>
        </t>
      </mdx>
      <mdx n="0" f="v">
        <t c="4" ct="en-US">
          <n x="1"/>
          <n x="2"/>
          <n x="4"/>
          <n x="9"/>
        </t>
      </mdx>
```

```
      <mdx n="0" f="v">
        <t c="4" ct="en-US">
          <n x="1"/>
          <n x="3"/>
          <n x="4"/>
          <n x="10"/>
        </t>
      </mdx>
      <mdx n="0" f="v">
        <t c="4" ct="en-US">
          <n x="1"/>
          <n x="2"/>
          <n x="4"/>
          <n x="10"/>
        </t>
      </mdx>
      <mdx n="0" f="v">
        <t c="4" ct="en-US">
          <n x="1"/>
          <n x="3"/>
          <n x="4"/>
          <n x="11"/>
        </t>
      </mdx>
      <mdx n="0" f="v">
        <t c="4" ct="en-US">
          <n x="1"/>
          <n x="2"/>
          <n x="4"/>
          <n x="11"/>
        </t>
      </mdx>
      <mdx n="0" f="v">
        <t c="4" ct="en-US">
          <n x="1"/>
          <n x="3"/>
          <n x="4"/>
          <n x="6"/>
        </t>
      </mdx>
```

```
      <mdx n="0" f="s">
        <ms ns="5" c="3" o="d">
          <n x="1"/>
          <n x="3"/>
          <n x="4"/>
        </ms>
      </mdx>
      <mdx n="0" f="c">
        <ms ns="5" c="3" o="d">
          <n x="1"/>
          <n x="3"/>
          <n x="4"/>
        </ms>
      </mdx>
    </mdxMetadata>
```

Regarding mdxMetadata:

- count indicates the number of mdx statements in the collection.

Regarding mdx, which is a particular mdx statement:

- n indicates the index of the record in metadataStrings containing the connection name.
- f indicates the name of the calling cube function in the workbook.

Regarding t, which is an mdx tuple:

- c is the count of member expressions in the mdx tuple.

Regarding n:

- x is the index value into metadataStrings indicating the particular member expression for this dimension of the tuple expression.

For example, cell C5 has a CUBEMEMBER function call expressing the result of "Internet Sales Amount of All Products for CY 2003". In sheet1.xml, cell C5 has vm="5", which means it has an associated valueMetadata record whose index is "5". Looking ahead into the valueMetadata records, the 5th (1-based) record points to the 4th (zero-based) mdx collection in mdxMetadata.

The 5th mdx collection:

```
<mdx n="0" f="m">
  <t c="3">
    <n x="1"/>
    <n x="2"/>
    <n x="4"/>
  </t>
</mdx>
```

Where <n x="1"/> corresponds to the 1st position in the string store, namely

```
<s v="[Measures].[Internet Sales Amount]"/>
```

and where <n x="2"/> corresponds to the 2nd position in the string store, namely

```
<s v="[Date].[Calendar].[Calendar Year].&amp;[2003]"/>
```

and where <n x="4"/> corresponds to the 4th position in the string store, namely

```
<s v="[Product].[Product Categories].[All Products]"/>.
```

Therefore this data point in the cube is addressed by intersecting these three hierarchies, one in each dimension of the OLAP cube:

- [Measures].[Internet Sales Amount]
- [Date].[Calendar].[Calendar Year].[2003]
- [Product].[Product Categories].[All Products]

Regarding ms:

- ns is the index of the mdx set definition in the string store.
- c is the number of sort-by member indicies, in this case 3 because the set is sorted by the contents of D5, which happens to be a member defined by 3 coordinates in the cube.
- o indicates the order of the sort; in this case, 'descending'.
- n is the index indicating the mdx expressions in the string store used to identify the members used to define the sort-by set.

3.8.3.2.5 valueMetadata

This collection defines cell or value metadata information (depending on the value of metadataType's cellMeta)

```
<valueMetadata count="26">
  <bk>
    <rc t="1" v="0"/>
  </bk>
```

```
      <bk>
        <rc t="1" v="1"/>
      </bk>
      <bk>
        <rc t="1" v="2"/>
      </bk>
      <bk>
        <rc t="1" v="3"/>
      </bk>
      <bk>
        <rc t="1" v="4"/>
      </bk>
      <bk>
        <rc t="1" v="5"/>
      </bk>
      <bk>
        <rc t="1" v="6"/>
      </bk>
      <bk>
        <rc t="1" v="7"/>
      </bk>
      <bk>
        <rc t="1" v="8"/>
      </bk>
      <bk>
        <rc t="1" v="9"/>
      </bk>
      <bk>
        <rc t="1" v="10"/>
      </bk>
      <bk>
        <rc t="1" v="11"/>
      </bk>
      <bk>
        <rc t="1" v="12"/>
      </bk>
      <bk>
        <rc t="1" v="13"/>
      </bk>
      <bk>
        <rc t="1" v="14"/>
      </bk>
```

```
    <bk>
      <rc t="1" v="15"/>
    </bk>
    <bk>
      <rc t="1" v="16"/>
    </bk>
    <bk>
      <rc t="1" v="17"/>
    </bk>
    <bk>
      <rc t="1" v="18"/>
    </bk>
    <bk>
      <rc t="1" v="19"/>
    </bk>
    <bk>
      <rc t="1" v="20"/>
    </bk>
    <bk>
      <rc t="1" v="21"/>
    </bk>
    <bk>
      <rc t="1" v="22"/>
    </bk>
    <bk>
      <rc t="1" v="23"/>
    </bk>
    <bk>
      <rc t="1" v="24"/>
    </bk>
    <bk>
      <rc t="1" v="25"/>
    </bk>
  </valueMetadata>
```

Regarding valueMetadata:

- `count` indicates the number of metadata block records.

Regarding bk, which is a metadata block, and rc, which is a metadata record:

- `t` indicates the index of the metadataType record in metadataTypes collection.
- `v` is the index of metadata record value in the storage corresponding to record type.

Looking at the first block using the bk element, the type of metadata with which this record is associated is the first (and only) metadataType record, which is of type "XLMDX". This indicates that the v index is pointing to the 0th mdxMetadata record.

3.9 Pivot Table, Pivot Cache, and Common Types

3.9.1 Feature Overview

PivotTables display aggregated views of data easily and in an understandable layout. Hundreds or thousands of pieces of underlying information can be aggregated on row & column axes, revealing the meanings behind the data. PivotTable reports are used to organize and summarize your data in different ways. Creating a PivotTable report is about moving pieces of information around to see how they fit together. In a few gestures the pivot rows and columns can be moved into different arrangements and layouts.

A PivotTable object has a row axis area, a column axis area, a values area, and a report filter area. Additionally, PivotTables have a corresponding field list pane displaying all the fields of data which can be placed on one of the PivotTable areas.

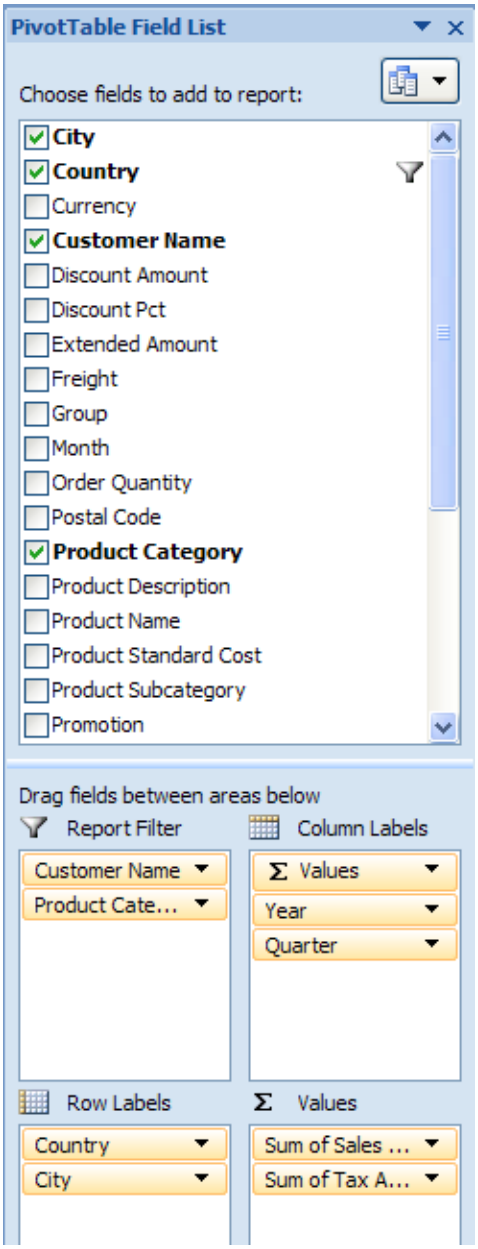
Consider this source data:

	A	C	F	H	I	O	P	Q	Z	AA	AB
1	Customer Name	Country	City	Product Category	Product Subcategory	Year	Quarter	Month	Sales Amount	Tax Amount	Freight
2	Michele Raman	Australia	Bendigo	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
3	Misty Raji	Australia	Bendigo	Bikes	Road Bikes	2001	3	July	3578.27	286.2616	89.4568
4	Tabitha E Arthur	Australia	Bendigo	Bikes	Road Bikes	2001	3	July	3578.27	286.2616	89.4568
5	Clarence D Rai	Australia	Bendigo	Bikes	Mountain Bikes	2001	3	July	3399.99	271.9992	84.9998
6	Jimmy L Moreno	Australia	Bendigo	Bikes	Mountain Bikes	2001	3	July	3399.99	271.9992	84.9998
7	Rob Verhoff	Australia	Bendigo	Bikes	Mountain Bikes	2001	3	July	3374.99	269.9992	84.3748
8	Levi Sai	Australia	Bendigo	Bikes	Road Bikes	2001	3	July	3578.27	286.2616	89.4568
9	Logan Gonzales	Australia	Brisbane	Bikes	Road Bikes	2001	3	July	3578.27	286.2616	89.4568
10	Dalton J Lee	Australia	Brisbane	Bikes	Road Bikes	2001	3	August	3578.27	286.2616	89.4568
11	Jessie J Ortega	Australia	Brisbane	Bikes	Road Bikes	2001	3	August	3578.27	286.2616	89.4568
12	Paul J. Shakespear	Australia	Caloundra	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
13	Joan R Martin	Australia	Caloundra	Bikes	Road Bikes	2001	3	September	699.0982	55.9279	17.4775
14	Casey Pal	Australia	Caloundra	Bikes	Road Bikes	2001	3	July	3578.27	286.2616	89.4568
15	Ethan G Coleman	Australia	Caloundra	Bikes	Road Bikes	2001	3	August	3578.27	286.2616	89.4568
16	Kendra Rubio	Australia	Caloundra	Bikes	Road Bikes	2001	3	August	3578.27	286.2616	89.4568
17	Bethany G Yuan	Australia	Cloverdal	Bikes	Mountain Bikes	2001	3	August	3399.99	271.9992	84.9998
18	Jasmine Wilson	Australia	Coffs Hart	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
19	Micah Wu	Australia	Coffs Hart	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
20	Warren L Zhang	Australia	Coffs Hart	Bikes	Road Bikes	2001	3	July	699.0982	55.9279	17.4775
21	Ariana Stewart	Australia	Coffs Hart	Bikes	Road Bikes	2001	3	August	3578.27	286.2616	89.4568
22	Suzanne K Lu	Australia	Coffs Hart	Bikes	Road Bikes	2001	3	August	3578.27	286.2616	89.4568
23	Randall M Rubio	Australia	Cranbourr	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
24	Deborah K Kumar	Australia	Cranbourr	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
25	Krystal Holt	Australia	Cranbourr	Bikes	Road Bikes	2001	3	July	3578.27	286.2616	89.4568
26	Patricia T Raman	Australia	Cranbourr	Bikes	Road Bikes	2001	3	August	3578.27	286.2616	89.4568
27	Wendy Dominguez	Australia	Cranbourr	Bikes	Mountain Bikes	2001	3	August	3374.99	269.9992	84.3748
28	Willie She	Australia	Darlinghu	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
29	Alan Zhu	Australia	Darlinghu	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
30	Dawn R Tang	Australia	Darlinghu	Bikes	Road Bikes	2001	3	July	3578.27	286.2616	89.4568

This data can be consolidated and summarized in a PivotTable. One way to organize the information would look like this:

	A	B	C	D	E	F	G	H
1								
2		Customer Name	(All)					
3		Product Category	(All)					
4								
5			Column Labels					
6			Sum of Sales Amount			Sum of Tax Amount		
7			2001		2001 Total	2001		2001 Total
8		Row Labels	3	4		3	4	
9		**Australia**	**606184.7066**	**702862.4912**	**1309047.198**	**48494.7771**	**56229**	**104723.7771**
10		Goulburn	40580.8864	25047.89	65628.7764	3246.471	2003.8312	5250.3022
11		Warrnambool	28091.32	27863.04	55954.36	2247.3056	2229.0432	4476.3488
12		Port Macquarie	25746.9882	24463.05	50210.0382	2059.7591	1957.044	4016.8031
13		Wollongong	24691.33	25390.4282	50081.7582	1975.3064	2031.2343	4006.5407
14		Bendigo	24488.05	17484.79	41972.84	1959.044	1398.7832	3357.8272
15		Geelong	21113.06	21088.06	42201.12	1689.0448	1687.0448	3376.0896
16		Hobart	21088.06	6953.26	28041.32	1687.0448	556.2608	2243.3056
17		Lavender Bay	21063.06	21965.4382	43028.4982	1685.0448	1757.2351	3442.2799
18		Matraville	20909.78	6799.98	27709.76	1672.7824	543.9984	2216.7808

Here is the corresponding PivotTable field list:



3.9.2 File Architecture

The workbook points to (and owns the longevity of) the pivotCacheDefinition part, which in turn points to and owns the pivotCacheRecords part. The workbook also points to and owns the sheet part, which in turn points to and owns a pivotTable part definition, when a PivotTable is on the sheet (there can be multiple PivotTables on a sheet). The pivotTable part points to the appropriate pivotCacheDefinition which it is using. Since multiple PivotTables can use the same cache, the pivotTable part does not own the longevity of the pivotCacheDefinition.

The pivotTable part describes the particulars of the layout of the PivotTable on the sheet. It indicates what fields are on the row axis, the column axis, report filter, and values areas of the PivotTable. It also indicates

formatting information about the PivotTable. If conditional formatting has been applied to the PivotTable, that is also expressed in the pivotTable part.

The pivotCacheRecords part contains the underlying data to be aggregated. It is a cache of the source data. The pivotCacheDefinition part defines each field in the pivotCacheRecords part, including field name and information about the data contained in the field. The pivotCacheDefinition part also defines pivot items that are shared among the pivotTable and pivotRecords parts.

3.9.3 Example - Native with Range Source

3.9.3.1 Illustration

Consider the source data pictured in the overview section. There are 28 fields of data in total (some aren't shown). A corresponding PivotTable summary of the data can look like this:

	A	B	C	D	E	F	G
1							
2		Country	(All)				
3		State	(All)				
4		City	(All)				
5							
6		Sum of Sales Amount	Column Labels				
7			2001				2001 Total
8			3			3 Total	
9		Row Labels	July	August	September		
10		Bikes	209652.9046	222538.2892	173993.5128	606184.7066	606184.7066
11		Mountain Bikes	64424.81	60899.82	10174.97	135499.6	135499.6
12		Road Bikes	145228.0946	161638.4692	163818.5428	470685.1066	470685.1066
13		Grand Total	209652.9046	222538.2892	173993.5128	606184.7066	606184.7066

Regarding the layout of the PivotTable, notice that "Country", "State", and "City" are in the report filter area of the PivotTable. "Product Category" and "Product Subcategory" are on the row axis ("Bikes" belongs to the "Product Category" field and both "Mountain Bikes" and "Road Bikes" belong to the "Product Subcategory" field). On the column axis are "Year" ("2001"), "Quarter" ("3"), and "Month" ("July", "August", and "September") fields.

Row Grand Totals are turned on, and column Subtotals are turned on for Quarter and Year (if there was more than 1 quarter in the source data the Year Subtotal would be more interesting).

3.9.3.2 XML - pivotCacheDefinition part

The pivotCacheDefinition part defines each field in the source data, including the name, the string resources of the instance data (for shared items), and information about the type of data appearing in the field. Note: some of the "Customer Name" and "City" values have been removed to improve readability and reduce length.

```
<pivotCacheDefinition xmlns:r="…" r:id="rId1" refreshedBy="AnonUser"
  refreshedDate="2006-05-22T10:07:16" createdVersion="3"
  refreshedVersion="3" minRefreshableVersion="3" recordCount="182">
  <cacheSource type="worksheet">
    <worksheetSource name="Table1"/>
  </cacheSource>
  <cacheFields count="28">
    <cacheField name="Customer Name" numFmtId="0">
      <sharedItems count="7">
        <s v="Michele  Raman"/>
        <s v="Misty  Raji"/>
        <s v="Tabitha E Arthur"/>
        <s v="Clarence D Rai"/>
        <s v="Jimmy L Moreno"/>
        <s v="Rob  Verhoff"/>
        <s v="Levi  Sai"/>
      </sharedItems>
    </cacheField>
    <cacheField name="Group" numFmtId="0">
      <sharedItems/>
    </cacheField>
    <cacheField name="Country" numFmtId="0">
      <sharedItems count="1">
        <s v="Australia"/>
      </sharedItems>
    </cacheField>
    <cacheField name="Region" numFmtId="0">
      <sharedItems/>
    </cacheField>
    <cacheField name="State" numFmtId="0">
      <sharedItems count="5">
        <s v="Victoria"/>
        <s v="Queensland"/>
        <s v="South Australia"/>
        <s v="New South Wales"/>
        <s v="Tasmania"/>
      </sharedItems>
    </cacheField>
```

```
        <cacheField name="City" numFmtId="0">
          <sharedItems count="7">
            <s v="Bendigo"/>
            <s v="Brisbane"/>
            <s v="Caloundra"/>
            <s v="Cloverdale"/>
            <s v="Coffs Harbour"/>
            <s v="Cranbourne"/>
            <s v="Darlinghurst"/>
          </sharedItems>
        </cacheField>
        <cacheField name="Postal Code" numFmtId="0">
          <sharedItems/>
        </cacheField>
        <cacheField name="Product Category" numFmtId="0">
          <sharedItems count="1">
            <s v="Bikes"/>
          </sharedItems>
        </cacheField>
        <cacheField name="Product Subcategory" numFmtId="0">
          <sharedItems count="2">
            <s v="Road Bikes"/>
            <s v="Mountain Bikes"/>
          </sharedItems>
        </cacheField>
        <cacheField name="Product Name" numFmtId="0">
          <sharedItems/>
        </cacheField>
        <cacheField name="Product Description" numFmtId="0">
          <sharedItems/>
        </cacheField>
        <cacheField name="Promotion Category" numFmtId="0">
          <sharedItems/>
        </cacheField>
        <cacheField name="Promotion" numFmtId="0">
          <sharedItems/>
        </cacheField>
        <cacheField name="Promotion Type" numFmtId="0">
          <sharedItems/>
        </cacheField>
```

```
<cacheField name="Year" numFmtId="0">
  <sharedItems count="1">
    <s v="2001"/>
  </sharedItems>
</cacheField>
<cacheField name="Quarter" numFmtId="0">
  <sharedItems containsSemiMixedTypes="0" containsString="0"
    containsNumber="1" containsInteger="1" minValue="3" maxValue="3"
    count="1">
    <n v="3"/>
  </sharedItems>
</cacheField>
<cacheField name="Month" numFmtId="0">
  <sharedItems count="3">
    <s v="September"/>
    <s v="July"/>
    <s v="August"/>
  </sharedItems>
</cacheField>
<cacheField name="Currency" numFmtId="0">
  <sharedItems/>
</cacheField>
<cacheField name="Order Quantity" numFmtId="0">
  <sharedItems containsSemiMixedTypes="0" containsString="0"
    containsNumber="1" containsInteger="1" minValue="1"
    maxValue="1"/>
</cacheField>
<cacheField name="Unit Price" numFmtId="0">
  <sharedItems containsSemiMixedTypes="0" containsString="0"
    containsNumber="1" minValue="699.09820000000002"
    maxValue="3578.27"/>
</cacheField>
<cacheField name="Extended Amount" numFmtId="0">
  <sharedItems containsSemiMixedTypes="0" containsString="0"
    containsNumber="1" minValue="699.09820000000002"
    maxValue="3578.27"/>
</cacheField>
<cacheField name="Discount Pct" numFmtId="0">
  <sharedItems containsSemiMixedTypes="0" containsString="0"
    containsNumber="1" containsInteger="1" minValue="0"
    maxValue="0"/>
</cacheField>
```

```
        <cacheField name="Discount Amount" numFmtId="0">
          <sharedItems containsSemiMixedTypes="0" containsString="0"
            containsNumber="1" containsInteger="1" minValue="0"
            maxValue="0"/>
        </cacheField>
        <cacheField name="Product Standard Cost" numFmtId="0">
          <sharedItems containsSemiMixedTypes="0" containsString="0"
            containsNumber="1" minValue="413.1463"
            maxValue="2171.2941999999998"/>
        </cacheField>
        <cacheField name="Total Product Cost" numFmtId="0">
          <sharedItems containsSemiMixedTypes="0" containsString="0"
            containsNumber="1" minValue="413.1463"
            maxValue="2171.2941999999998"/>
        </cacheField>
        <cacheField name="Sales Amount" numFmtId="0">
          <sharedItems containsSemiMixedTypes="0" containsString="0"
            containsNumber="1" minValue="699.09820000000002"
            maxValue="3578.27"/>
        </cacheField>
        <cacheField name="Tax Amount" numFmtId="0">
          <sharedItems containsSemiMixedTypes="0" containsString="0"
            containsNumber="1"
            minValue="55.927900000000001" maxValue="286.26159999999999"/>
        </cacheField>
        <cacheField name="Freight" numFmtId="0">
          <sharedItems containsSemiMixedTypes="0" containsString="0"
            containsNumber="1" minValue="17.477499999999999"
            maxValue="89.456800000000001"/>
        </cacheField>
      </cacheFields>
    </pivotCacheDefinition>
```

In the context of pivotCacheDefinition:

- r:id indicates the relationship id pointing to the corresponding pivotCacheRecords part.
- refreshedBy indicates the username of whomever last refreshed the PivotCache.
- refreshedDate indicates when the PivotCache was last refreshed.
- createdVersion indicates the version of the producer which created the PivotCache.
- refreshedVersion indicates the version of the producer which last refreshed the PivotCache.
- minRefreshableVersion indicates the minimum version of the producer required to be able to refresh this PivotCache.

In the context of cacheSource:

- type indicates that data in a worksheet is the source for this PivotCache.
- worksheetSource identifies the particular location of the source data. In this case, it is a named range whose name is "Table1".

In the context of cacheFields, which is a collection of all the field definitions in the source data:

- cacheField indicates the name of the field and provides number format information.

In the context of cacheField:

- sharedItems indicates various flags about the data in this field. Child elements express the values of the shared items.

In the context of sharedItems:

- containsSemiMixedTypes "1" indicates that this field contains text values possibly mixed with other types of values, this can contain blanks. In this example the value is "0".
- containsString value of "1" indicates that this field contains a text value. In this example, the value is "0".
- containsNumber value of "1" indicates that this field contains numeric values.
- containsInteger indicates that this field contains integer values.
- minValue indicates that this field's minimum value is "3".
- maxValue indicates that this field's maximum value is "3".
- s indicates string content for this item value (expressed in v).
- n indicates the numeric content for this item value (expressed in v).

If there are no shared items expressed for a particular field, then the values are expressed directly in the pivotCacheRecords part.

Items in the PivotCacheDefinition can be shared, in order to reduce the redundancy of those values, since they're referenced in multiple places across all the PivotTable parts. For example, a value might be part of a filter, it might appear on a row or column axis, and will appear in the pivotCacheRecords definition as well. However, because of the performance cost of creating the optimized shared items, items are only shared if they are actually in use in the PivotTable. Therefore, depending on user actions on the PivotTable layout, the pivotCacheDefinition and underlying PivotCacheRecords part may be updated.

3.9.3.3 XML - pivotCacheRecords part

This part expresses the underlying source data that the PivotTable is aggregating. (Note that the data has been trimmed down to two records to increase readability.)

```
<pivotCacheRecords … xmlns:r="…" count="2">
  <r>
    <x v="0"/>
    <s v="Pacific"/>
```

```
      <x v="0"/>
      <s v="Australia"/>
      <x v="0"/>
      <x v="0"/>
      <s v="3550"/>
      <x v="0"/>
      <x v="0"/>
      <s v="Road-150 Red, 62"/>
      <s v="This bike is ridden by race winners. Developed with the Adventure
Works Cycles professional race team, it has a extremely light heat-treated
aluminum frame, and steering that allows precision control."/>
      <s v="No Discount"/>
      <s v="No Discount"/>
      <s v="No Discount"/>
      <x v="0"/>
      <x v="0"/>
      <x v="0"/>
      <s v="Australian Dollar"/>
      <n v="1"/>
      <n v="3578.27"/>
      <n v="3578.27"/>
      <n v="0"/>
      <n v="0"/>
      <n v="2171.2941999999998"/>
      <n v="2171.2941999999998"/>
      <n v="3578.27"/>
      <n v="286.26159999999999"/>
      <n v="89.456800000000001"/>
    </r>
    <r>
      <x v="1"/>
      <s v="Pacific"/>
      <x v="0"/>
      <s v="Australia"/>
      <x v="0"/>
      <x v="0"/>
      <s v="3550"/>
      <x v="0"/>
      <x v="0"/>
      <s v="Road-150 Red, 44"/>
      <s v="This bike is ridden by race winners. Developed with the Adventure
Works Cycles professional race team, it has a extremely light heat-treated
aluminum frame, and steering that allows precision control."/>
```

```
        <s v="No Discount"/>
        <s v="No Discount"/>
        <s v="No Discount"/>
        <x v="0"/>
        <x v="0"/>
        <x v="1"/>
        <s v="Australian Dollar"/>
        <n v="1"/>
        <n v="3578.27"/>
        <n v="3578.27"/>
        <n v="0"/>
        <n v="0"/>
        <n v="2171.2941999999998"/>
        <n v="2171.2941999999998"/>
        <n v="3578.27"/>
        <n v="286.26159999999999"/>
        <n v="89.456800000000001"/>
      </r>
    </pivotCacheRecords>
```

In the context of pivotCacheRecords:

- r contains one record.

In the context of r:

- x is an index value referencing an item for this field, as defined in the pivotCacheDefinition part.
- s indicates that a value is being expressed inline in this record, and it is a string value.
- n indicates that a value is being expressed inline in this record, and it is a numeric value.

3.9.3.4 XML - pivotTable part

The pivotTable part is organized into 11 sections.

- Top-level attributes
- Location information
- Collection of Fields
- Fields on the row axis
- Items on the row axis (specific values)
- Fields on the column axis
- Items on the column axis (specific values)
- Fields in the report filter area
- Fields in the values area
- Style information
- This is what the shell of that structure looks like:

```
<pivotTableDefinition>
  <location/>
  <pivotFields/>
  <rowFields/>
  <rowItems/>
  <colFields/>
  <colItems/>
  <pageFields/>
  <dataFields/>
  </dataFields>
  <conditionalFormats/>
  <pivotTableStyleInfo/>
</pivotTableDefinition>
```

Each collection will now be addressed section by section.

3.9.3.4.1 Attributes on pivotTableDefinition

```
<pivotTableDefinition xmlns:sh="…" name="PivotTable2" cacheId="5"
  applyNumberFormats="0" applyBorderFormats="0" applyFontFormats="0"
  applyPatternFormats="0" applyAlignmentFormats="0"
  applyWidthHeightFormats="1"
  dataCaption="Values" updatedVersion="3" minRefreshableVersion="3"
  showCalcMbrs="0" useAutoFormatting="1" colGrandTotals="0"
  itemPrintTitles="1"
  createdVersion="3" indent="0" outline="1" outlineData="1"
  multipleFieldFilters="0">
```

In the context of pivotTableDefinition:

- name indicates the name of the PivotTable.
- cacheId references by Id a particular pivotCache in the pivotCaches collection listed in workbook.xml.
- applyNumberFormats value of "1" means to apply legacy autoformat number format properties.
- applyBorderFormats value of "1" means to apply legacy autoformat border format properties.
- applyFontFormats value of "1" means to apply legacy autoformat Font format properties.
- applyPatternFormats value of "1" means to apply legacy autoformat pattern format properties.
- applyAlignmentFormats value of "1" means to apply legacy autoformat alignment format properties.
- applyWidthHeightFormats value of "1" means to apply legacy autoformat width and height format properties.
- dataCaption is the name of the values area header cell which can appear in the PivotTable when two or more fields are in the values area.
- updatedVersion is the Pivot version that last updated the PivotTable.
- minRefreshableVersion is the minimum Pivot version required to update this PivotTable's Pivot Cache.

- showCalcMbrs indicates whether calculated members should be shown in the PivotTable. Only applies to PivotTables based on OLAP sources.
- useAutoFormatting indicates whether autoformatting has been applied to the PivotTable.
- colGrandTotals indicates whether column grand totals are on for this PivotTable.
- rowGrandTotals defaults to "1" and therefore is not written.
- itemPrintTitles flag indicating whether PivotItem names should be repeated at the top of each printed page.
- createdVersion The Pivot version that created the cache.
- indent indentation increment for compact row axis, which means the Report Layout is set to Compact Form.
- outline flag indicating whether new fields should have their outline form flag set to "1".
- outlineData flag indicating whether the values field in the PivotTable should be displayed in outline form.
- multipleFieldFilters flag indicating whether each field of a pivot table can have multiple filters set on it.

3.9.3.4.2 Location Information

Location provides details on where the PivotTable is located in the sheet.

```
<location ref="B6:G13" firstHeaderRow="1" firstDataRow="4"
  firstDataCol="1" rowPageCount="3" colPageCount="1"/>
```

In the context of location:

- ref the location of the PivotTable area, not including the report filter area.
- firstHeaderRow the first row of the PivotTable header, relative to the top left cell in ref value.
- firstDataRow the first row of the PivotTable values area, relative to the top left cell in ref value.
- firstDataCol the first column of the PivotTable values area, relative to the top left cell in ref value.
- rowPageCount indicates how many rows the report filter area will occupy, as fields are added to it, before taking up another column (there can be multiple rows and columns of fields in the report filter area). By default there is a single column of report filter fields and the fields occupy as many rows as there are fields..
- colPageCount indicates how many columns the report filter region will occupy, as fields are added to it, before taking up another row (there can be multiple rows and columns of fields in the report filter region). By default, there is a single column of report filter fields and the fields occupy as many rows as there are fields.

3.9.3.4.3 PivotTable Fields

This collection expresses item order and field information for each field associated with the PivotTable, whether shown in the PivotTable report or not. (Note that items have been removed from the "Customer Name" and "City" fields (1st and 6th) to shorten the example.)

```
<pivotFields count="28">
  <pivotField showAll="0" includeNewItemsInFilter="1">
    <items count="8">
      <item x="66"/>
      <item x="133"/>
      <item x="74"/>
      <item x="27"/>
      <item x="118"/>
      <item x="63"/>
      <item x="141"/>
      <item t="default"/>
    </items>
  </pivotField>
  <pivotField showAll="0" includeNewItemsInFilter="1"/>
  <pivotField axis="axisPage" showAll="0" includeNewItemsInFilter="1">
    <items count="2">
      <item x="0"/>
      <item t="default"/>
    </items>
  </pivotField>
  <pivotField showAll="0" includeNewItemsInFilter="1"/>
  <pivotField axis="axisPage" showAll="0" includeNewItemsInFilter="1">
    <items count="6">
      <item x="3"/>
      <item x="1"/>
      <item x="2"/>
      <item x="4"/>
      <item x="0"/>
      <item t="default"/>
    </items>
  </pivotField>
  <pivotField axis="axisPage" showAll="0" includeNewItemsInFilter="1">
    <items count="8">
      <item x="0"/>
      <item x="1"/>
      <item x="2"/>
      <item x="3"/>
      <item x="4"/>
      <item x="5"/>
      <item x="6"/>
      <item t="default"/>
    </items>
  </pivotField>
```

```
      <pivotField showAll="0" includeNewItemsInFilter="1"/>
      <pivotField axis="axisRow" showAll="0" includeNewItemsInFilter="1">
        <items count="2">
          <item x="0"/>
          <item t="default"/>
        </items>
      </pivotField>
      <pivotField axis="axisRow" showAll="0" includeNewItemsInFilter="1">
        <items count="3">
          <item x="1"/>
          <item x="0"/>
          <item t="default"/>
        </items>
      </pivotField>
      <pivotField showAll="0" includeNewItemsInFilter="1"/>
      <pivotField showAll="0" includeNewItemsInFilter="1"/>
      <pivotField showAll="0" includeNewItemsInFilter="1"/>
      <pivotField showAll="0" includeNewItemsInFilter="1"/>
      <pivotField showAll="0" includeNewItemsInFilter="1"/>
      <pivotField axis="axisCol" showAll="0" includeNewItemsInFilter="1">
        <items count="2">
          <item x="0"/>
          <item t="default"/>
        </items>
      </pivotField>
      <pivotField axis="axisCol" showAll="0" includeNewItemsInFilter="1">
        <items count="2">
          <item x="0"/>
          <item t="default"/>
        </items>
      </pivotField>
      <pivotField axis="axisCol" showAll="0" includeNewItemsInFilter="1">
        <items count="4">
          <item x="1"/>
          <item x="2"/>
          <item x="0"/>
          <item t="default"/>
        </items>
      </pivotField>
      <pivotField showAll="0" includeNewItemsInFilter="1"/>
      <pivotField showAll="0" includeNewItemsInFilter="1"/>
      <pivotField showAll="0" includeNewItemsInFilter="1"/>
      <pivotField showAll="0" includeNewItemsInFilter="1"/>
```

```
      <pivotField showAll="0" includeNewItemsInFilter="1"/>
      <pivotField showAll="0" includeNewItemsInFilter="1"/>
      <pivotField showAll="0" includeNewItemsInFilter="1"/>
      <pivotField showAll="0" includeNewItemsInFilter="1"/>
      <pivotField dataField="1" showAll="0" includeNewItemsInFilter="1"/>
      <pivotField showAll="0" includeNewItemsInFilter="1"/>
      <pivotField showAll="0" includeNewItemsInFilter="1"/>
    </pivotFields>
```

In the context of pivotField:

- showAll flag indicating whether to show all items for this field.
- includeNewItemsInFilter Flag indicating if new items in the data source are included in the filter automatically after refresh when there was at least one hidden item for the field.
- axis indicates on which axis this field is shown on the PivotTable.
- dataField indicates that this field is in the values area of the PivotTable.

In the context of items, which is a listing of items (by index) in this field. The order in which the items are listed is the order they would appear on a particular axis (row or column, for example). In this example, the first field is "Customer Name" and the first item referenced here is <item x="66"/>,which references the value "Adam L Flores" in the pivotCacheDefinition. Therefore if one added "Customer Name" to the row axis, "Adam L Flores" would be the first row item listed.

In the context of item:

- t value of '`default`' indicates the subtotal or total item.

3.9.3.4.4 Row Axis Fields

This collection indicates which fields are on the row axis of the PivotTable.

```
    <rowFields count="2">
      <field x="7"/>
      <field x="8"/>
    </rowFields>
```

In the context of field within rowFields:

- x is a zero based index into the pivotFields collection.

For this example, this collection indicates that "Product Category" and "Product Subcategory" are on the row axis of the PivotTable, in that order.

3.9.3.4.5 Row Items

This collection is a listing of all the values on the row axis of the PivotTable. In the spreadsheet example, the item values are found in cells B10:B13. For example, "Bikes" is in B10, and corresponds to the first I element below.

```
<rowItems count="4">
  <i>
    <x/>
  </i>
  <i r="1">
    <x/>
  </i>
  <i r="1">
    <x v="1"/>
  </i>
  <i t="grand">
    <x/>
  </i>
</rowItems>
```

In the context of rowItems:

- i expresses all the values (for all fields) in one row of the row axis. There will be an I element for every row in the PivotTable.

In the context of i:

- r indicates how many fields/item values to "fill down" from the previous row item.

Note that the first item has no r explicitly written. Since a default of "0" is specified in the schema, for any item whose r is missing, a default value of "0" is implied.

In the context of x:

- v is a zero-based index referencing a pivotField item value. There will be as many x elements as there are item values in any particular row. Note that these x elements may not be explicitly written, but instead "inherited" from the previous row/i element, via the value of r. Note also that the pivotField items don't list values explicitly, but instead reference a shared item value in the pivotCacheDefinition part.

Note that the first instance of x has no attribute value v associated with it, so v's default value of "0" is implied.

Looking at the layout of the PivotTable in the spreadsheet for this example, "Bikes" is the first (and only) item value in the first row, in cell B10. In the XML defining the PivotTable row item values, the first I element corresponds to the first row. There is a single index element x. The first (and only) x element corresponds to

the first field on the row axis, namely "Product Category", and an index value of "0" indicates that the 0th item in the items collection for that pivotField definition is how to obtain the item value. Note that "Bikes" isn't explicitly listed as a value here, but instead the 0th item is an index to this field's shared items collection in the pivotCacheDefinition part.

For the second row, there are two item values, one item value (Bikes) from the first field in that row (Product Category) and one item value (Mountain Bikes) from the second field in that row (Product Subcategory). In the PivotTable, the first item value "Bikes" is hidden from view. In the XML for this example, the second I element expresses both item values for this row. The first item value "Bikes" is expressed implicitly, because the value of r on the second i element is '1', indicating that the first item value from the previous row will be reused again as the first item value for the current row. The second item value is expressed explicitly via the x element under the second i element. The index of '0' indicates that the 0th item in the pivotField element for that field is how to obtain the item value. Note again that the 0th item is itself an index into this field's shared items collection in the pivotCacheDefinition part.

The item values for the third row can be discovered in a similar way, so will not be discussed in detail here.

In the context of item:

- t value of 'default' indicates a grand total as the last row item value.

3.9.3.4.6 Column Axis Fields

This collection indicates which fields are on the column axis of the PivotTable.

```
<colFields count="3">
  <field x="14"/>
  <field x="15"/>
  <field x="16"/>
</colFields>
```

In the context of field:

- x is a zero based index into the pivotFields collection defined in this part.

For this example, the collection indicates that "Year", "Quarter" and "Month" are on the column axis of the PivotTable, in that order.

3.9.3.4.7 Column Items

This collection is a listing of all the values on the column axis of the PivotTable. In this example, the item values are found in cells C6:H8. For example, "2001" / "3" / "July" values are in C7:C9. Those are the first column item values and are referenced by the first <i> element below.

```
<colItems count="5">
  <i>
    <x/>
    <x/>
    <x/>
  </i>
  <i r="2">
    <x v="1"/>
  </i>
  <i r="2">
    <x v="2"/>
  </i>
  <i t="default" r="1">
    <x/>
  </i>
  <i t="default">
    <x/>
  </i>
</colItems>
```

In the context of colItems:

- i expresses all the values (for all fields) in one column of the column axis. There will be an i element for every column in the PivotTable column area.

In the context of i:

- r indicates how many fields/item values to "fill right" from the previous column.

Note that the first item has no r explicitly written so the default value of "0" is implied.

In the context of x:

- v is a zero-based index referencing a pivotField item value. There will be as many x elements as there are item values in any particular column. Note that these x elements sometimes are not explicitly written, but instead "inherited" from the previous column/i element, via the value of r. Note also that the pivotField items don't list values explicitly, but instead reference a shared item value in the pivotCacheDefinition part.

Note that the first instance of x has no attribute value v associated with it, so v's default value of "0" is implied.

The first i collection represents all item values for the first column in the column axis area of the PivotTable. The first x in the first i corresponds to the first field in the columns area of the PivotTable, namely "Year". The implied index value of '0' on this x indicates that the item value for this first item in the column is the 0th item

for this pivotField. The 0th item for this pivotField is itself an index to an item value into this field's shared items collection in the pivotCacheDefinition part, namely "2001".

The item values corresponding to the second and third x elements can be found in the same way, arriving at "3" for the second item value, and arriving at "July" for the third item value for this first column.

The second i collection expresses all three item values for the second column in the column axis area. The r value of '2' indicates that the first two item values from the previous column will be repeated here, which means that the first item value for this second column will be "2001" again and the second item value for this second column will be "3". The third item value is expressed by the only x element under this second i element, and without further explanation is understood to reference the item value "August".

3.9.3.4.8 Report Filter Area Fields

This collection describes which fields are found in the report filter area of the PivotTable.

```
<pageFields count="3">
  <pageField fld="2" hier="0"/>
  <pageField fld="4" hier="0"/>
  <pageField fld="5" hier="0"/>
</pageFields>
```

In the context of pageField:

- fld is a zero-based index indicating the field to be on the report filter area.
- hier is an index of the OLAP hierarchy to which this belongs.

3.9.3.4.9 Values Area Fields

This collection describes which fields are found in the values area of the PivotTable.

```
<dataFields count="1">
  <dataField name="Sum of Sales Amount" fld="25" baseField="0" baseItem="0"/>
</dataFields>
```

In the context of dataField:

- name is the name of the values field.
- fld is the index of the field being summarized.
- baseField is the index of the base field when showDataAs calculation is in use.
- baseItem is the index of the base item when showDataAs calculation is in use.

3.9.3.4.10 PivotTable Style Information

Styles information is discussed in the informative section on spreadsheetML styles. Therefore the XML is provided for completeness, but will not be discussed here.

```
<pivotTableStyleInfo name="PivotStyleDark8" showRowHeaders="1"
  showColHeaders="1" showRowStripes="0" showColStripes="0"
  showLastColumn="1"/>
</pivotTableDefinition>
```

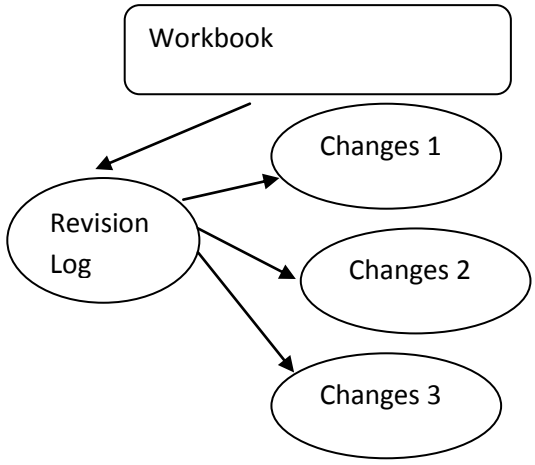
3.10 Shared Workbook Revisions

3.10.1 Overview

The Shared Workbooks architecture enables a spreadsheet application to record revisions made to a workbook (e.g., track changes), and is designed to enable multiple users to edit the same workbook at the same time. Therefore, the application needs to support the ability to read changes made by another user, and update its own state of the same workbook with those changes, even when those changes are made concurrently with other changes made by other users. Inevitably, there will be conflicts, and therefore merge conflict resolution should be supported by the runtime application.

This architecture supports the ability to track changes made by a single user as well.

3.10.2 How It Works



Relationship diagram

A Shared Workbook must have shared mode turned on. For unsaved workbooks, this will require a save, because revisions will be stored in the file.

Changes to the workbook are saved as Shared Workbook Revision Header parts within the document at each save or time interval specified.

A table summarizing the revision logs (revisionHeaders.xml) tracks when changes are made, who made them, and lists the relationship id to the specific Shared Workbook Revision Log part.

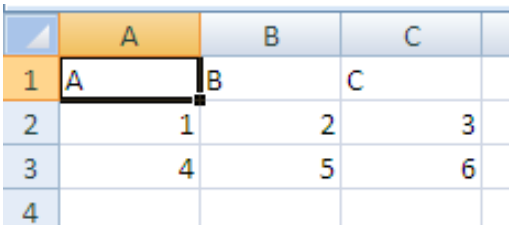
The application scans the summary table for new change logs and merges them into the workbook.

3.10.3 Example

Consider a series of edits made by different users.

3.10.3.1 First Edit

Starting with a blank workbook, the first user types "A, B, C" into A1:C1, and "1, 2, 3; 4, 5, 6" into A2:C3, like this:



Once the file is saved to disk after these edits, the summary table is updated and the revision log for this change is written.

3.10.3.1.1 Summary Revision Table

Contents of the Shared Workbook Revision Header part (revisionHeaders.xml).

Inside the summary table there is a revision header definition corresponding to the time of the edit:

```
<header guid="{902054C2-C7B5-48BA-BFB2-4D439D9758D6}"
  dateTime="2006-04-14T10:33:16" maxSheetId="4" userName="User 1"
  r:id="rId2" minRId="1" maxRId="11">
  <sheetIdMap count="3">
    <sheetId val="1"/>
    <sheetId val="2"/>
    <sheetId val="3"/>
  </sheetIdMap>
</header>
```

Notice that the user name, userName, and date and time stamp, dateTime, for the edit is stored along with an outline of the sheet structure. Use the r:id value of rId2 and then follow the relationship expressed in revisionHeaders.xml.rels. In this way, the corresponding Shared Workbook Revision Log can be located.

3.10.3.1.2 First Edit Revision Log

Inside the corresponding Shared Workbook Revision Log part is the following content:

```
<revisions xmlns="…" xmlns:r="…">
  <rcc rId="1" sId="1">
    <nc r="A1" t="inlineStr">
      <is>
        <t>A</t>
      </is>
    </nc>
  </rcc>
  <rcc rId="2" sId="1">
    <nc r="B1" t="inlineStr">
      <is>
        <t>B</t>
      </is>
    </nc>
  </rcc>
  <rcc rId="3" sId="1">
    <nc r="C1" t="inlineStr">
      <is>
        <t>C</t>
      </is>
    </nc>
  </rcc>
  <rrc rId="4" sId="1" eol="1" ref="A2:XFD2" action="insertRow"/>
  <rcc rId="5" sId="1">
    <nc r="A2">
      <v>1</v>
    </nc>
  </rcc>
  <rcc rId="6" sId="1">
    <nc r="B2">
      <v>2</v>
    </nc>
  </rcc>
  <rcc rId="7" sId="1">
    <nc r="C2">
      <v>3</v>
    </nc>
  </rcc>
  <rrc rId="8" sId="1" eol="1" ref="A3:XFD3" action="insertRow"/>
```

```
    <rcc rId="9" sId="1">
      <nc r="A3">
        <v>4</v>
      </nc>
    </rcc>
    <rcc rId="10" sId="1">
      <nc r="B3">
        <v>5</v>
      </nc>
    </rcc>
    <rcc rId="11" sId="1">
      <nc r="C3">
        <v>6</v>
      </nc>
    </rcc>
  </revisions>
```

rId is the revision Id, and indicates the order in which the particular revision should be applied.

sId indicates the sheet to which this revision applies.

rcc means "revision cell change"

nc means new cell, and is of type CT_Cell (see §3.2 for more information on the cell definition). Note that instead of using a shared string table, strings are expressed inline for these cells.

rrc means "revision row/column". Note that rrc can have an associated action, like insertRow (or deleteRow), which would cause a row to be inserted (or deleted) at that step in the series of revisions.

3.10.3.2 Second Edit

During the second edit, bold facing has been applied to A1:C1, and a formula has been applied to A4:C4 to sum the data in the table. For example, A4 contains =SUM(A2:A3).

	A	B	C
1	**A**	**B**	**C**
2	1	2	3
3	4	5	6
4	5	7	9

Once the file is saved to disk after these edits, the summary table is updated and the revision log for this change is written.

3.10.3.2.1 Summary Revision Table

Contents of the Shared Workbook Revision Header part (revisionHeaders.xml).

Inside the summary table there is a revision header definition corresponding to the time of the edit:

```
<header guid="{A3A5EE09-2092-433C-895D-77D5A15DC847}"
  dateTime="2006-04-14T10:34:10" maxSheetId="4" userName="User 2"
  r:id="rId3" minRId="12" maxRId="15">
  <sheetIdMap count="3">
    <sheetId val="1"/>
    <sheetId val="2"/>
    <sheetId val="3"/>
  </sheetIdMap>
</header>
```

This time the user name has been updated. Use the r:id value of `rId3` and then follow the relationship expressed in revisionHeaders.xml.rels. In this way, the corresponding Shared Workbook Revision Log can be located.

3.10.3.2.2 Second Edit Revision Log

Inside the corresponding Shared Workbook Revision Log part is the following content:

```
<revisions xmlns="…" xmlns:r="…">
  <rfmt sheetId="1" sqref="A1:C1" start="0" length="2147483647">
    <dxf>
      <font>
        <b/>
        </font>
    </dxf>
  </rfmt>
  <rrc rId="12" sId="1" eol="1" ref="A4:XFD4" action="insertRow"/>
  <rcc rId="13" sId="1">
    <nc r="A4">
      <f>SUM(A2:A3)</f>
    </nc>
  </rcc>
  <rcc rId="14" sId="1">
    <nc r="B4">
      <f>SUM(B2:B3)</f>
    </nc>
  </rcc>
```

```
    <rcc rId="15" sId="1">
      <nc r="C4">
        <f>SUM(C2:C3)</f>
      </nc>
    </rcc>
    <rcv guid="{34804977-BBD3-40C9-87A7-1779BEE2183C}" action="add"/>
  </revisions>
```

rfmt indicates a formatting revision

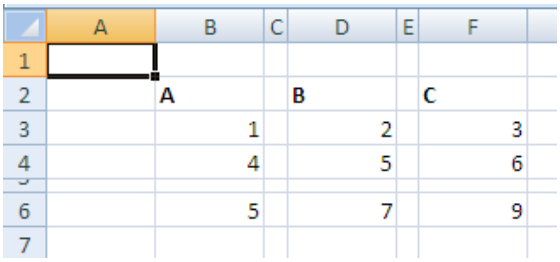
start and length indicate where to apply the formatting on the string

eol indicates that an insert is happening at the end of a list of data (end row)

rcv means "revision custom view", and indicates that a custom view is to be added.

3.10.3.3 Third Edit

During this editing session, column A has been inserted, and columns have been inserted between the data. Additionally, a row has been inserted between the data and the summary formula row, and at the top of the worksheet.



	A	B	C	D	E	F
1						
2		A		B		C
3		1		2		3
4		4		5		6
6		5		7		9
7						

Once the file is saved to disk after these edits, the summary table is updated and the revision log for this change is written.

3.10.3.3.1 Summary Revision Table

Contents of the Shared Workbook Revision Header part (revisionHeaders.xml).

```
  <header guid="{894981D2-DACF-4C1B-951C-EB199EA01DBF}"
    dateTime="2006-04-14T10:36:10" maxSheetId="4" userName="User 2"
    r:id="rId4" minRId="16" maxRId="20">
    <sheetIdMap count="3">
      <sheetId val="1"/>
      <sheetId val="2"/>
      <sheetId val="3"/>
    </sheetIdMap>
  </header>
```

Use the r:id value of rId4 and then follow the relationship expressed in revisionHeaders.xml.rels. In this way, the corresponding Shared Workbook Revision Log part can be located.

3.10.3.3.2 Third Edit Revision Log

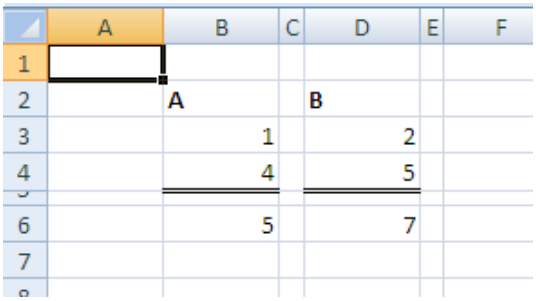
Inside the corresponding Shared Workbook Revision Log part is the following content:

```
<revisions xmlns="…" xmlns:r="…">
  <rrc rId="16" sId="1" ref="A1:XFD1" action="insertRow"/>
  <rrc rId="17" sId="1" ref="A1:A1048576" action="insertCol"/>
  <rrc rId="18" sId="1" ref="C1:C1048576" action="insertCol"/>
  <rrc rId="19" sId="1" ref="E1:E1048576" action="insertCol"/>
  <rrc rId="20" sId="1" ref="A5:XFD5" action="insertRow"/>
  <rcv guid="{34804977-BBD3-40C9-87A7-1779BEE2183C}" action="delete"/>
  <rcv guid="{34804977-BBD3-40C9-87A7-1779BEE2183C}" action="add"/>
</revisions>
```

rrc indicates a "revision to row/column". There are several row inserts and column inserts expressed here.

3.10.3.4 Fourth Edit

During this edit, a double-underscore cell border was applied to B4 and D4. Also, column F (the data titled "C") was deleted.



Once the file is saved to disk after these edits, the summary table is updated and the revision log for this change is written.

3.10.3.4.1 Summary Revision Table

Contents of the Shared Workbook Revision Header part (revisionHeaders.xml).

```
<header guid="{A478A962-DEB9-43AA-BB25-2C54AFA155F1}"
  dateTime="2006-04-14T10:37:14" maxSheetId="4" userName="User 2"
  r:id="rId5" minRId="21">
```

```
    <sheetIdMap count="3">
      <sheetId val="1"/>
      <sheetId val="2"/>
      <sheetId val="3"/>
    </sheetIdMap>
  </header>
```

Use the r:id value of rId5 and then follow the relationship expressed in revisionHeaders.xml.rels. In this way, the corresponding Shared Workbook Revision Log part can be located.

3.10.3.4.2 Fourth Edit Revision Log

Inside the corresponding Shared Workbook Revision Log part is the following content:

```
  <revisions xmlns="…" xmlns:r="…">
    <rrc rId="21" sId="1" ref="F1:F1048576" action="deleteCol">
      <rfmt sheetId="1" xfDxf="1" sqref="F1:F1048576" start="0" length="0"/>
      <rcc rId="0" sId="1" dxf="1">
        <nc r="F2" t="inlineStr">
          <is>
            <t>C</t>
          </is>
        </nc>
        <ndxf>
          <font>
            <b/>
            <sz val="11"/>
            <color theme="1"/>
            <name val="Calibri"/>
            <scheme val="minor"/>
          </font>
        </ndxf>
      </rcc>
      <rcc rId="0" sId="1">
        <nc r="F3">
          <v>3</v>
        </nc>
      </rcc>
      <rcc rId="0" sId="1">
        <nc r="F4">
          <v>6</v>
        </nc>
      </rcc>
      <rcc rId="0" sId="1">
        <nc r="F6">
```

```
            <f>SUM(F3:F4)</f>
          </nc>
        </rcc>
      </rrc>
      <rfmt sheetId="1" sqref="B4" start="0" length="0">
        <dxf>
          <border>
            <left/>
            <right/>
            <top/>
            <bottom style="double">
              <color auto="1"/>
            </bottom>
          </border>
        </dxf>
      </rfmt>
      <rfmt sheetId="1" sqref="D4" start="0" length="0">
        <dxf>
          <border>
            <left/>
            <right/>
            <top/>
            <bottom style="double">
              <color auto="1"/>
            </bottom>
          </border>
        </dxf>
      </rfmt>
    </revisions>
```

The first rrc element, with action="deleteCol" expresses that column F was deleted. Additionally, child collections of rrc contain all the column, formatting, and cell information (values and formulas) that was deleted as part of deleting column F.

xfDxf `true` means a whole row/column of formatting was affected.

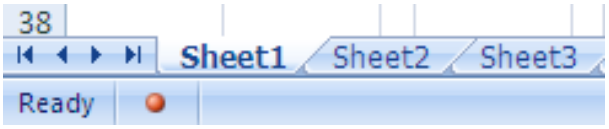
dxf `true` means cell change includes format change

The rfmt collections at the bottom of this XML indicate that borders were applied to B4 and D4.

3.10.3.5 Fifth Edit

Rename "Sheet1" to "Published Numbers".

Before:



After:

Once the file is saved to disk after these edits, the summary table is updated and the revision log for this change is written.

3.10.3.5.1 Summary Revision Table

Contents of the Shared Workbook Revision Header part (revisionHeaders.xml).

```
<header guid="{B0CB8BC9-63A4-4830-8821-E03C053BD326}"
  dateTime="2006-04-14T10:40:24" maxSheetId="4" userName="User 2"
  r:id="rId6" minRId="22">
  <sheetIdMap count="3">
    <sheetId val="1"/>
    <sheetId val="2"/>
    <sheetId val="3"/>
  </sheetIdMap>
</header>
```

Use the r:id value of `rId6` and then follow the relationship expressed in revisionHeaders.xml.rels. In this way the corresponding Shared Workbook Revision Log part can be located.

3.10.3.5.2 Fifth Edit Revision Log

Inside the corresponding Shared Workbook Revision Log part is the following content:

```
<revisions xmlns="…" xmlns:r="…">
  <rsnm rId="22" sheetId="1" oldName="[SharedWorkbook.xlsx]Sheet1"
  newName="[SharedWorkbook.xlsx]Published Numbers"/>
</revisions>
```

rsnm means "revision: sheet name".

oldName indicates the name of the sheet before renaming it.

newName indicates the name of the sheet after renaming it.

3.11 Query Tables

3.11.1 Overview

A *QueryTable object* is a range that is bound to an external data source. It is a cohesive range of cells in a sheet that share a common collection of properties and behaviors, separate from the connection itself. A QueryTable object can be associated with a cell range or a table definition.

3.11.2 Web Query Example

This example illustrates a range, B2:D6, QueryTable rendering, which is data-bound to a table found on http://www.msn.com, specifically the financial information table usually found on that page.

	A	B	C	D	E	F
1						
2		This table charts the key U.S. financial indices by their last				
3		Index	Last	Change		
4		Dow	11,642.65	2.88		
5		NASDAQ	2,320.74	-17.51		
6		S&P	1,322.85	-2.29		

3.11.2.1 QueryTable XML

```
<queryTable xmlns="…" name="www.msn" preserveFormatting="0"
  connectionId="1"
  autoFormatId="16" applyNumberFormats="0" applyBorderFormats="0"
  applyFontFormats="1" applyPatternFormats="1" applyAlignmentFormats="0"
  applyWidthHeightFormats="0"/>
```

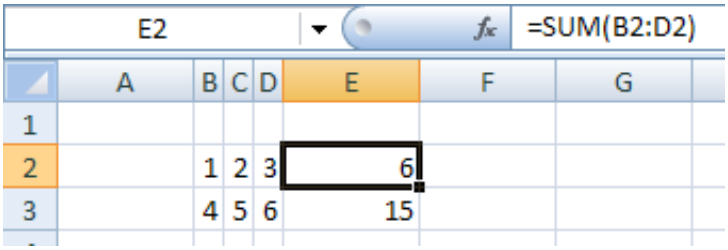
In the context of queryTable:

- name is the name of the QueryTable.
- preserveFormatting indicates whether to retain user-applied formatting after refresh or re-apply source data formatting.
- connectionId is the workbook connection's id.
- autoFormatId identifies (by implied index) the auto-format applied to the QueryTable.

All remaining attributes beginning with apply… indicate whether to apply this particular aspect of the auto-format definition.

3.11.3 Text Import Example

This example illustrates a range (B2:D3) QueryTable rendering which is data-bound to a text file. Notice that formulas are entered in E2:E3, the column directly to the right of the QueryTable range.



3.11.3.1 QueryTable XML

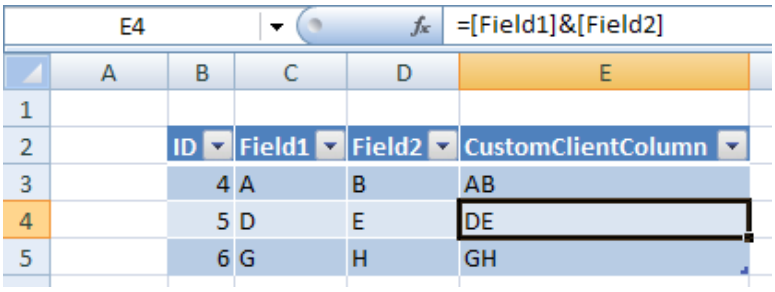
```
<queryTable xmlns="…" name="Text" refreshOnLoad="1" fillFormulas="1"
  removeDataOnSave="1" connectionId="3" autoFormatId="16"
  applyNumberFormats="0" applyBorderFormats="0" applyFontFormats="1"
  applyPatternFormats="1" applyAlignmentFormats="0"
  applyWidthHeightFormats="0"/>
```

In the context of queryTable:

- refreshOnLoad value of 1 indicates that this QueryTable should be refreshed when the workbook is opened.
- fillFormulas indicates that this QueryTable has immediately adjacent columns (which are not part of the QueryTable range) containing formulas that need to be filled down as the QueryTable grows and shrinks in size after refresh.
- removeDataOnSave indicates that the data in the worksheet resulting from the QueryTable refresh should be removed from the worksheet when saved and closed.

3.11.4 Access Table Example

This example demonstrates a QueryTable that is applied to a Table object. This data came from connecting to an Access database table with four fields: "ID", "Field1", "Field2", and "Field3". "Field3" has been deleted from the QueryTable in the worksheet below. Notice that a calculated column has been added to the Table, in the column titled "CustomClientColumn", which concatenates the values from "Field1" and "Field2".



3.11.4.1 QueryTable XML

```
<queryTable xmlns="…" name="Database1.accdb" connectionId="2"
  autoFormatId="16" applyNumberFormats="0" applyBorderFormats="0"
  applyFontFormats="0" applyPatternFormats="0" applyAlignmentFormats="0"
  applyWidthHeightFormats="0">
```

```
    <queryTableRefresh nextId="6" unboundColumnsRight="1">
      <queryTableFields count="4">
        <queryTableField id="1" name="ID" tableColumnId="1"/>
        <queryTableField id="2" name="Field1" tableColumnId="2"/>
        <queryTableField id="3" name="Field2" tableColumnId="3"/>
        <queryTableField id="5" dataBound="0" tableColumnId="4"/>
      </queryTableFields>
      <queryTableDeletedFields count="1">
        <deletedField name="Field3"/>
      </queryTableDeletedFields>
    </queryTableRefresh>
  </queryTable>
```

In the context of queryTableRefresh:

- nextId is the next available Id that can be assigned to a field. This is an optimization done for load/save, to avoid recalculating the value.
- unboundColumnsRight are the number of columns on the right side of the QueryTable that aren't data bound (don't come from the external data)

In the context of queryTableFields:

- Each of the queryTableField elements expresses information about one of the columns in the Table that is part of the QueryTable. For example, the right-most column's dataBound is set to `0`, indicating that this column is not bound to external data.

In the context of queryTable:

- queryTableDeletedFields collection expresses which fields returned by the connection have been deleted from the QueryTable. This is tracked so that the connection information does not have to be updated with which columns are no longer required.

3.12 External Connection

3.12.1 Overview

Many spreadsheet users want to be able to access data from external sources: databases, text files, web pages, XML web services, OLAP cubes. Typically, a spreadsheet application will provide abilities for the user to locate, browse, connect to, and query external data sources. Once the data source has been located, connected to, and queried, the resulting data must be rendered in the spreadsheet application, and made available for further analysis.

Data sources such as databases are made available for browsing and consumption via data-provider technologies. Typically, the data provider provides a standard interface for accessing the data, and removes the complexity introduced due to each database application's providing non-standard data access APIs. In this

way, OLEDB providers, for example, can be written for myriad database implementations, and a consumer can always use a single interface (defined by OLEDB) to access these disparate data sources.

A live connection to a data source is established by the application at runtime, and can only exist as a live connection while the application is running. There are two types of information about a particular connection:

- The information used to establish the connection.
- The information and properties about how the connection should be used and how the connection should behave in conjunction with the application.

Information about a connection can be supplied by the user as the connection is being established—for example, providing a password, picking a table, applying a filter, or setting behavioral properties such as whether to refresh the data when the workbook is opened and whether to store refreshed data in the worksheet when the workbook is saved.

Information about a connection can also be persisted in a connection file separately from the workbook file. In this way a directory or file share containing a variety of these connection files can be considered a library of data connections, for example.

Any time a connection is established, whether by using information from a connection file or by gathering the connection information directly from the user, a copy of the connection information is stored in the workbook.

Data providers and connection types discussed below are:

- ODBC
- OLEDB
- ADO
- DAO
- Text Import
- Web

The corresponding features in SpreadsheetML that render and analyze the data are:

- Query Table
- Table
- XML Map
- Pivot Table
- The CUBE* Functions

3.12.2 OLAP Connection

Below is a PivotTable that is rendering data from an OLAP source:

	A	B	C
1			
2		Row Labels	Amount
3		Current Year's Actuals	398755.69
4		Adjustment for Budget input	565238.13
5		Current Year's Budget	565238.13
6		Forecast	565238.13
7		Grand Total	398755.69
8			

3.12.3 Pivot XML fragment

In this example, the PivotTable data cache records /cacheSource@connectionId="2", which associates this PivotTable to the connection whose id="2" in the workbook connections part.

```
<pivotCacheDefinition … saveData="0" refreshedBy="Chad Rothschiller"
  refreshedDate="2006-04-13T16:02:14" backgroundQuery="1"
    createdVersion="3"
  refreshedVersion="3" minRefreshableVersion="3" recordCount="0">
  <cacheSource type="external" connectionId="2"/>
  <cacheFields count="2">
   <cacheField name="[Category].[Category Description]"
     caption="Category
     Description" numFmtId="0" hierarchy="1" level="1">
     <sharedItems count="4">
       <s v="[Category].[All Category].[Current Year's Actuals]"
         c="Current Year's Actuals"/>
```

3.12.4 Connection XML

Looking at the XML in the workbook connections part that describes the connection whose id="2":

```
<connection id="2" odcFile="
  c:\…\externalData.odc" keepAlive="1" name="externalData"
  description="FoodMart 2000 - Budget Planing" type="5"
  refreshedVersion="3" background="1">
  <dbPr connection="Provider=MSOLAP.2;Integrated Security=SSPI;Persist
    Security Info=True;Data Source=xlextdat8;Initial
      Catalog=AdamTest;Client
    Cache Size=25;Auto Synch Period=10000;MDX Compatibility=1"
      command="Budget" commandType="1"/>
  <olapPr sendLocale="1" rowDrillCount="1000"/>
</connection>
```

Attributes on the connection element express properties about the connection.

- odcFile indicates the location of the data connection file which was used to create this connection. Data connection files can be created on the local machine, on any network share, or any web location

whenever the connection is created so that the connection information can be reused if desired. The connection information is copied from the connection file into the spreadsheet. When a connection cannot be established, the spreadsheet application can check the external connection file to see if a newer definition of the connection is available.

- name indicates the friendly name of the connection. This name must be unique within a workbook.
- keepAlive indicates that the application should hold the connection open once established, instead of closing the connection after retrieving the data.
- type value of 5 indicates that this connection type is OLEDB.
- refreshedVersion indicates the version of the application which last refreshed this connection.
- background value of 1 indicates that background refresh (asynchronous refresh) is enabled. Note that this is not a guarantee that the connection will be refreshed asynchronously. Certain objects may require a connection to be refreshed either synchronously or asynchronously regardless of this setting.

Attributes on the dbPr element express additional properties on the connection.

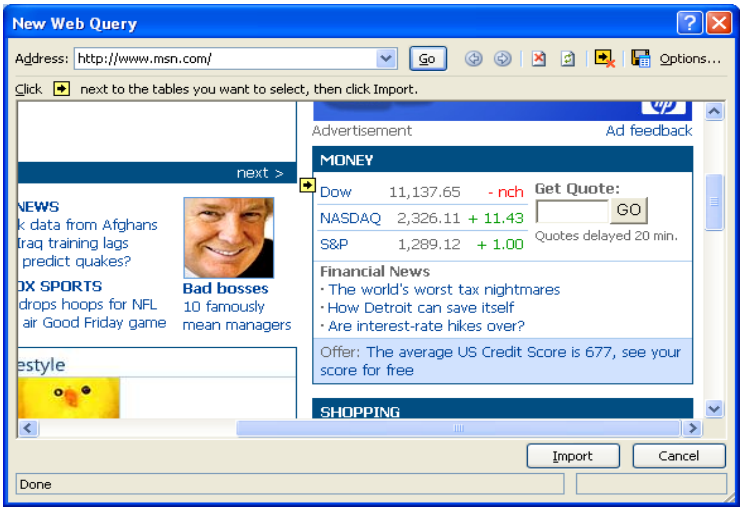
- connection expresses the connection string that is needed to establish a connection to the external data source.
- command can indicate a table name, a cube name, or an SQL expression requesting data.
- commandType indicates what kind of information is found in command: 1 means the value of command is the name of an OLAP cube.

Attributes on the olapPr element express properties that apply to connections to OLAP data sources.

- sendLocale value of 1 indicates that the client application should send its user interface language locale to the OLAP data provider in order to receive back from the server localized OLAP cube member string values.
- rowDrillCount is number of rows to return on a drill through request.

3.12.5 Web Query

A possible user interface for picking a web source is:



A possible rendering in the spreadsheet grid might be:

B	C	D
Web Query to MSN Money		
Dow	11,137.65	7.68
NASDAQ	2,326.11	11.43
S&P	1,289.12	1

3.12.6 QueryTable XML

The XML expressing the definition of the QueryTable indicates that it is using the connection whose Id value is 1 (connectionId):

```
<queryTable … name="msn" connectionId="1" autoFormatId="16"
  applyNumberFormats="0" applyBorderFormats="0" applyFontFormats="1"
  applyPatternFormats="1" applyAlignmentFormats="0"
applyWidthHeightFormats="0"/>
```

3.12.7 Connection XML

The workbook connection whose Id is 1 is expressed below.

```
<connection id="1" name="Connection" type="4" refreshedVersion="3"
  background="1" saveData="1">
  <webPr sourceData="1" parsePre="1" consecutive="1" xl2000="1"
    url="http://www.msn.com" htmlTables="1">
    <tables count="1">
      <x v="2"/>
    </tables>
  </webPr>
</connection>
```

Attributes and elements that have been previously discussed are not discussed here.

- type value of 4 indicates the connection is a web query connection.
- saveData value of 1 indicates that refreshed data will be kept in the sheet when saving the workbook. 0 indicates to remove the data from the workbook when saving the workbook.
- sourceData value of 1indicates to import and parse the XML data rather than consume the web page's HTML definition.
- parsePre value of 1 indicates that text in <PRE> tags is interpreted as tables.
- consecutive value of 1 means consecutive delimiters are treated as one delimiter
- url is the address indicating where to retrieve data for this query.
- htmlTables `true` means only import html tables

tables indicates which HTML table to import from the web page.

- v value of 2 indicates that the second table is the one to import.

3.12.8 Unused Connection

A connection can be expressed in a workbook, but not currently used. It remains until deleted by the user explicitly. This simply means that there is no object or feature in the workbook; that is referencing the connection.

3.12.9 ODBC

A database table imported to the grid, where the data provider is ODBC:

	A	B	C	D
1	ID	Field1	Field2	Field3
2	3	Foo	Bar	Goo
3	4	The	Quick	Brown
4	5	Fox	Jumped	Over
5	6	The	Lazy	Dog
6				

When a table object is used to render external data, it is associated with a QueryTable object to store the properties used when a range is associated with external data. Therefore, the Table object references the QueryTable name, which in turns references connectionId to identify the connection in the workbook connections part.

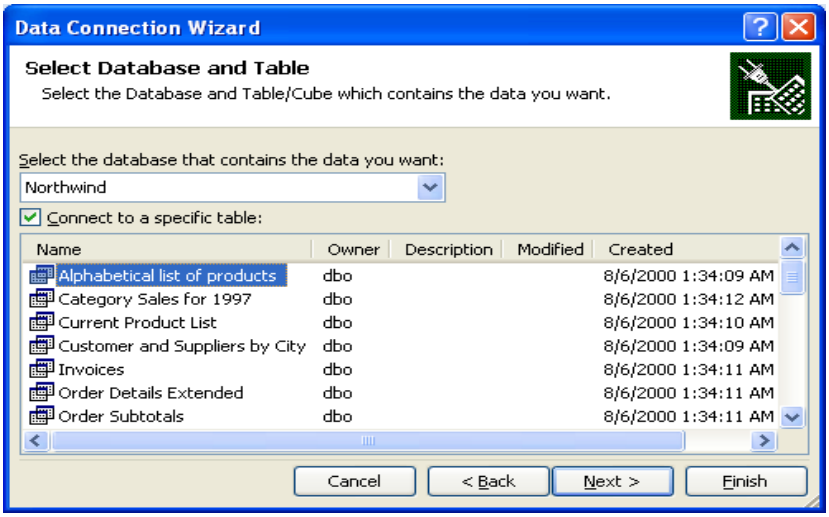
3.12.10 Connection XML

```
<connection id="4" name="Query from MS Access Database" type="1"
  refreshedVersion="3" background="1" saveData="1">
  <dbPr connection="DSN=MS Access
     Database;DBQ=E:\…\Database1.accdb;DefaultDir=E:\Documents and
     Settings\chadroth\Desktop;DriverId=25;FIL=MS
     Access;MaxBufferSize=2048;PageTimeout=5;"
       command="SELECT Table1.ID,
       Table1.Field1, Table1.Field2,
       Table1.Field3_x000d__x000a_FROM
       `E:\…\Database1.accdb`.Table1 Table1"/>
</connection>
```

- type value of 1 indicates ODBC connection type.
- command contents are an SQL select statement.

3.12.11 SQL

An implementation might use a data connection wizard to connect to a SQL table; for example:



The resulting data is rendered in the grid:

	A	B	C	D
1				
2		SQL Connection		
3		ShippedDate	OrderID	Subtotal
4		7/16/1996 0:00	10248	440
5		7/10/1996 0:00	10249	1863.4
6		7/12/1996 0:00	10250	1552.6
7		7/15/1996 0:00	10251	654.06
8		7/11/1996 0:00	10252	3597.9
9		7/16/1996 0:00	10253	1444.8
10		7/23/1996 0:00	10254	556.62
11		7/15/1996 0:00	10255	2490.5
12		7/17/1996 0:00	10256	517.8
13		7/22/1996 0:00	10257	1119.9
14		7/23/1996 0:00	10258	1614.88
15		7/25/1996 0:00	10259	100.8
16		7/29/1996 0:00	10260	1504.65
17		7/30/1996 0:00	10261	448
18		7/25/1996 0:00	10262	584

In this example, a table object is used to render external data, it is associated with a QueryTable object to store the properties used when a range is associated with external data. Therefore, the Table object references the QueryTable name, which in turns references `connectionId` to identify the connection in the workbook connections part.

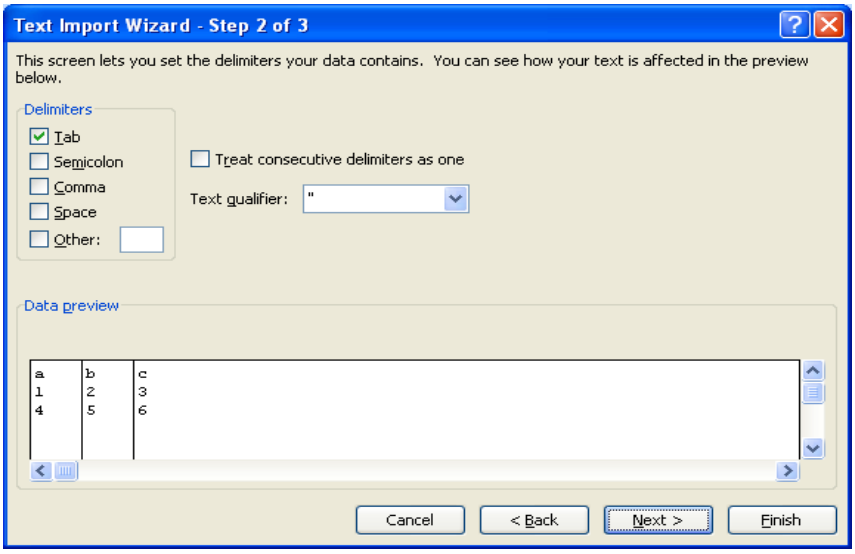
3.12.12 Connection XML

```
<connection id="6" odcFile="c:\…\xlextdat8 Northwind Summary of Sales by
  Year.odc" keepAlive="1"
  name="xlextdat8 Northwind Summary of Sales by Year"
  type="5" refreshedVersion="3" background="1" saveData="1">
  <dbPr connection="Provider=SQLOLEDB.1;Integrated Security=SSPI;
    Persist Security Info=True;Initial Catalog=Northwind;Data
    Source=xlextdat8;Use Procedure for Prepare=1;Auto
    Translate=True;Packet Size=4096;Workstation ID=CHADROTHO12;Use
    Encryption for Data=False;Tag with column collation when
    possible=False" command="&amp;quot;Northwind&amp;quot;.&amp;quot;dbo&amp;quot;
   .&amp;quot;Summary of Sales by Year&amp;quot;" commandType="3"/>
</connection>
```

- type value of 5 indicates that this connection is using an OLEDB data provider.
- commandType value of 3 specifies that a table name is in command
- command specifies a table name.

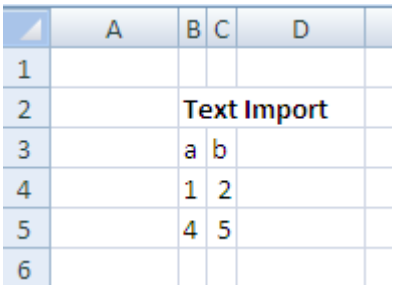
3.12.13 Text Import

Text Import settings:



Note that there are additional settings not pictured here.

The resulting data in the grid:



	A	B	C	D
1				
2		Text Import		
3		a	b	
4		1	2	
5		4	5	
6				

The range is associated with a QueryTable object. This query table definition references the connectionId used to retrieve the data.

3.12.14 Connection XML

```
<connection id="5" name="Text" type="6" refreshedVersion="3"
  background="1" saveData="1">…
  <textPr codePage="437" sourceFile="E:\ …\Text.txt">
    <textFields count="3">
      <textField type="text"/>
      <textField position="5"/>
      <textField type="skip" position="10"/>
    </textFields>
  </textPr>
</connection>
```

connection defines the connection

- type value of 6 indicates that this is a text import type of connection.

textPr expresses properties which are specific to text import connections.

- codePage value of 437 indicates that the text file is using the IBM PC (OEM) code page 437 character set.
- sourceFile indicates where the file is located.

textFields expresses information about the particular fields in the text file.

- delimited value of 1 (default) indicates that the text is delimited (variable length). Since this example uses the default value, it is not saved as part of the connection information.
- type indicates the data type (user-specified) of the particular field.
- position indicates the starting position of the field for fixed-width fields.
- thousands specifies the thousands separator character (not in this example, but of enough interest to mention).
- tab, space, comma attributes with values of 1 would flag these characters as delimiters (not in this example, but of enough interest to mention).

3.13 External Links

3.13.1 Overview

An *external link* is used to link a workbook to other workbook or to external data. The most frequent occurrence for linking a workbook to other workbooks has to do with formulas. In this case, a formula references a range or name defined in another workbook. Hyperlinks on cells and other spreadsheet objects are also considered an external link. OLE links are yet another technology used to link the workbook to another object. Finally, Dynamic Data Exchange (DDE) servers can be used to access external data. DDE servers are accessed through formulas in the workbook.

The goal of the way in which external links are saved is to always write the target source in a relationship file, so that external resources are easily discoverable in lightweight relationship XML rather than deep in the application's XML.

3.13.2 Formula Example

Consider cells B2 and C2 in the following worksheet, Sheet1:

	A	B	C
1		**Link to an external workbook range:**	**Link to an external workbook name:**
2		=SUM('C:\[Source.xlsx]Sheet1'!$A$1:$A$3)	='C:\Source2.xlsx'!NameInExternalWorkbook
3			
4		W3C Hyperlink	
5			

Here, the formulas themselves are displayed in the cells.

	A	B	C	D
1		**Link to an external workbook range:**	**Link to an external workbook name:**	
2		6	2	
3				
4		W3C Hyperlink		

Here, the results of the formulas are displayed in the cells.

The formula is expressed in Sheet1's XML, as shown in the following subclause.

3.13.3 Sheet XML

The corresponding content from Sheet1.xml is:

```
<worksheet …>
  <dimension ref="B1:C4"/>
  <sheetViews>
    <sheetView tabSelected="1" workbookViewId="0">
      <selection activeCell="B2" sqref="B2"/>
    </sheetView>
  </sheetViews>
  <sheetFormatPr defaultRowHeight="15"/>
  <cols>
    <col min="1" max="1" width="1.7109375" customWidth="1"/>
  </cols>
  <sheetData>
    <row r="1" spans="2:3" customFormat="1" ht="9" customHeight="1"/>
    <row r="2" spans="2:3" customFormat="1">
      <c r="B2">
        <f>SUM([1]Sheet1!$A$1:$A$3)</f>
        <v>6</v>
      </c>
      <c r="C2">
        <f>[2]!NameInExternalWorkbook</f>
        <v>2</v>
      </c>
    </row>
    <row r="4" spans="2:3" customFormat="1">
      <c r="B4" s="1" t="s">
        <v>0</v>
      </c>
    </row>
  </sheetData>
  <hyperlinks>
    <hyperlink ref="B4" r:id="rId1"/>
  </hyperlinks>
  <printOptions/>
  <pageMargins left="0.7" right="0.7" top="0.75" bottom="0.75"
    header="0.3" footer="0.3"/>
  <headerFooter/>
</worksheet>
```

3.13.3.1 Cell B2

The formula expressed in cell B2 (cell B2 is the c element whose r="B2") is this:

```
SUM([1]Sheet1!$A$1:$A$3)
```

The external reference to another workbook in this case is tokenized to `[1]`. The value inside the brackets is a 1-based index to the externalReferences collection in the workbook part.

3.13.3.2 Cell C2

The formula expressed in cell C2 (cell C2 is the `c` element whose `r` is `C2`) is this:

```
[2]!NameInExternalWorkbook
```

The external reference to another workbook in this case is tokenized to `[2]`. The value inside the brackets is a 1-based index to the externalReferences collection in the workbook part.

3.13.3.3 Workbook XML

The corresponding content from workbook.xml is

```
<workbook …>
  <fileVersion lastEdited="4" lowestEdited="4" rupBuild="4012"/>
  <workbookPr defaultThemeVersion="123820"/>
  <bookViews>
    <workbookView xWindow="360" yWindow="270" windowWidth="18735"
      windowHeight="11445"/>
  </bookViews>
  <sheets>
    <sheet name="Sheet1" sheetId="1" r:id="rId1"/>
    <sheet name="Sheet2" sheetId="2" r:id="rId2"/>
    <sheet name="Sheet3" sheetId="3" r:id="rId3"/>
  </sheets>
  <externalReferences>
    <externalReference r:id="rId4"/>
    <externalReference r:id="rId5"/>
  </externalReferences>
  <calcPr calcId="122211"/>
  <webPublishing codePage="1252"/>
</workbook>
```

The workbook part's externalReferences collection indicates that there are two external workbook references in this workbook. The first supporting external workbook data cache, also stored in this workbook, can be found by following the relationship from the workbook whose Id value is `rId4`. The second supporting external workbook data cache, also stored in this workbook, can be found by following the relationship from the workbook whose Id value is `rId5`.

3.13.4 Workbook Relationships

The corresponding content from workbook.xml.rels is:

```
<Relationships xmlns="http://…/package/2006/relationships">
  <Relationship Id="rId8" Type="http://…/sharedStrings"
Target="sharedStrings.xml"/>
  <Relationship Id="rId3" Type="http://…/worksheet"
Target="worksheets/sheet3.xml"/>
  <Relationship Id="rId7" Type="http://…/styles" Target="styles.xml"/>
  <Relationship Id="rId2" Type="http://…/worksheet"
Target="worksheets/sheet2.xml"/>
  <Relationship Id="rId1" Type="http://…/worksheet"
Target="worksheets/sheet1.xml"/>
  <Relationship Id="rId6" Type="http://…/theme" Target="theme/theme1.xml"/>
  <Relationship Id="rId5" Type="http://…/externalLink"
    Target="externalLinks/externalLink2.xml"/>
  <Relationship Id="rId4" Type="http://…/externalLink"
    Target="externalLinks/externalLink1.xml"/>
  <Relationship Id="rId9" Type="http://…/calcChain" Target="calcChain.xml"/>
</Relationships>
```

These relationship expressions indicate that cell B2 is supported by the external workbook data cache located at `externalLinks/externalLink1.xml` in the package. These relationship expressions also indicate that cell C2 is supported by the external workbook data cache located at `externalLinks/externalLink2.xml` in the package.

3.13.5 Supporting Workbook Cache (Cell C2)

The corresponding content from externalLink2.xml is:

```
<externalLink …>
  <externalBook xmlns:r="http://schemas.openxmlformats.org
    /officeDocument/2006/relationships" r:id="rId1">
    <sheetNames>
      <sheetName val="Sheet1"/>
      <sheetName val="Sheet2"/>
      <sheetName val="Sheet3"/>
    </sheetNames>
    <definedNames>
      <definedName name="NameInExternalWorkbook"
        refersTo="='Sheet1'!$B$1"/>
    </definedNames>
```

```
      <sheetDataSet>
        <sheetData sheetId="0">
          <row r="1">
            <cell r="B1">
              <v>2</v>
            </cell>
          </row>
        </sheetData>
        <sheetData sheetId="1"/>
        <sheetData sheetId="2"/>
      </sheetDataSet>
    </externalBook>
  </externalLink>
```

Supporting workbook data caches store the top-level structure of the workbook (sheet names, defined names, cell table). Only the cells referenced are cached. This supporting workbook data cache indicates that the workbook being referenced by C2 has three sheets, whose names are "Sheet1", "Sheet2", and "Sheet3", and has a defined name of "NameInExternalWorkbook". Additionally, the cell table shows that cell B1 in this workbook is the cell being referenced. A copy of the cell table is stored locally, inside the workbook containing the external link.

The r:id="rId1" on the top level externalLink element indicates the Id of the relationship from the `externalLink2.xml` part, which indicates the location of the actual external workbook.

3.13.6 External Link (Cell C2)

The corresponding content from externalLink2.xml.rels is

```
  <Relationships …>
    <Relationship Id="rId1" Type="http://…/externalLinkPath"
      Target="file:///C:\Source2.xlsx" TargetMode="External"/>
  </Relationships>
```

This relationship indicates that the supporting workbook that C2 references resides on the local drive, at `c:\source2.xlsx`.

3.13.7 Supporting Workbook Cache (Cell B2)

The corresponding content from externalLink1.xml is:

```
<externalLink …>
  <externalBook xmlns:r="http://…/relationships" r:id="rId1">
    <sheetNames>
      <sheetName val="Sheet1"/>
      <sheetName val="Sheet2"/>
      <sheetName val="Sheet3"/>
    </sheetNames>
    <sheetDataSet>
      <sheetData sheetId="0">
        <row r="1">
          <cell r="A1">
            <v>1</v>
          </cell>
        </row>
        <row r="2">
          <cell r="A2">
            <v>2</v>
          </cell>
        </row>
        <row r="3">
          <cell r="A3">
            <v>3</v>
          </cell>
        </row>
      </sheetData>
      <sheetData sheetId="1"/>
      <sheetData sheetId="2"/>
    </sheetDataSet>
  </externalBook>
</externalLink>
```

This supporting workbook data cache indicates that the workbook being referenced by B2 has three sheets, whose names are "Sheet1", "Sheet2", and "Sheet3". Additionally, the cell table shows that cells A1, A2, and A3, whose values are 1, 2, and 3, respectively, in this workbook are being referenced. A copy of the cell table is stored locally, inside the workbook containing the external link.

The r:id="rId1" on the top level externalLink element indicates the Id of the relationship from the `externalLink1.xml` part, which indicates the location of the actual external workbook.

3.13.8 External Link (Cell B2)

The corresponding content from externalLink1.xml.rels is

```
<Relationships …>
  <Relationship Id="rId1" Type="http://…/externalLinkPath"
    Target="file:///C:\Source.xlsx" TargetMode="External"/>
</Relationships>
```

This relationship indicates that the supporting workbook that C2 references resides on the local drive, at `c:\source.xlsx`.

3.13.9 Hyperlink Example

Consider the following worksheet:

	A	B	C
1		**Link to an external workbook range:**	**Link to an external workbook name:**
2		=SUM('C:\[Source.xlsx]Sheet1'!$A$1:$A$3)	='C:\Source2.xlsx'!NameInExternalWorkbook
3			
4		W3C Hyperlink	
5			

Cell B4 contains a hyperlink, whose friendly name is "W3C Hyperlink", and whose target is "`http://www.w3.org/`".

3.13.10 Worksheet XML

See §3.13.3 for the full XML. Here is the snippet expressing the hyperlink information, whose collection appears immediately after the sheetData collection in this example.

```
<hyperlinks>
  <hyperlink ref="B4" r:id="rId1"/>
</hyperlinks>
```

The hyperlink XML indicates that cell B4 of this sheet has a hyperlink, whose target can be found by following the relationship whose Id="rId1" from the current sheet. The 'friendly' name of the hyperlink is stored in the cell definition.

3.13.11 Relationship

The corresponding content from sheet1.xml.rels is:

```
<Relationships …>
  <Relationship Id="rId1" Type="http://…/hyperlink"
    Target="http://www.w3.org/" TargetMode="External"/>
</Relationships>
```

This hyperlink points external to the workbook (TargetMode="External"), and the URL is found in the value of Target to be "`http://www.w3.org/`".

3.14 Volatile Dependencies

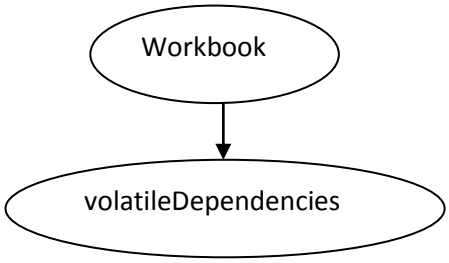
3.14.1 Overview

The volatile dependencies part provides a supporting cache of data for Real Time Data (RTD) and CUBE functions in the workbook. Both of these types of functions require connectivity to external servers to retrieve their data. For RTD functions, an RTD interface has been defined for how to provide (on the server side) and retrieve (on the client side) external data. Similarly, CUBE functions are able to access data in OLAP cubes. Each type of function has its own function syntax resulting in a specific piece of information being returned.

In the event that the server providing the data is unavailable, a spreadsheet application may want to cache the most recently retrieved values so that when recalculating the spreadsheet, calculated results may be acquired instead of errors.

The volatile dependencies part provides that cache of data and supporting information about these functions and their data servers and connections.

3.14.2 File Architecture - Relationships



The workbook holds the relationship to the volatile dependencies part.

3.14.3 Example

In this example, both a Real Time Data (RTD) function and CUBE functions are in use.

3.14.3.1 Illustration

	A	B
1	aaa: 672	
2		
3	Corporate	
4	$80,450,596.98	
5	Set	
6	#N/A	
7	#N/A	
8	Not Applicable	
9	Growth in Customer Base Goal	
10		

(values shown)

	A
1	=RTD("jrtdx.rtd",,"aaa")
2	
3	=CUBEMEMBER("xlextdat9 Adventure Works DW Adventure Works","[Department].[Departments].[Corporate]")
4	=CUBEVALUE("xlextdat9 Adventure Works DW Adventure Works",A3)
5	=CUBESET("xlextdat9 Adventure Works DW Adventure Works","[Customer].[Customer Geography].[All Customers].[United Kingdom].children","Set")
6	=CUBERANKEDMEMBER("xlextdat9 Adventure Works DW Adventure Works",$A$3,ROW(A1))
7	=CUBESETCOUNT(A3)
8	=CUBEMEMBERPROPERTY("xlextdat9 Adventure Works DW Adventure Works","[Product].[Product].[All Products].[Blade]","Class")
9	=CUBEKPIMEMBER("xlextdat9 Adventure Works DW Adventure Works","Growth in Customer Base",2)
10	

(functions shown)

3.14.3.2 volatileDependencies.xml

```
<volTypes xmlns="…">
  <volType type="realTimeData">
    <main first="jrtdx.rtd">
      <tp t="s">
        <v>aaa: 4447</v>
        <stp/>
        <stp>aaa</stp>
        <tr r="A1" s="1"/>
      </tp>
    </main>
  </volType>
  <volType type="olapFunctions">
    <main first="xlextdat9 Adventure Works DW Adventure Works">
      <tp t="e">
        <v>#N/A</v>
        <stp>1</stp>
        <tr r="A6" s="1"/>
        <tr r="A9" s="1"/>
        <tr r="A8" s="1"/>
        <tr r="A5" s="1"/>
        <tr r="A4" s="1"/>
        <tr r="A3" s="1"/>
      </tp>
    </main>
  </volType>
</volTypes>
```

3.14.3.2.1 RTD Supporting Data

`/voltypes/volType@type` indicates that the supporting information pertains to an RTD function call. Valid values are `realTimeData` and `olapFunctions`

/volTypes/volType/main@first indicates the ProgId of the RTD server. This value corresponds to the first argument of an RTD function in a worksheet.

/volTypes/volType/main/tp contains a listing of topics within the main topic. For the RTD function, this collection will express the remaining parameters of the function, and indicate the last known value and data type of that value.

/volTypes/volType/main/tp@t indicates the data type of the value associated with this topics. For this RTD example, the value is "aaa: 4447" whose data type is string.

/volTypes/volType/main/tp/v expresses the last known value of this RTD function, "aaa: 4447".

/volTypes/volType/main/tp/stp expresses the remaining topics, or function parameters, for this RTD function. Notice that in the example, the second parameter is left empty, and the third parameter is "aaa".

/volTypes/volType/main/tp/tr expresses the cells which are dependent on this particular set of topics, and which are associated with this supporting information.

3.14.3.2.2 Cube Function Supporting Data

Cube functions use the same persistence structure as the RTD supporting data, but the information is interpreted slightly differently. At a high level, main@first indicates the connection name, and the tr elements spell out the cells with cube function calls dependent on this connection. In most cases (when the <stp> value is equal to "1") the remaining information can be ignored.

/volTypes/volType/main@first indicates the connection name for the related cube functions.

/volTypes/volType/main/tp@t can be ignored when stp value is 1.

/volTypes/volType/main/tp/v contains an error value of "#N/A", which can be ignored when stp value is 1.

/volTypes/volType/main/tp/stp value of 1 indicates that all of the related cells with calling cube functions have been refreshed.

/volTypes/volType/main/tp/tr expresses the cells contain cube functions which are dependent on this connection, and which are associated with this supporting information.

3.15 Custom XML Mappings

3.15.1 Overview

With the pervasiveness of XML data structures and XML web services, it is appropriate for a spreadsheet application to consume XML data structures and render the data in the sheet grid. Furthermore it is appropriate and desirable for the spreadsheet application to be able to generate XML data structures. Finally, since XML is extensible, the kinds of XML structures that can be consumed or produced by a spreadsheet application should be as varied as the number of XML schemas that exist.

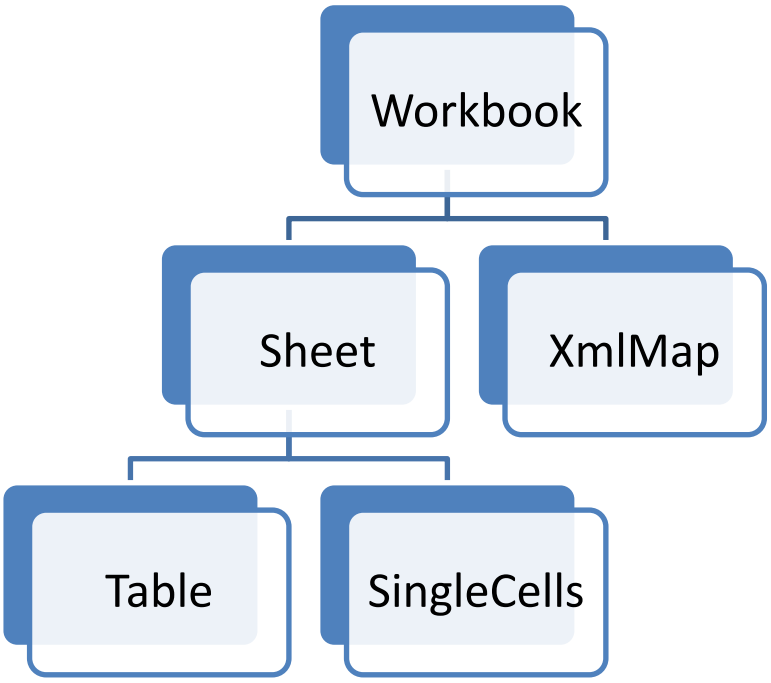
The XML Mapping feature enables adding arbitrary XML data structures and arbitrary XML schema definitions to the workbook, then mapping the various XML nodes to cells and ranges in the workbook. Once an XML Mapping is set up, the application is able to import and export XML instance structures according to the schema definition.

While the original schema or XML definition may reside on disk or at some file location outside the workbook, a copy of the schema is stored in the workbook.

Every time an XML instance or schema is added to the workbook, a new map object is created which ties together the schemas and where the various elements are mapped in the workbook.

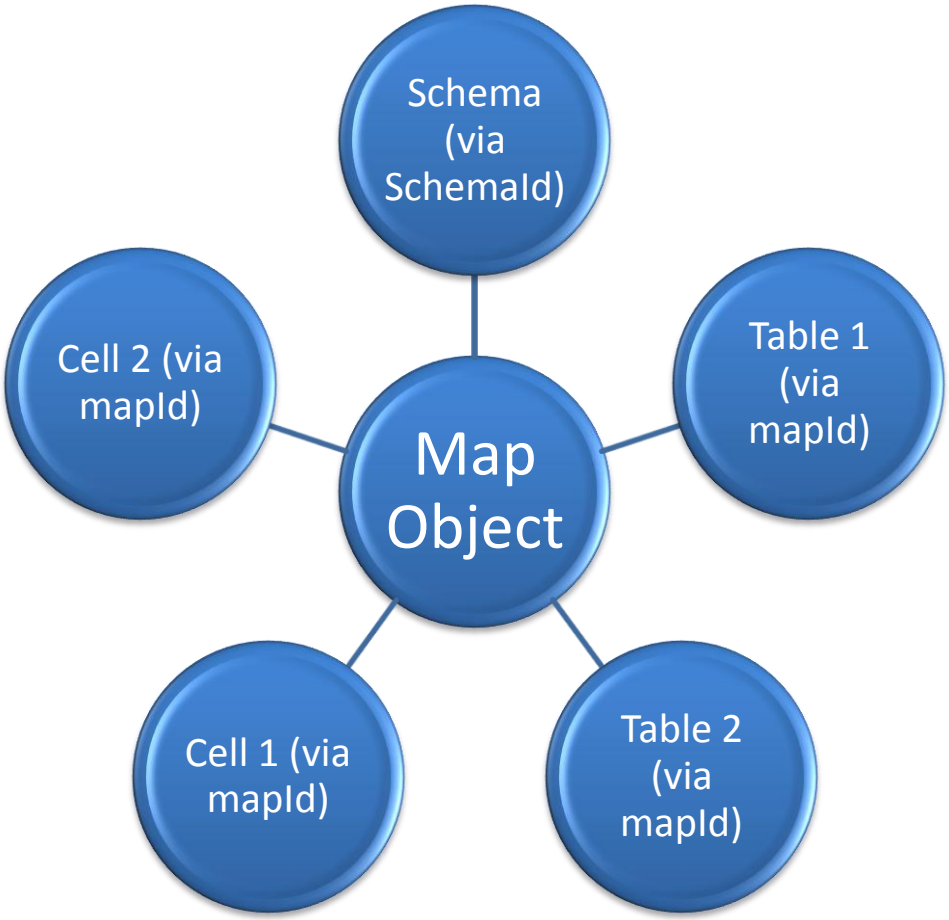
Additional properties are stored on each cell and each column of a Table that has an XML map association.

3.15.2 File Architecture - Relationships



The workbook owns sheets and the xmlMap definitions. Each sheet references Tables and single cells which are mapped to XML structures.

3.15.3 Conceptual Model

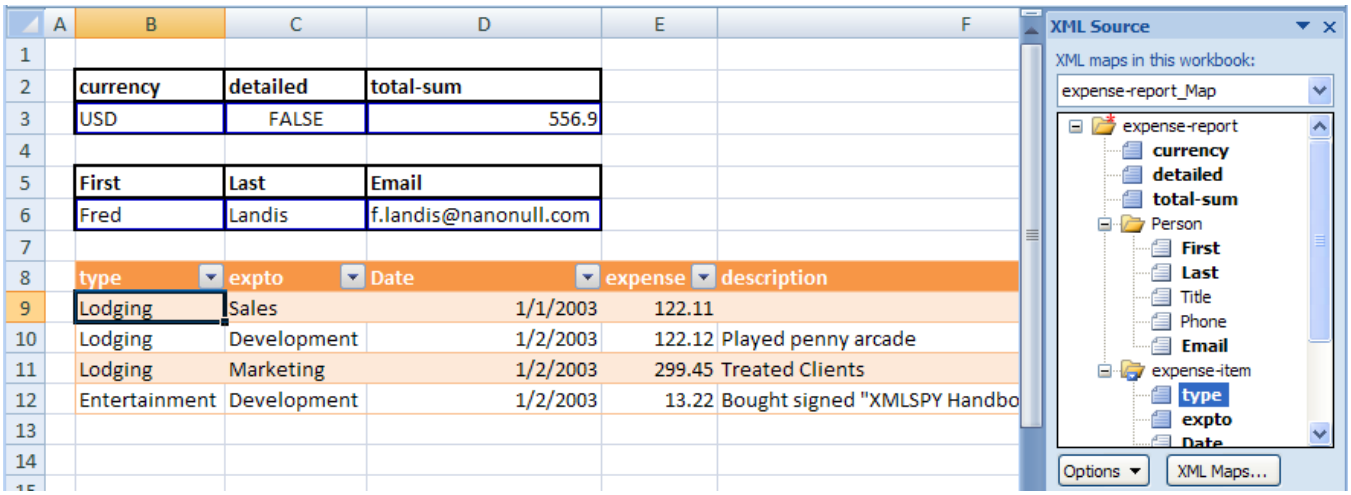


Conceptually all the objects reference a common Map Object, by `@id`. It is in this way that they come together into a single working feature that can import and export custom XML data.

3.15.4 Example

In this example both single cells and a Table are shown to have a data binding to an XML structure.

3.15.4.1 Illustration



currency	detailed	total-sum
USD	FALSE	556.9

First	Last	Email
Fred	Landis	f.landis@nanonull.com

type	expto	Date	expense	description
Lodging	Sales	1/1/2003	122.11	
Lodging	Development	1/2/2003	122.12	Played penny arcade
Lodging	Marketing	1/2/2003	299.45	Treated Clients
Entertainment	Development	1/2/2003	13.22	Bought signed "XMLSPY Handbo

The table in B8:G12 is also data bound to an XML mapping object. The first column, titled "type", is associated with the XML Map named "expense-report_Map", specifically the attribute identified by the xpath expression `/expense-report/expense-item@type` pointing into the corresponding XML structure. In similar fashion, each of the columns in the Table correspond with elements or attributes in the related XML Map structure.

Additionally, cells B3:D3 and B6:D6 are each bound to a single, non-repeating element or attribute from the same XML Map structure. For example, cell B3 corresponds to `/expense-report@currency`.

In this way XML instance structures can be refreshed into the cells and Table region, and XML instance structures can be generated from the data in those ranges of the spreadsheet. In other words, XML structures can be imported and exported to and from the worksheet via the XML Mapping feature.

3.15.4.2 The xmlMap XML

The xmlMaps part stores the custom schema that has been added to the workbook, and also stores the xmlMap definitions. There can be multiple schemas and xmlMaps in a single workbook.

```
<MapInfo SelectionNamespaces="">
    <Schema ID="Schema1">
        <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <xsd:element nillable="true" name="expense-report">
                <xsd:complexType>
                    <xsd:sequence minOccurs="0">
                        <xsd:element minOccurs="0" nillable="true"
name="Person" form="unqualified">
                            <xsd:complexType>
                                <xsd:sequence minOccurs="0">
                                    <xsd:element minOccurs="0" nillable="true"
type="xsd:string" name="First" form="unqualified"></xsd:element>
```

```
                                    <xsd:element minOccurs="0" nillable="true"
type="xsd:string" name="Last" form="unqualified"></xsd:element>
                                    <xsd:element minOccurs="0" nillable="true"
type="xsd:string" name="Title" form="unqualified"></xsd:element>
                                    <xsd:element minOccurs="0" nillable="true"
type="xsd:string" name="Phone" form="unqualified"></xsd:element>
                                    <xsd:element minOccurs="0" nillable="true"
type="xsd:string" name="Email" form="unqualified"></xsd:element>
                                </xsd:sequence>
                            </xsd:complexType></xsd:element>
                        <xsd:element minOccurs="0" maxOccurs="unbounded"
nillable="true" name="expense-item" form="unqualified">
                            <xsd:complexType>
                                <xsd:all>
                                    <xsd:element minOccurs="0" nillable="true"
type="xsd:date" name="Date" form="unqualified"></xsd:element>
                                    <xsd:element minOccurs="0" nillable="true"
type="xsd:double" name="expense" form="unqualified"></xsd:element>
                                    <xsd:element minOccurs="0" nillable="true"
type="xsd:string" name="description" form="unqualified"></xsd:element>
                                    <xsd:element minOccurs="0" nillable="true"
name="Misc" form="unqualified">
                                        <xsd:complexType>
                                            <xsd:attribute name="misctype"
form="unqualified" type="xsd:string"></xsd:attribute>
                                        </xsd:complexType>
                                    </xsd:element>
                                </xsd:all>
                                <xsd:attribute name="type" form="unqualified"
type="xsd:string"></xsd:attribute>
                                <xsd:attribute name="expto" form="unqualified"
type="xsd:string"></xsd:attribute>
                            </xsd:complexType>
                        </xsd:element>
                    </xsd:sequence>
                    <xsd:attribute name="currency" form="unqualified"
type="xsd:string"></xsd:attribute>
                    <xsd:attribute name="detailed" form="unqualified"
type="xsd:boolean"></xsd:attribute>
                    <xsd:attribute name="total-sum" form="unqualified"
type="xsd:double"></xsd:attribute>
                </xsd:complexType>
            </xsd:element>
```

```
            </xsd:schema>
        </Schema>
        <Map ID="1" Name="expense-report_Map" RootElement="expense-report"
    SchemaID="Schema1" ShowImportExportValidationErrors="false" AutoFit="true"
    Append="false" PreserveSortAFLayout="true" PreserveFormat="true">
            <DataBinding FileBinding="true" DataBindingLoadMode="1"/>
        </Map>
    </MapInfo>
```

- /MapInfo@SelectionNamespaces ties the prefix to the actual namespace. This is used when writing xpath expressions at runtime against the XML instance structures, because the xpath expressions use namespace prefixes instead of the fully spelled out namespace.
- /MapInfo/Schema stores the schemas for a particular XML map object. There can be multiple <Schema> elements in a workbook, one for each XML map.
- /MapInfo/Schema@ID identifies the schema collection used to define a particular XML map object.
- /MapInfo/Map/@ID identifies the map object.
- /MapInfo/Map@Name is the friendly name of the map object.
- /MapInfo/Map@RootElement is the name of the root element of the XML instance (schemas can define more than one root node).
- /MapInfo/Map@SchemaID identifies which schema collection the map uses.
- /MapInfo/Map@ShowImportExportValidationErrors indicates that when an XML instance is imported or exported, the schema should be used to validate the instance, and schema errors should be shown to the user.
- /MapInfo/Map@AutoFit indicates that after refresh, all the cells should be 'best fitted'.
- /MapInfo/Map@Append means that when refreshed, don't discard existing data, but append new data to it.
- /MapInfo/Map@PreserveSortAFLayout indicates whether to keep filters on (Tables).
- /MapInfo/Map@PreserveFormat indicates whether to keep the cell formatting applied or re-apply based on schema data type.

3.15.4.3 The Table XML

The only difference with table definitions that are bound to XML is that @tableType="xml" and each column has an additional set of xml-specific properties, contained in the <xmlColumnPr> collection, which appears once for every column in the Table which has an XML data binding.

```
<table xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/5/main"
id="7" name="Table7" displayName="Table7" ref="B8:G12" tableType="xml"
totalsRowShown="0" connectionId="1">
    <autoFilter ref="B8:G12"/>
    <tableColumns count="6">
        <tableColumn id="1" uniqueName="type" name="type">
            <xmlColumnPr mapId="1" xpath="/expense-report/expense-item/@type"
xmlDataType="string"/>
```

```
            </tableColumn>
            <tableColumn id="2" uniqueName="expto" name="expto">
                <xmlColumnPr mapId="1" xpath="/expense-report/expense-item/@expto"
    xmlDataType="string"/>
            </tableColumn>
            <tableColumn id="3" uniqueName="Date" name="Date">
                <xmlColumnPr mapId="1" xpath="/expense-report/expense-item/Date"
    xmlDataType="date"/>
            </tableColumn>
            <tableColumn id="4" uniqueName="expense" name="expense">
                <xmlColumnPr mapId="1" xpath="/expense-report/expense-item/expense"
    xmlDataType="double"/>
            </tableColumn>
            <tableColumn id="5" uniqueName="description" name="description">
                <xmlColumnPr mapId="1" xpath="/expense-report/expense-
    item/description" xmlDataType="string"/>
            </tableColumn>
            <tableColumn id="6" uniqueName="misctype" name="misctype">
                <xmlColumnPr mapId="1" xpath="/expense-report/expense-
    item/Misc/@misctype" xmlDataType="string"/>
            </tableColumn>
        </tableColumns>
        <tableStyleInfo name="TableStyleMedium7" showFirstColumn="0"
    showLastColumn="0" showRowStripes="1" showColumnStripes="0"/>
    </table>
```

The column in the Table titled "type" is bound to an XML mapping, whose map object Id `@mapId` is "1". The `@xpath` value indicates an xpath expression to which this Table column is associated. In this example the Table column "type" corresponds to `@type`, which is an attribute of the `<expense-item>` collection. The corresponding custom schema definition for `@type` indicates a data type of string. This is stored as an xml column property as well, in `@xmlDataType`. This is used for interpreting the data on import and export, and is also used to format the cells for proper rendering in the range.

The remaining columns have similar properties set and can be understood from the discussion above.

3.15.4.4 Single Cell XML

Contents of tableSingleCells.xml

```
    <singleXmlCells
    xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/5/main">
        <singleXmlCell id="1" name="Table1" displayName="Table1" r="B3"
    connectionId="1">
            <xmlCellPr id="1" uniqueName="currency">
```

```
            <xmlPr mapId="1" xpath="/expense-report/@currency"
xmlDataType="string"/>
        </xmlCellPr>
    </singleXmlCell>
    <singleXmlCell id="2" name="Table2" displayName="Table2" r="C3"
connectionId="1">
        <xmlCellPr id="1" uniqueName="detailed">
            <xmlPr mapId="1" xpath="/expense-report/@detailed"
xmlDataType="boolean"/>
        </xmlCellPr>
    </singleXmlCell>
    <singleXmlCell id="3" name="Table3" displayName="Table3" r="D3"
connectionId="1">
        <xmlCellPr id="1" uniqueName="total-sum">
            <xmlPr mapId="1" xpath="/expense-report/@total-sum"
xmlDataType="double"/>
        </xmlCellPr>
    </singleXmlCell>
    <singleXmlCell id="4" name="Table4" displayName="Table4" r="B6"
connectionId="1">
        <xmlCellPr id="1" uniqueName="First">
            <xmlPr mapId="1" xpath="/expense-report/Person/First"
xmlDataType="string"/>
        </xmlCellPr>
    </singleXmlCell>
    <singleXmlCell id="5" name="Table5" displayName="Table5" r="C6"
connectionId="1">
        <xmlCellPr id="1" uniqueName="Last">
            <xmlPr mapId="1" xpath="/expense-report/Person/Last"
xmlDataType="string"/>
        </xmlCellPr>
    </singleXmlCell>
    <singleXmlCell id="6" name="Table6" displayName="Table6" r="D6"
connectionId="1">
        <xmlCellPr id="1" uniqueName="Email">
            <xmlPr mapId="1" xpath="/expense-report/Person/Email"
xmlDataType="string"/>
        </xmlCellPr>
    </singleXmlCell>
</singleXmlCells>
```

A single cell which has been mapped to an XML node is expressed in much the same way that an entire table is expressed.

The <singleXmlCell> collection is the top level object, like the Table, which identifies the cell in question.

The <xmlCellPr> collection identifies the name for the only 'column' in this structure, the single cell. In this way it is much like a table column definition and the table column-level properties.

The <xmlPr> collection expresses the xml properties for this cell.

3.16 Formulas

3.16.1 Introduction

A SpreadsheetML *formula* is an equation that performs a calculation that typically involves the values of one or more cells in one or more worksheets.

A formula is an expression that can contain the following: constants (§3.16.2), operators (§3.16.3), cell references (§3.16.4), calls to functions (§3.16.5), and names (§3.16.6).

Consider the formula `PI()*(A2^2)`. In this case,

- `PI()` results in a call to the function `PI`, which returns the value of π.
- The cell reference `A2` returns the value in that cell.
- `2` is a numeric constant.
- The caret (^) operator raises its left operand to the power of its right operand.
- The parentheses, `(` and `)`, are used for grouping.
- The asterisk (*) operator performs multiplication of its two operands.

3.16.2 Constants

A *constant* is a predefined value that is not calculated, and, therefore, does not change. A constant can be any of the following:

- A real number
- The logical values TRUE and FALSE
- A string literal
- An array constant
- An error constant

3.16.3 Operators

An *operator* is a symbol that specifies the type of operation to perform on one or more operands. There are arithmetic, comparison, text, and reference operators.

Operators			
Family	*Operator*	*Description*	*Precedence*

Operators			
Reference operators	:	Binary range operator, which takes two cell reference (§3.16.3) operands, and results in one reference to the cells inclusive of, and between, those references. For example, `SUM(B5:C15)`, which references 11 cells.	highest
	,	Binary union operator, which takes two cell reference (§3.16.3) operands, and results in one reference to all those, possibly non-contiguous, cells. For example, `SUM((B5:B15,D5:D15))`, which references 22 cells, 11 from column B, and 11 from column D. The grouping parentheses are necessary to indicate that the comma is an operator rather than a punctuator separating two arguments.	
	space	Binary intersection operator, which takes two cell reference (§3.16.3) operands, and results in one reference to those, possibly non-contiguous, cells that are common. If the intersection is empty, the result is `#NULL!`. For example, `COUNT((B1:C1) (C1:D1))`, which results in a reference to `C1`, while `COUNT((B1:D1) (B1,D1))` results in a single reference to `B1` and `D1`.	
Arithmetic operators	-	Unary minus	
	%	Percentage (unary postfix), which divides its operand by 100. For example, `10.5%`, which results in `0.105`.	
	^	Exponentiation	
	*	Multiplication	
	/	Division	
	+	Addition	
	-	Subtraction	
Text operator	&	Text concatenation (Each of the two operands is converted to text, if necessary, before concatenation.)	
Comparison operators	=	Equal-to	lowest
	<>	Not-equal-to	
	<	Less-than	
	<=	Less-than or equal-to	
	>	Greater-than	

Operators			
	>=	Greater-than-or-equal-to	

Given that cell E38 contains the value 4, and cell F38 contains the value 2, the formula

```
((-1+E38^2)*3-F38)/2
```

produces the result 21.5.

3.16.4 Cell References

Each set of horizontal cells in a worksheet is a *row*, and each set of vertical cells is a *column*. A cell's row and column combination designates the location of that cell.

A *cell reference* designates one or more cells on the same worksheet. Using references, one can:

- Use data contained in different parts of the same worksheet in a single formula.
- Use the value from a single cell in several formulas.
- Refer to cells on other sheets in the same workbook, and even to other workbooks. (References to cells in other workbooks are called *links*.)

There are two cell reference styles: A1 and R1C1.

- In the A1 reference style, each row has a numeric heading numbered sequentially from the top down, starting at 1. Each column has an alphabetic heading named sequentially from left-to-right, A–Z, then AA–AZ, BA–BZ, …, ZA–ZZ, AAA–AAZ, ABA–ABZ, and so on. Column letters are not case-sensitive.

  A relative reference to a single cell is written as its column letter immediately followed by its row number. A relative reference to a whole row is written as its row number. A relative reference to a whole column is written as its column letter. A reference to a range of two or more cells is written as two single-cell references separated by the binary range operator (:). An absolute A1 reference is made up of a cell's column letter followed by its row number, with each being preceded by a dollar character ($). For example, A2, B34, and B5:D8 are relative A1 references. $A$2, $B$34, and $B$5:$D$8 are absolute A1 references. $A2, B$34, and $B5:D$8 are mixed A1 references.
- In the R1C1 reference style, each row has a numeric heading numbered sequentially from the top down, starting at 1. Each column has a numeric heading numbered sequentially from left-to-right, starting at 1.

  A whole row is referenced by omitting the column, and a whole column is referenced by omitting the row. An absolute row or column reference uses absolute row or column numbers, respectively. A relative row or column reference uses, respectively, row or column offsets from the cell containing the formula, with a negative offset indicating a row to the left or a column above, and a positive offset indicateing a row to the right or a column below. Specifying an offset of zero is equivalent to omitting

that offset and its delimiting brackets. For example, `R[-2]C` refers to the cell two rows up and in the same column, `R[2]C[2]` refers to the cell two rows down and two columns to the right, `R2C2` refers to the cell in the second row and in the second column, `R[-1]` refers to the entire row above the active cell, and `R` refers to the current row.

The R1C1 alternate reference style can only be used at runtime.

3.16.5 Functions

A *function* is a named formula that takes zero or more arguments, performs an operation, and, optionally, returns a result. Some examples of function calls are: `PI(), POWER(A1,B3)`, and `SUM(C6:C10)`.

There are more than 300 predefined functions defined by this Office Open XML specification. User-defined functions are also permitted.

3.16.6 Names

A *name* is an alias for a constant, a cell reference, or a formula. A name in a formula can make it easier to understand the purpose of that formula. For example, the formula `SUM(FirstQuarterSales)` is easier to identify than `SUM(C20:C30)`.

3.16.7 Types and Values

Each *expression* has a type. SpreadsheetML formulas support the following types: array, error, logical, number, and text.

An array value or constant represents a collection of one or more elements, whose values can have any type (i.e., the elements of an array need not all have the same type).

3.16.8 Error values

The evaluation of an expression can result in an error having one of a number of *error value*s. These error values are:

Error Value	Reason for Occurrence
`#DIV/0!`	Intended to indicate when any number, including zero, is divided by zero.
`#N/A`	Intended to indicate when a designated value is not available. For example, Some functions, such as `SUMX2MY2`, perform a series of operations on corresponding elements in two arrays. If those arrays do not have the same number of elements, then for some elements in the longer array, there are no corresponding elements in the shorter one; that is, one or more values in the shorter array are not available. This error value can be produced by calling the function `NA`.
`#NAME?`	Intended to indicate when what looks like a name is used, but no such name has been defined. For example, `XYZ/3`, where `XYZ` is not a defined name. `Total is & A10`, where neither `Total` nor `is` is a defined name. Presumably, `"Total is " & A10` was intended. `SUM(A1C10)`, where the range `A1:C10` was intended.

Error Value	Reason for Occurrence
#NULL!	Intended to indicate when two areas are required to intersect, but do not. For example, In the case of SUM(B1 C1), the space between B1 and C1 is treated as the binary intersection operator, when a comma was intended.
#NUM!	Intended to indicate when an argument to a function has a compatible type, but has a value that is outside the domain over which that function is defined. (This is known as a *domain error*.) For example, Certain calls to ASIN, ATANH, FACT, and SQRT might result in domain errors. Intended to indicate that the result of a function cannot be represented in a value of the specified type, typically due to extreme magnitude. (This is known as a *range error*.) For example, FACT(1000) might result in a range error.
#REF!	Intended to indicate when a cell reference is invalid. For example, If a formula contains a reference to a cell, and then the row or column containing that cell is deleted, a #REF! error results. If a worksheet does not support 20,001 columns, OFFSET(A1,0,20000) will result in a #REF! error.
#VALUE!	Intended to indicate when an incompatible type argument is passed to a function, or an incompatible type operand is used with an operator. For example, In the case of a function argument, a number was expected, but text was provided. In the case of 1+"ABC", the binary addition operator is not defined for text.

3.16.9 Dates and Times

Each unique instant in SpreadsheetML time is represented as a distinct non-negative numeric *serial value*, which is made up of an integer date component and a fractional time component. As dates and times are numeric values, they can take part in arithmetic operations.

Numerous functions take as arguments one or more serial values or strings representing dates and/or times. Functions that care only about the date shall ignore any time information that is provided. Functions that care only about the time shall ignore any date information that is provided.

3.16.9.1 Date Representation

Going forward in time, the date component of a serial value increases by 1 each day.

There are two different bases for serial values:

- In the *1900 date base system*, the lower limit is January 1, 1900, which has serial value 1. The upper-limit is December 31, 9999, which has serial value 2,958,465.
- In the *1904 date base system*, the lower limit is January 1, 1904, which has serial value 0. The upper-limit is December 31, 9999, which has serial value 2,957,003.

As to which date base system an implementation uses by default or whether it allows its users to switch between date base systems, is unspecified.

For the 1900 date base system:

DATEVALUE("01-Jan-1900") results in the serial value 1.0000000…
DATEVALUE("03-Feb-1910") results in the serial value 3687.0000000…
DATEVALUE("01-Feb-2006") results in the serial value 38749.0000000…
DATEVALUE("31-Dec-9999") results in the serial value 2958465.0000000…

For the 1904 date base system:

DATEVALUE("01-Jan-1904") results in the serial value 0.0000000…
DATEVALUE("03-Feb-1910") results in the serial value 2225.0000000…
DATEVALUE("01-Feb-2006") results in the serial value 37287.0000000…
DATEVALUE("31-Dec-9999") results in the serial value 2957003.0000000…

3.16.9.2 Time Representation

The time component of a serial value ranges in value from 0–0.99999999, and represents times from 0:00:00 (12:00:00 AM) to 23:59:59 (11:59:59 P.M.), respectively.

Going forward in time, the time component of a serial value increases by 1/86,400 each second. (As such, the time 12:00 has a serial value time component of 0.5.)

TIMEVALUE("00:00:00") results in the serial value 0.0000000…
TIMEVALUE("00:00:01") results in the serial value 0.0000115…
TIMEVALUE("10:05:54") results in the serial value 0.4207639…
TIMEVALUE("12:00:00") results in the serial value 0.5000000…
TIMEVALUE("23:59:59") results in the serial value 0.9999884…

3.16.9.3 Combined Date and Time Representation

Any date component can be added to any time component to produce a serial value for that date/time combination.

For the 1900 date base system:

DATE(1910,2,3)+TIME(10,5,54) results in the serial value 3687.4207639…
DATE(1900,1,1)+TIME(12,0,0) results in the serial value 1.5000000…
DATE(9999,12,31)+TIME(23,59,59) results in the serial value 2958465.9999884…

For the 1904 date base system:

DATE(1910,2,3)+TIME(10,5,54) results in the serial value 2225.4207639…
DATE(1904,1,1)+TIME(12,0,0) results in the serial value 0.5000000…
DATE(9999,12,31)+TIME(23,59,59) results in the serial value 2957003.9999884…

3.16.10 XML Representation

A formula is represented in a worksheet's XML by an f element that contains the text of the formula, and a v element that contains the text version of the last computed value for that formula. This pair of elements is inside a c element, which is, in turn, is inside a row element. Consider the scalar formula SQRT(C2^2+D2^2), where C2 refers to a cell containing the number 12.5, and D2 refers to a cell containing the number 9.6. The corresponding XML might be as follows:

```
<row r="2" spans="2:4">
  <c r="B2" s="40">
    <f>SQRT(C2^2+D2^2)</f>
    <v>15.761027885261798</v>
  </c>
  <c r="C2" s="0">
    <v>12.5</v>
  </c>
  <c r="D2" s="0">
    <v>9.6</v>
  </c>
</row>
```

In the scalar formula CONCATENATE("The total is ",C7," units"), C7 refers to a cell containing the number 23. The corresponding XML might be as follows:

```
<row r="7" spans="2:4" ht="285">
  <c r="B7" s="4" t="str">
    <f>CONCATENATE("The total is ",C7," units")</f>
    <v>The total is 23 units</v>
  </c>
  <c r="C7" s="0">
    <v>23</v>
  </c>
</row>
```

As the function CONCATENATE returns a string, the value for the cell's t attribute is str.

4. Introduction to PresentationML

This clause is informative.

This clause contains a detailed introduction to the structure of a PresentationML document.

The PresentationML file format can be broken down into the following subjects:

- Presentation
- Slides
- Slide Content
- Animation

There are other schemas—most notably DrawingML—that make up a sizeable chunk of the PresentationML file format. These schemas are addressed separately in §5.

4.1 Basics

4.1.1 Introduction

This subclause provides a high-level overview of the content described in the following schemas: pml-baseTypes.xsd, pml-presentation.xsd, pml-presentationProperties.xsd, and pml-viewProperties.xsd.

The PresentationML file format can be broken down into the following subjects:

- Presentation
- Slides
- Slide Content
- Animation

The best way to understand the content in each of these subjects is to cover them in that particular order.

The eight schemas that collectively represent the PresentationML file format can be grouped by subject as follows:

Presentation	Slide	Slide content	Animation
pml-baseTypes.xsd	pml-slide.xsd	pml-embedding.xsd	pml-animationInfo.xsd
pml-presentation.xsd		pml-comments.xsd	
pml-presentationProperties.xsd			
pml-viewProperties.xsd			

There are other schemas—most notably DrawingML—that make up a sizeable chunk of the PresentationML file format. These schemas are addressed separately.

This subclause introduces the first subject, “Presentation”. Other subclauses build on this foundation.

4.1.2 Basic Utilities

The schema pml-baseTypes.xsd contains a set of complex types and simple types that are used by other schemas. The types, or utilities, are used in a variety of cases. Their single implementation provides for rapid and less error-prone changes throughout an implementation.

To provide some insight into the type of information that is being repurposed, here is a sample of these utilities:

- Empty Element
- Name
- Direction
- Index and Index Range
- Slide Show ID
- Slide List Choice
- Slide Relationship
- Customer Data
- Future Extensibility

Each of these is discussed in the following subclauses.

4.1.2.1 Empty Element

Sometimes, the simple presence of an element is sufficient to convey meaning. That is, in some cases, you do not necessarily need information to be a Boolean, an integer, or complex type.

A simple example is the Show Type element group. In this case, a slide show can be one of three types: present, browse, or kiosk. The schema for this element group is as follows:

```
<xsd:group name="EG_ShowType">
  <xsd:choice>
    <xsd:element name="present" type="CT_Empty">
    </xsd:element>
    <xsd:element name="browse" type="CT_ShowInfoBrowse">
    </xsd:element>
    <xsd:element name="kiosk" type="CT_Empty">
    </xsd:element>
  </xsd:choice>
</xsd:group>
```

4.1.2.2 Name

Many constructs within a presentation have names associated with them. In some cases, the names are machine-generated, such as shape names (e.g., rectangle1), while others are user-defined, such as slide shows (e.g., customer-ready).

In one implementation the name simple type is simply an xsd:string. The intent is to restrict this to the appropriate pattern allowed for named constructs. The tentative restriction pattern is:

```
[ \t]*[^ \t].*
```

4.1.2.3 Direction

This multi-purpose simple type is used to convey horizontal versus vertical direction of a variety of types. Such usage can be found in the definition of slide transitions and various shape effects.

4.1.2.4 Index and Index Range

These two utilities are generally used to denote a contiguous set of items within a list. The classic example of usage would be the selection of a set of slides to print.

From a schema-perspective, there is no way to enforce that the start index be equal to or less than the end index.

4.1.2.5 Slide Show ID

This defines the ID for a slide show (also called a custom show). Because slide shows can be named, and that name can change, an implementation needs a method of referring to a slide show that can withstand name changes made by the user. In many cases, for example, with a slide, we can leverage the fact that each slide has a part within the package, in which case we can use the relationship ID. However, since there is no part for each slide show, we are forced to generate an unsigned integer for each slide show and use that.

There is nothing in the schema that prevents two or more slide shows from having the same ID.

4.1.2.6 Slide List Choice

There are many cases in which a user needs to specify a set of slides for an operation. The canonical example is what slides to include in your slide show. Because this operation is frequently required in the file format, one implementation has provided a utility to facilitate this:

```
<xsd:group name="EG_SlideListChoice">
  <xsd:choice>
    <xsd:element name="sldAll" type="CT_Empty" />
    <xsd:element name="sldRg" type="CT_IndexRange" />
    <xsd:element name="custShow" type="CT_CustomShowId" />
  </xsd:choice>
</xsd:group>
```

As the schema above declares, when selecting a set of slides, the user can select all of the slides, a slide range (by declaring a pair of start and end indices) or a particular custom show.

4.1.2.7 Slide Relationship

As described in the Slide Show ID paragraphs above, there are many situations where the format needs to store an ordered list of slides, and does so by storing their slide IDs. This is implemented using two types: a list entry complex type and a list complex type:

```
<xsd:complexType name="CT_SlideRelationshipListEntry">
  <xsd:attribute ref="r:id" use="required"/>
</xsd:complexType>
<xsd:complexType name="CT_SlideRelationshipList">
  <xsd:sequence>
    <xsd:element name="sld" type="CT_SlideRelationshipListEntry"
      minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

4.1.2.8 Customer Data

There is a set of utilities that facilitate the storage of customer XML data within the file format. Although a topic for a separate paper, essentially, this functionality comes down to the ability to store customer-defined XML in the file format in a way that it can be easily queried, modified and/or surfaced in the presentation. Suffice it to say, the data is stored in a separate part within the package, and hence the utility pairs the object using it with the part within the package.

4.1.2.9 Future Extensibility

There is functionality that provides the ability to extend a subset of objects within the file format for inclusion of additional data over the lifetime of the file format. The utilities provide both the ability to add an alternative representation (e.g., provide a raster image in addition to the XML data for a diagram) as well as additional properties to the objects.

4.1.3 The Presentation Object

The schema pml-presentation.xsd defines the content of the principal or start part for a PresentationML document. This content includes both structural and presentation-level data for the presentation.

Astute readers will quickly identify an apparent duplication of presentation-level data, as there is also a separate schema file, pml-presentationProperties.xsd, which contains presentation-level data. That being said, there is actually no duplication. Rather, the differentiation of what presentation-level data goes into which part is based on two user scenarios: document signatures and document sanitization.

In a document signature scenario, assume a user digitally signs a presentation. There exist two types of data within the presentation package: data which changes the “content” of the presentation and data which is intended to configure an editor or the behavior of an editor. In the first case, any modification to data which

changes the “content” of the presentation must invalidate the signature; in the second case, any modification to that data should not invalidate the signature.

A classic example of this scenario deals with Kinsoku information and the publish path in the HTML settings. If the user changes the Kinsoku information in a file, the file will look (and potentially mean) something different. This is in contrast to a user setting a new HTML publish path for their particular computer.

In a document sanitization scenario, users want to remove all non-necessary information from the file. A typical usage case would be posting a presentation to a company’s Internet site. In this case, you don’t want certain configuration information publicly available. The ideal manner of removal would be to remove an entire part from the presentation package as opposed to editing a part from a package.

Going back to our Kinsoku and HTML publish path example above, the Kinsoku information needs to remain with the file. The HTML publish path could give away internal information about web servers that could be used to facilitate an attack or, more likely, simply provide information about the author to the public (e.g., the path c:\documents and settings\shawnv\webpages strongly implies that “shawnv” published this document).

Going back to the original question—what presentation-level data goes in which part—we see that data that will not invalidate a digital signature or data that should be removed during a sanitization pass should be stored in the part associated with the pml-presentationProperties.xsd schema and other presentation-level data should be stored in the part associated with the pml-presentation.xsd schema.

In addition to structural and presentation-level data defined by this schema, there are also definitions for handling customer data and future extensibility. Again, both of these will be addressed in additional papers.

4.1.3.1 Structural Information

From a structural information perspective, there are two sets of data defined in this schema: core lists and sizes.

The schema first defines a number of lists that serve as the foundation for most objects in the presentation. These lists are as follows:

- Slide IDs
- Slide Masters
- Notes Masters
- Handout Masters
- Custom Shows

It is essential that the reader fully understand the implementation of usage of these lists as they are the foundation for almost all solutions that operate—open, interrogate, modify, write—against the PresentationML file format.

As mentioned above, the lists are defined as a part of list entry and list complex types. The slide master list is defined as follows:

```
<xsd:complexType name="CT_SlideMasterIdListEntry">
  <xsd:attribute ref="r:id" use="required" />
</xsd:complexType>
<xsd:complexType name="CT_SlideMasterIdList">
  <xsd:sequence>
    <xsd:element name="sldMasterId" type="CT_SlideMasterIdListEntry"
        minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>
```

Although not complex or difficult to understand, the lists are called out because they are vital to any solution.

The next pieces of structural information are the sizes for the slides and the notes slides. By storing this information at the presentation level, the implication is that all slides (or all notes slides) in a presentation have the same size. This further implies that all slides in a presentation share the same orientation (i.e., they are all landscape-oriented or all portrait-oriented).

4.1.3.2 Presentation-Level Properties

The presentation-level properties defined in this schema can be grouped into the following groupings:

- Text-Related
- Save-Related
- Editor-Related
- Content-Related

A description for each property within each group follows.

4.1.3.2.1 Text-Related Properties

The first property stores information related to the Kinsoku settings. Kinsoku settings define the list of characters that are not allowed to start or end a line of text for a given East Asian language.

The schema definition of the Kinsoku settings is relatively straightforward: identify the language, the set of invalid start characters, and the set of invalid end characters:

```
<xsd:complexType name="CT_Kinsoku">
  <xsd:attribute name="lang" type="xsd:string" use="optional">
  </xsd:attribute>
  <xsd:attribute name="invalStChars" type="xsd:string" use="required">
  </xsd:attribute>
  <xsd:attribute name="invalEndChars" type="xsd:string" use="required">
  </xsd:attribute>
</xsd:complexType>
```

The second property stores a flag to use strict characters for starting and ending a line of Japanese text. Naturally, this is a simple Boolean attribute:

```
<xsd:attribute name="strictFirstAndLastChars"
  type="xsd:boolean" use="optional" default="true"/>
```

The final text-related property stores information related to any fonts that are embedded in the presentation. To do this, we need to store a list of embedded fonts that reference each part that stores font data (generally, there is a one-font-to-one-part mapping, although this is not a strict rule). This information is defined using three complex types:

```
<xsd:complexType name="CT_EmbeddedFontList">
  <xsd:sequence>
    <xsd:element name="embeddedFont" type="CT_EmbeddedFontListEntry"
        minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="CT_EmbeddedFontListEntry">
  <xsd:sequence>
    <xsd:element name="font" type="a:CT_TextFont" minOccurs="1"
      maxOccurs="1" />
    <xsd:element name="regular" type="CT_EmbeddedFontDataId"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element name="bold" type="CT_EmbeddedFontDataId"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element name="italic" type="CT_EmbeddedFontDataId"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element name="boldItalic" type="CT_EmbeddedFontDataId"
      minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="CT_EmbeddedFontDataId" >
  <xsd:attribute ref="r:id" use="required"/>
</xsd:complexType>
```

4.1.3.2.2 Save-Related Properties

There is a set of properties that indicate to the editor what should be saved as part of the presentation.

The first such property controls the inclusion of Personally Identifiable Information (“PII”). PII is any information that can be used to identify the author or contributor to a presentation. And while there are cases where this information is exposed visually to the user (e.g., author name in a comment shape), there are other cases where the information is not immediately evident to the user (e.g., the document author name in the list of document properties).

An implementation can provide a mechanism by which the author of a presentation can configure a file to always remove any PII that might otherwise by normally included during a regular save operation. While not a guarantee that no PII is stored in the file (e.g., consider a shape with my name in it—in some cases it describes

content in the file [my position in my group's organization chart] whereas in others it is an editorial directive ["check with ShawnV on this point"]. Given this ambiguity, we cannot solve all cases of this. As a result, this is more a convenience feature than a privacy management feature.

The second set of save-related properties has two groupings of properties. The first controls whether or not fonts will be embedded into the package representing the presentation. The second, enabled by setting the first, allows an implementation to optimize such font embedding to keep the size minimal, at the cost of future editing on other machines.

```
<xsd:attribute name="embedTrueTypeFonts" type="xsd:boolean"
  use="optional" default="false"/>
<xsd:attribute name="saveSubsetFonts" type="xsd:boolean"
  use="optional" default="false"/>
```

The user scenario behind these properties is as follows. Assume you are putting together a presentation to distribute to external customers. You happen to use an East Asian font with an on-disk file size of around 5 megabytes.

Assuming that this font is not a standard font that is widely distributed, not including this font will cause font substitution when the presentation is opened on machines that don't have a copy of the font. In any case, this can radically change the visual appearance of the presentation; in some cases, it can render the presentation unreadable.

Because you cannot afford the presentation to be unreadable or to look unprofessional, you decide to embed the font. By default, the implementation will set embedTrueTypeFonts to true and embed the entire 5 megabyte font file in the presentation package. This will clearly bloat the file, but will ensure that anyone viewing or editing this file will have the same font experience as you originally had (subject to licensing restrictions, of course).

Since you are distributing the presentation, and your primary purpose is for people to view the presentation, you can reduce the amount of font data embedded in the presentation package. By setting the second property (saveSubsetFonts) to true, only those characters in the font that were actually used to create the presentation are saved. This yields less font data stored in the file at the cost of not being able to use unused characters in future edits of the presentation on different machines.

The third property related to saving, controls whether or not an implementation can automatically compress pictures contained in the presentation. This is particularly important given the proliferation of digital cameras and scanners and the increasing importance of small files (e.g., to save network bandwidth, reduce storage required for mail and file servers, etc.).

The final property in this set specifies a password that is required to enable editing of the file using the implementation. Because this is a convenience feature intended to prevent accidental changes to information, it is stored in clear text as an xsd:string.

By storing this information in the file, the implementation will prompt the user for this password in order to open the file read/write; if the user does not provide the correct modify password, the implementation will open the file read-only.

4.1.3.2.3 Editor-Related Properties

The presentation file itself contains data that provide configuration information for the implementation's editor.

For example, the presentation can define a set of smart tags for use while editing the particular presentation. Because Smart Tags are stored in a separate part, the presentation object contains the relationship ID of the Smart Tag part.

In a fully functioning OLE server, PresentationML objects can be embedded into OLE containers, during which time a customer can set a zoom scale. This is stored in the file as a percentage called serverZoom.

An internationalized application might support configuring the editor to respect different screen orientations. For example, in regions of the world where Complex Scripts are in use, it is customary to orient the screen right-to-left. As such, a presentation can request the editor reconfigure itself for such usage scenarios.

Finally, due to changes in the file format and functionality (e.g., graphics and text engines), PresentationML introduces some end-user complexity when working collaboratively with other customers using older versions. To help remedy this, an implementation might support a Compatibility Mode, which restricts the functionality exposed by the editor to optimize the output for the best cross-version collaboration story possible. As a result, each presentation needs to opt-into this mode.

4.1.3.2.4 Content-Related Properties

This set of properties is related to the actual content in the presentation.

End-users can define the starting slide number for numbered slides in each presentation. While it typically starts at one, the user can select any positive number to begin slide numbering. The primary user scenario is when compiling a mega-presentation that is a collection of multiple presentations. A secondary user scenario is when including a presentation in the middle of or at the end of a printed document where you want the slide/page numbers to continue.

Another content-related property controls whether or not header/footer placeholders are to be shown on title slides. In many cases users will use special shapes called header and footer placeholder that contain built-in field codes that control the display of various sorts of information like the date/time and slide number.

In most cases, users like to keep their title slides as simple as possible (much like in the printed world where you want your first page to be clean and streamlined) and hence do not want data like date/times and slide numbers to show up on such slides. This attribute defines this presentation-wide.

The final property relates to creating photo albums. The implementation has a feature that allows the user to generate automatically a presentation based on a set of pictures. During this process, the user can select from a variety of settings, including, but not limited to, what pictures to include, the layout of the pictures on the

slides (e.g., one picture per slide, two pictures per slide, etc.), what type of frame shape to use, etc. All of this information is stored in the presentation for future photo album creation.

4.1.4 Presentation Properties

Those properties that apply to the presentation as a whole, and that are likely to be removed during document sanitization, or are not going to invalidate a digital signature, are defined in pml-presentationProperties.xsd. These properties can be grouped into three primary groupings.

- HTML Publish Properties
- Print Properties
- Slide Show Properties
- View Properties

In addition to this grouping, there are properties that define a Most Recently Used ("MRU") list of colors as well as providing for future extensibility. (The MRU will be discussed in a DrawingML paper and the extensibility will be discussed in a similar paper.)

4.1.4.1 HTML Publish Properties

An implementation must have the ability to save (and publish) a presentation to a web-friendly format like HTML or MHTML. Various parameters are used to configure the application for saving such formats as well as to control what content gets generated. The parameters that configure the application are the HTML Publish properties whereas the content properties are the Web Properties.

The HTML Publish properties provide the author with the ability to control what content gets displayed in the browser when the resulting file—either HTML or MHTML—is viewed using that type of an application. For example, the speaker notes can either be displayed in the frameset or can be hidden from view. This is particularly useful when a speaker's notes are not necessarily in a customer-ready format. It's useful but not necessarily secure.

The author can also specify the title to be displayed in the browser. Although this defaults to the actual file name, or if that is missing, to the content of the first slide's title placeholder, it can be overridden by the author.

Finally the author can specify a publish path to use when saving this file in this format. This is particularly useful for two reasons.

First, because there is a transformation happening, it sometimes takes a few iterations of publishing to get the browser-based experience to be exactly what you want. A classic example of this is the differing animation capabilities between the implementation and certain browsers: it is important to verify that the change in animation behavior continues to work after publishing; if you are not satisfied with the experience, sometimes you need to change the animation in the implementation and republish.

The second reason storing the path is useful is that web server paths can be cumbersome and are often not on the tip of each user's tongue. This allows the user to specify the path once and then publish using the same

location without having to re-specify it. Naturally, being stored in the file format, this allows this data to persist across session.

Indirectly, the HTML Publish properties can prime the Web Properties by defining a target web browser generation (i.e., third, fourth or third and fourth). This is done by setting the appropriate ST_HtmlPublishWebBrowserSupport attribute:

```
<xsd:complexType name="CT_HtmlPublishProperties">
  <xsd:sequence>
    <xsd:group ref="EG_SlideListChoice" minOccurs="1" maxOccurs="1">
    </xsd:group>
  </xsd:sequence>
  <xsd:attribute name="showSpeakerNotes" type="xsd:boolean"
    use="optional" default="true" />
  <xsd:attribute name="pubBrowser"
    type="ST_HtmlPublishWebBrowserSupport"
    use="optional" default="v3v4" />
  <xsd:attribute name="title" type="xsd:string" use="optional"
    default="">
  </xsd:attribute>
    <xsd:attribute ref="r:id" use="required">
  </xsd:attribute>
</xsd:complexType>
```

By providing a target generation, the Web Properties will be set to a predefined package defined for the specified browser generation. Naturally, the user can override the individual Web Property settings.

4.1.4.1.1 Web Properties

As mentioned in the previous subclause, these properties configure the output of the presentation when saved using the HTML or MHTML formats. In this case, a number of parameters can be controlled.

In all multi-slide cases where the presentation is saved using one of these formats, the implementation will create a frameset to bring the various parts of a presentation—the slide content, the speaker notes and the outline—together as well as provide for simple navigation. The color of the HTML frames, the background used and the user interface controls can be controlled to leverage browser settings, use high contrast, etc.

The author can also control how much interactivity will be exposed in the resulting output. For example, the user may elect to disable slide animations and transitions and opt for a more static presentation. Similarly, the author may elect to disable certain scripting features like the ability to resize dynamically the output to match the size of the browser window.

Somewhat related to this is the ability to specify the target screen size which is especially important when targeting the earlier browser generations or user environments where features like JavaScript are disabled.

For an internationalized implementation, there is the ability to control the encoding of text used in the generation of the HTML or MHTML output.

Finally there are a set of parameters that configure the on-disk storage of the resulting output. For example, if the customer knows something about the machine configurations of her audience, she can opt to use better raster graphic formats like PNG that support alpha transparency or elect to include Vector Markup Language ("VML") representations only for vector images.

The customer can also provide some indication as to how the output will be used. If the customer knows that the output will be used like regular files (perhaps passed around on CDs or moved between file shares) the user may elect to store the files in a folder to ensure that a straggling file is not lost; if, however, the target scenario is to put the files on a web server, the user can skip the folder and save the individual files in a flat directory. Similarly, if the customer knows that they are using a web server that only handles "8.3" file names, they can configure the implementation to generate files using names that are "8.3" compliant, as opposed to using long file names that may otherwise cause such web servers problems.

4.1.4.2 Print Options Properties

There is also a set of properties that control the default print behavior for a presentation. The inclusion of this information in the file format simply primes the Print dialog when this presentation is used. It does not force options nor does it represent the last-used set of print options for a presentation.

Using these properties, the author can control the type of output printed. For example, in some cases, authors need to print their slides (one slide per printed page) while in other cases, they want to provide printed handouts for the audience on which to take their own notes (handout pages that can contain anywhere from three to nine slides per printed page, as well as option lines for note taking). In other cases, the author would like to print out notes where each printed page has one slide (anchored at the top) and a text box (anchored at the bottom) with the speaker notes included or simply print the textual outline of the presentation.

The author can also control whether or not hidden slides are included in the printed output, as well as whether or not the output is sent to the printer in color, in grayscale, or in pure black and white.

There is also a set of properties that the author can set that determine if slides are framed on the printed page, if the slides are scaled up to the printed page (e.g., consider non 4x3 aspect ratio slides), etc.

4.1.4.3 Slide Show Properties

This set of presentation-level properties controls the default slide show.

Among the parameters that can be controlled is one that defines the type of slide show. Generally, the classic slide show is characterized by a presenter presenting the presentation to an audience. The presenter controls the flow of the presentation, etc. This is referred to as a "present" slide show. In some cases, however, the presentation is distributed and individuals walk themselves through the slide show. This is referred to as a "browse" slide show. Finally, there are cases where a slide show is prepackaged and used as a kiosk; naturally, it is referred to as a "kiosk" slide show.

Furthermore, the customer can control which slides are to be included in the slide show, what color the pen should be, etc.

Finally, the customer can control various interactivity settings that are to be used for the slide show. This provides the customer the ability to configure their slide show outside the typical settings for a particular slide show type. For example, the user may create a slide show that has a pre-configured animation built with timings (i.e., the time between particular builds or the time between slide transitions), even though she is going to be presenting the content to an audience.

4.1.4.4 View Properties

The schema pml-viewProperties.xsd defines the properties on all of the views found in the implementation.

MS's implementation currently supports the following views:

- Slide View
- Slide Master View
- Notes View
- Handout View
- Notes Master View
- Outline View
- Slide Sorter View

Additionally, the default view, Normal View, is a composite view that pulls from three multiple view property sets.

In general, there is a significant amount of commonality among views. For example, each view contains four common components:

Scale	The zoom scale for the view
Origin	The origin of the view
Variable Scaling	A special zoom scale that configures the application to fit the content of the view into whatever view size is provided
Draft Mode	Controls whether or not a view is in draft mode which is a mode designed to provide the fastest editing/redraw possible by dropping properties like font face, certain colors, pictures, etc.

For those views based on a slide (e.g., slide master view) there are additional common components:

Guide List	Represents the list of drawing guides for this view
Guide Properties	Represents guide properties like direction and position of each guide in the view

Guide List	Represents the list of drawing guides for this view
Guide Settings	Determines if guides should be shown for this view
Snap Settings	Determines if shapes should be snapped to the grid and/or snapped to other shapes for this view

4.2 Slides, Masters, Layouts, and Placeholders

4.2.1 Introduction

This subclause provides a high-level overview of the content described in the pml-slide.xsd schema.

The important aspects of the PresentationML Slides file format are introduced in the following order.

- Masters
- Presentation Slide
- Slide Notes
- Slide Layouts

This subclause provides a structured introduction to the slides portion of the PresentationML file format. Other subclauses build on this foundation and explain more about topics such as animation, comments, and the presentation object.

4.2.2 Masters

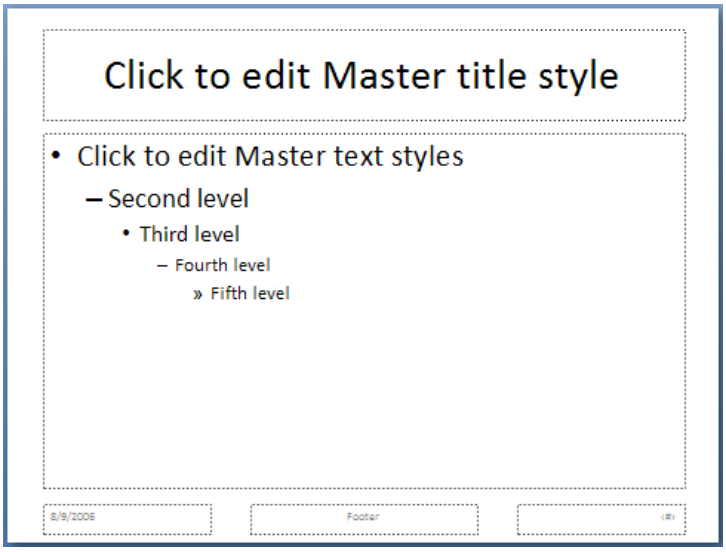
For slides the notion of hierarchy and inheritance applies. A master represents a common layout for the page type in question. For instance, if a slide master had a background set to a gradient fill then all slides referencing to that slide master would have the same background. In addition to setting common attributes of the slides such as background and styling information, the slide master also provides numerous layouts within itself in order to make a presentation that both follows a layout theme and incorporates a high level of variety. The variety is supported through slide layouts which will be discussed in a later section.

4.2.2.1 Slide Master

A *slide master* is a master that is tied specifically to presentation slides. The presentation slides are those that are shown during a presentation. These will be discussed in more detail in a later section on the Presentation Slide. Within a slide master are some common structural elements that should be understood, namely:

- *Common Data* - Common properties that will be inherited by the other slides as well as layout information for presentation slides based on the master slide.
- *Header and Footer* - Header and footer properties for the presentation slides to inherit.
- *Color Map* - Color Mapping for the presentation slides to inherit.
- *Text Styles* - Text Styling information to be used within each placeholder on a presentation slide.
- *Slide Layout List* - A list of slide layouts that provide the variety needed within any presentation. These are applied to a presentation slide which will inherit both the layout of the slide layout in addition to the slide design of the slide master.
- *Timing Information* - Common timing properties used for animation, controls, etc.

- *Transition Information* - Slide transitioning information to be inherited by each presentation slide.

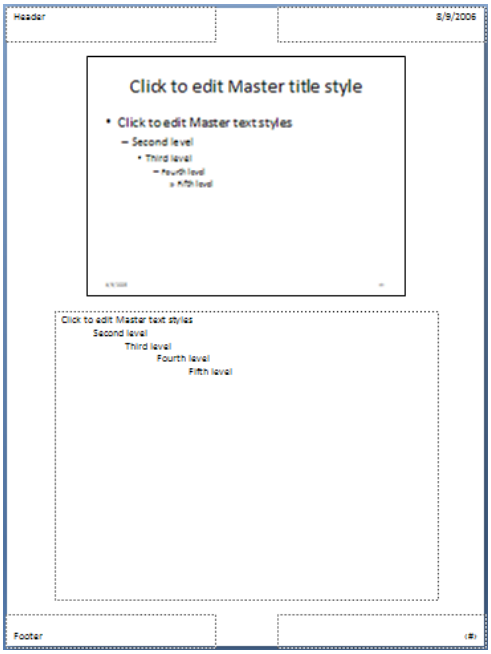


Slides inheriting information from a slide master do have the ability to specify properties that override those specified in the slide master.

4.2.2.2 Notes Master

A *notes master* is a master that specifies properties for slide notes pages. The notes page associated with a presentation slide stores a thumbnail of the presentation slide as well as the presenter's notes about the slide. These will be discussed in more detail in a later section. Within a notes master the important common structural elements are:

- *Common Data* - Common properties that will be inherited by other notes pages as well as the layout information for notes pages based on this master slide. The notes master serves as the pattern for all notes pages.
- *Color Map* - Color Mapping for the notes pages to inherit.

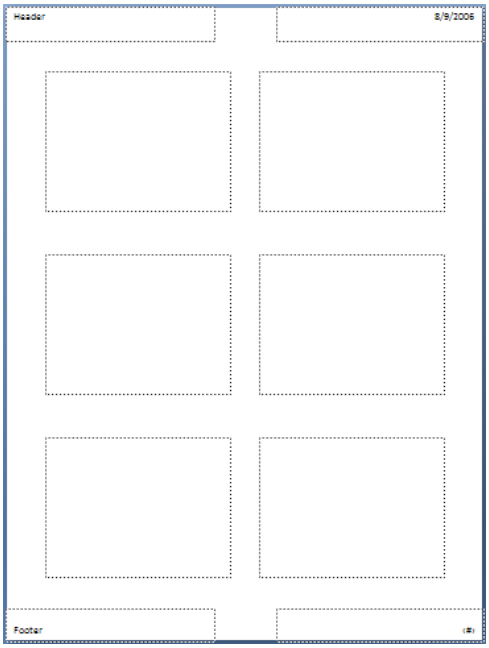


Notes pages inheriting information from a notes master do have the ability to specify properties that override those specified in the notes master.

4.2.2.3 Handout Master

A handout master determines the layout for all handout pages. The handout pages consist of a place to store a thumbnail of each slide with additional elements such as header, footer or graphical information. These will be discussed in more detail in a later section. Within a handout master are some common structural elements that should be understood, namely the following.

- *Common Slide Data* - Common properties and layout information that will be used by all handout pages. The handout master represents how each handout page will look.
- *Header and Footer* - Header and footer properties for all handout pages.
- *Color Map* - Color Mapping for all handout pages.



4.2.3 Presentation Slide

A presentation slide is a slide that inherits slide properties from the corresponding slide master and layout information from the corresponding slide layout. Each presentation slide has the ability to override any of this information that it chooses by specifying local attribute values within the presentation slide. Much like the master slide, the presentation slide contains some common structural elements, namely the following.

- *Common Slide Data* - Common properties and layout information for this presentation slide. Properties listed here that conflict with existing elements specified in the slide master will override those specified in the slide master.
- *Color Map Override* - Color Mapping that will override the inherited color mapping for this presentation slide.
- *Timing Information* - Common timing properties used for animation, controls, etc.
- *Transition Information* - Slide transitioning information for this presentation slide.

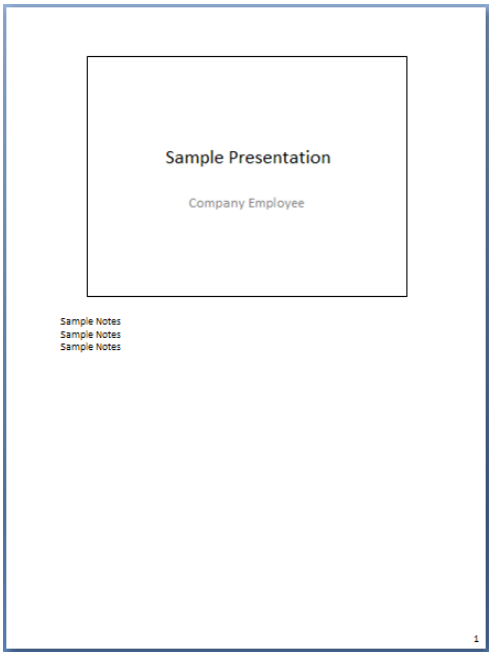
The above list defines the areas that can be used to override inherited components from the master slide and the layout slide. That is, these can be specifically defined on a per-slide basis via the above elements.

4.2.4 Notes Page

A notes page inherits slide properties from the corresponding notes master. The initial layout for a notes page is defined by the single notes master slide. Each notes page has the ability to override any of this information that it chooses by specifying local attribute values within the notes slide. Much like the notes master, the notes page contains some common structural elements, namely the following.

- *Common Slide Data* - Common properties and layout information for this notes page.
- *Color Map Override* - Color Mapping to override the inherited color mapping for this notes page.

The above list defines the areas that can be used to override inherited components from the notes master. That is, these can be specifically defined on a per-slide basis via the above elements.

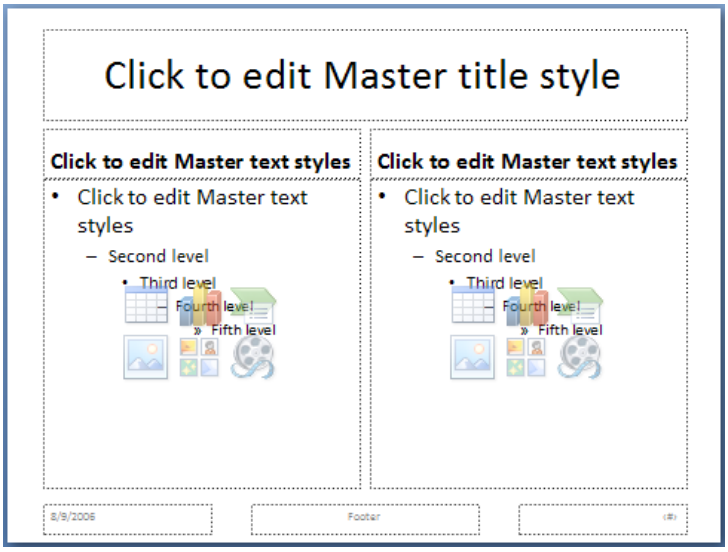


4.2.5 Slide Layouts

A *slide layout* inherits slide properties from the corresponding slide master and sets layout information for all presentation slides that utilize this layout. Each presentation slide has the ability to override any of this information that it chooses by specifying local attribute values within the presentation slide. Much like the slide master, the slide layout contains some common structural elements:

- *Common Slide Data* - Common properties and layout information that will override properties set within the slide master but will be inherited by all presentation slides that utilize this layout.
- *Color Map Override* - Color Mapping that will override the inherited color mapping from the slide master but will be inherited by all presentation slides that utilize this layout.
- *Header and Footer* - Header and footer properties that will override properties set within the slide master but will be inherited by all presentation slides that utilize this layout.
- *Timing Information* - Common timing properties used for animation, controls, etc. These will override properties set within the slide master but will be inherited by all presentation slides that utilize this layout.
- *Transition Information* - Slide transitioning information to be inherited by each presentation slide. These will override properties set within the master slide but will be inherited by all presentation slides that utilize this layout

The above list defines the areas that can be used to override inherited components from the master slide. That is, these can be specifically defined on a per-layout basis via the above elements.



4.3 Comments

4.3.1 Introduction

This document describes the commenting feature for presentations as expressed in PresentationML. The schema that defines this feature is pml-comments.xsd.

Note that it is important to keep in mind that comments are not shapes. The representation of them within the document is left entirely up to the generating application and are thus implementation specific.

4.3.2 Functional Overview

Readers of a presentation can provide feedback to the presentation author in the form of *comments*. Comments can only be applied to slides; they cannot be applied to masters of any type or to notes slides.

At first glance, comments appear to be shapes on the slide surface; however, they are not. Comments differ from regular shapes in two ways:

- Comments cannot be formatted or resized
- The text contained within a comment cannot be formatted

4.3.3 Comment Author List

Presentations contain a list of all authors who have comments in the presentation. This list is commonly referred to as the *Comment Author List* (CAL). The CAL contains one entry for each author. Each entry is made up of five pieces of data: ID, Author Name, Author Initials, Last Index, and Color Index.

Each author that comments on a presentation is assigned an ID, which is a simple integer. This ID is unique within the presentation, and is assigned by the application itself.

The Author Name and Author Initials are taken from the application itself. If no initials are known to the application, the comment author is prompted upon the insertion of the initial comment. Both the Author Name and Author Initials are simple strings; that is, there is no association of the values with an identity (from a security or authentication perspective).

The Last Index (lastIdx) is an integer that documents how many comments the associated author has made in this presentation. When the author makes another comment, that comment is numbered using the next integer, and then this value is updated once again.

The Color Index (clrIdx) is an integer into a color table that is used to provide the solid background fill for the comment shape. The utility that this provides is that all of the comments by a particular author share the same color.

Here is an example of such a CAL:

```
<p:cmAuthorLst>
  <p:cmAuthor id="0" name="Shawn" initials="SV" lastIdx="3" clrIdx="0" />
  <p:cmAuthor id="1" name="Brian" initials="BJ" lastIdx="7" clrIdx="1" />
</p:cmAuthorLst>
```

To determine if an author is already in the CAL, one must consider only the Author Name and Author Initials data. If they both match an entry in the CAL, the author is already considered to be in the CAL; otherwise, the author is considered unique, and a separate entry is added for that author in the CAL.

When the presentation is saved using PresentationML, a separate Comment Authors part is created that contains the CAL.

4.3.4 Comment List

Each slide within a presentation may contain zero or more comments. Each slide with at least one comment starts a list of comments for that slide. Each entry in that list is made up of the following pieces of data:

- Author ID: This represents the ID of the author who created the comment. It matches an entry in the CAL.
- Date/Time: This represents the date and time of the last modification of this particular comment. Although expressed in UTC, its accuracy is dependent on the state of the machine making the edits.
- Index: This is the number assigned to this particular comment, and is one of the comments associated with the specified author. This number should be equal to, or less than, the Last Index value for the author in the CAL. There cannot be duplicate Indexes for the same author.
- Position: This defines the 2D coordinate for the location at which the comment shows up on the slide surface. This is the position of the upper left point of the comment shape.
- The Text data includes all of the text that makes up the body of the comment. Note that this text is expressed differently than other text as expressed in DrawingML. As this text contains no formatting, and is strictly limited to text input, there is no additional data that needs to be stored.

Here is an example of a comment list for a slide:

```
<p:cmLst>
  <p:cm authorId="0" dt="2006-01-30T22:45:13.597" idx="3">
    <p:pos x="10" y="10" />
    <p:text>Need to check with Mary on exact data values</p:text>
  </p:cm>
  <p:cm authorId="1" dt="2006-01-30T22:46:22.082" idx="1">
    <p:pos x="106" y="106" />
    <p:text>This chart is hard to read from afar</p:text>
  </p:cm>
</p:cmLst>
```

When the presentation is saved using PresentationML, a separate Comments part is created for each comment list.

4.4 Animation

4.4.1 Introduction

This subclause provides a high-level overview of the animation settings in PresentationML. This schema is loosely based on the syntax and concepts from the Synchronized Multimedia Integration Language (SMIL), a W3C Recommendation for describing multimedia presentations using XML.

The schema describes all the animations effects on that reside on a slide; it also describes the animation that occurs when going from slide to slide (slide transition).

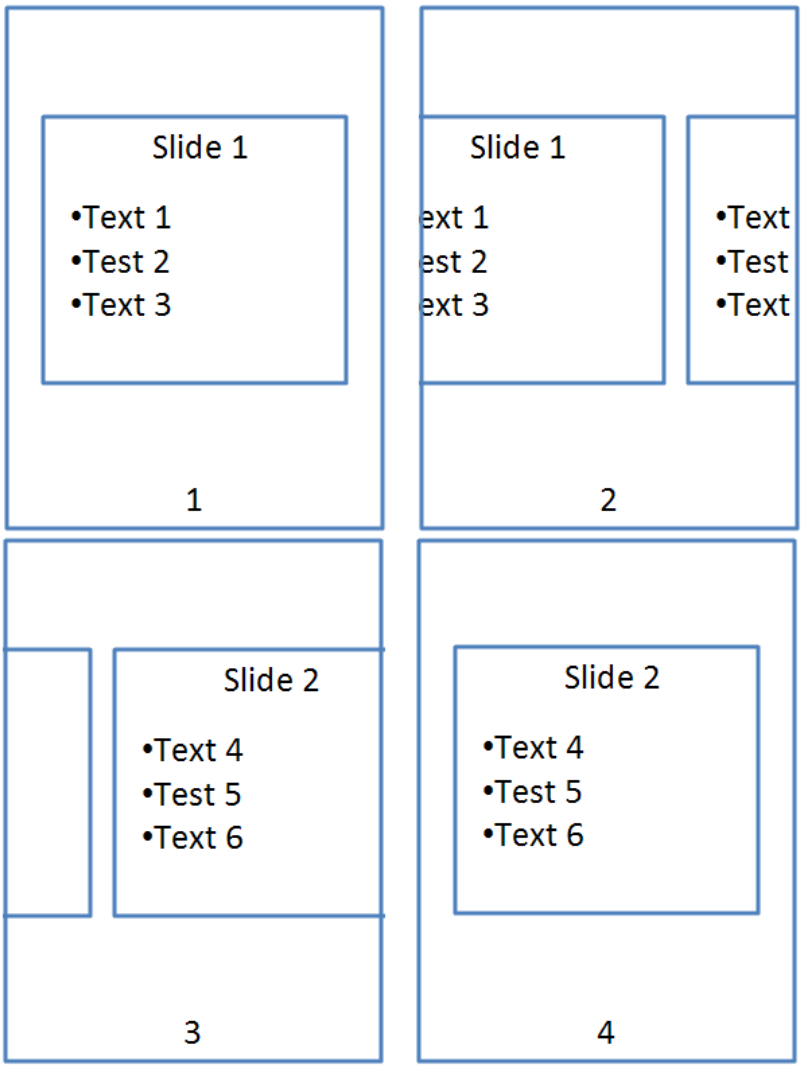
Animations on a slide are inherently time-based and consist of an animation effects on an object or text. However, slide transitions do not follow this concept and always appear before any animation on a slide.

All elements described in this schema are contained within the slide XML file. Superficially, they are in the transition and the timing element as shown below:

```
<p:sld>
  <p:cSld> …
  <p:clrMapOvr> …
  <p:transition> …
  <p:timing> …
</p:sld>
```

4.4.2 Slide Transitions

Slide transitions are the animation effects that displayed in between slides. They are specified in the transition element in the slide XML file. For example, consider a slide with a "push" slide transition as shown below:



The push element should be used as follows:

```
<p:transition>
  <p:push dir="r"/>
</p:transition>
```

4.4.3 Timeline Overview

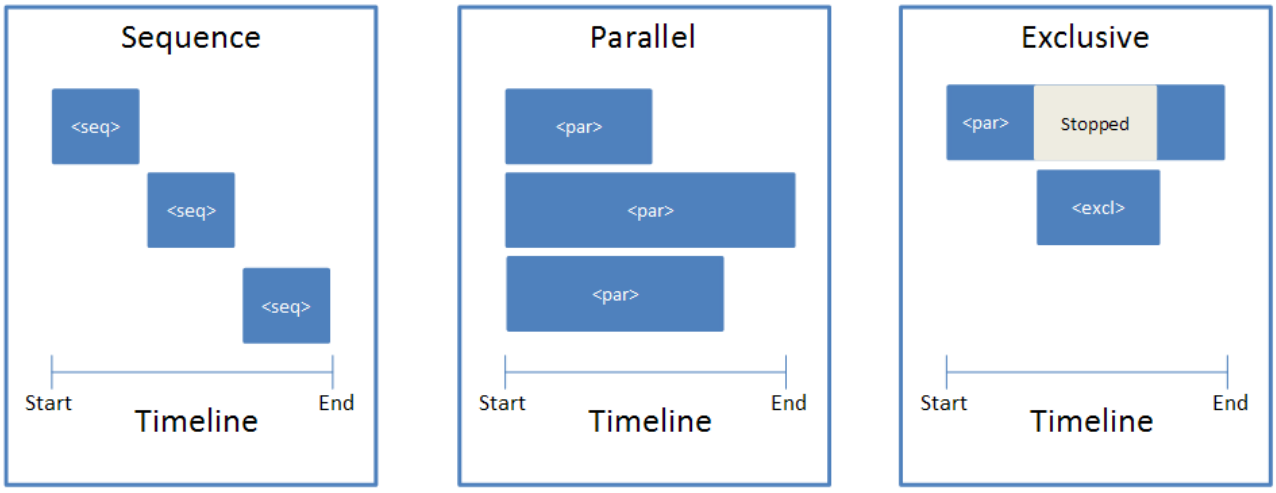
The timeline is an important aspect for animations on a slide. It moderates the amount of time that the animations are run from beginning to end. For example, it allows animation to be started when the slide is loaded or based on an event.

A timeline is composed of timing nodes that dictate at which point a certain animation is shown. A timeline can contain unlimited number of timing nodes; it can also have time nodes nested within them.

There are three types of time nodes:

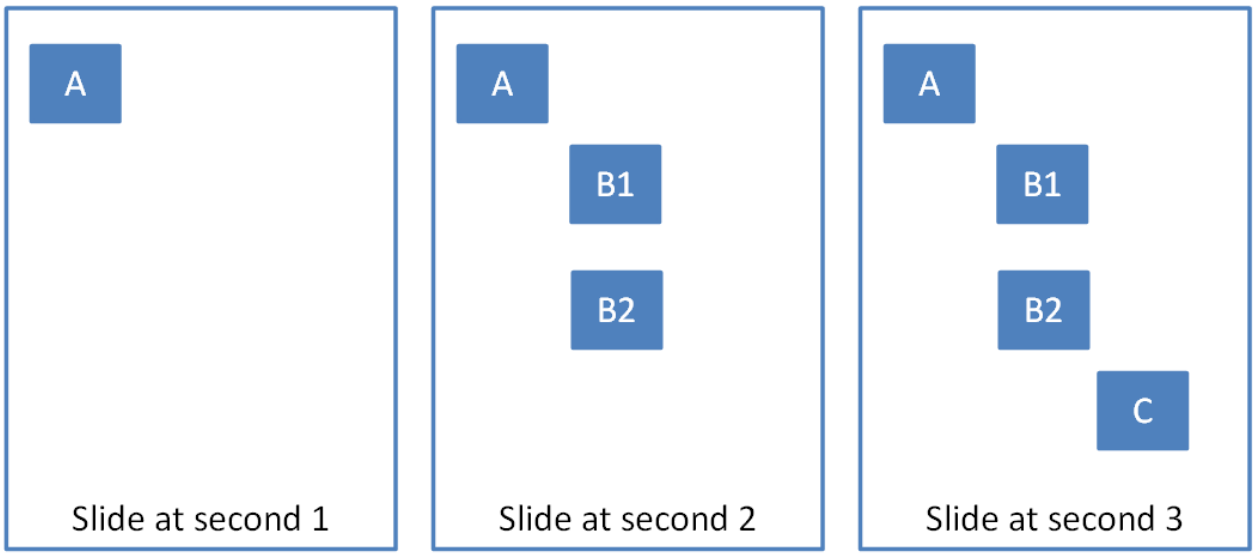
Element	Name	Description
par	Parallel	This is a parallel time node and can be activated along with other parallel time node containers.
seq	Sequence	This is a sequence time node and it can only be activated when the one before it finishes.
excl	Exclusive	This time node is used to pause all other timelines when it is activated.

A conceptual diagram of this is shown below:

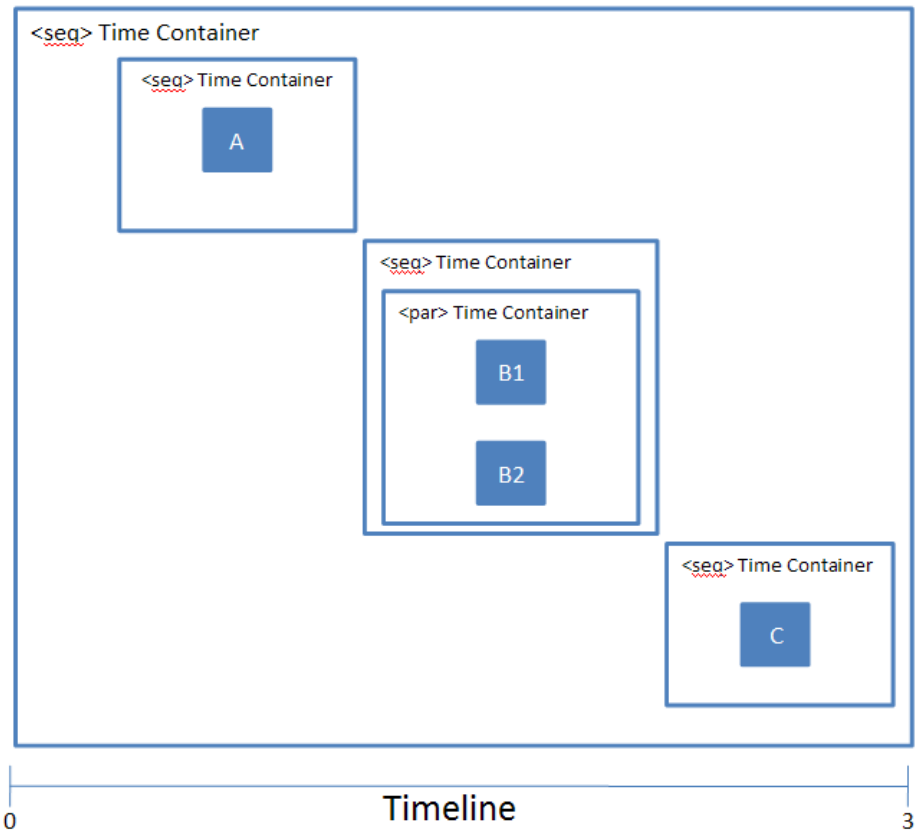


4.4.4 Timeline Construction

To illustrate what the timeline looks like in the slide XML file, suppose we have four rectangles named A, B1, B2, and C that appear on a timeline three seconds long. Rectangle A appears at second 1, B1 and B2 appear together at second 2, and C appears at second 3, as shown below:



The timeline and time containers could look something like:



A typical timeline consists of the following structure:

```
<p:timing>
  <p:tnLst>
    <p:seq concurrent="1" nextAc="seek">
      <p:stCondLst> …
      <p:cTn id="2" dur="indefinite" nodeType="mainSeq">
        <p:childTnLst>
          <p:seq> …   // Square A
          <p:seq>
            <par>….   // Square B1
            <par>….   // Square B2
          </p:seq>
          <p:seq> …   // Square C
        </p:childTnLst>
      </p:cTn>
      <p:prevCondLst> …
      <p:nextCondLst> …
    </seq>
  </p:tnLst>
  <p:bldLst> …
</p:timing>
```

As show, this timeline starts with a timing element that represents the timeline. Within this timeline, there is a child element tnList, which contains a list of time nodes.

In this case, there is one main timing container, which is the seq element. Within this element there are a three of conditional elements, namely stCondList, nextCondList, and prevCondList. These elements contain condition properties that allow for the starting/stopping of the particular time node. This is explained in more detail in §4.4.6.

Following the stCondList element is the cTn element, which describes the properties for this node. Within this element is the childTnList, which contains the nested time nodes that describe the animation sequence mentioned above.

Finally, we have the bldList element, which is used to specify how objects with sub-shapes should be animated. More information can be found in §4.4.7.

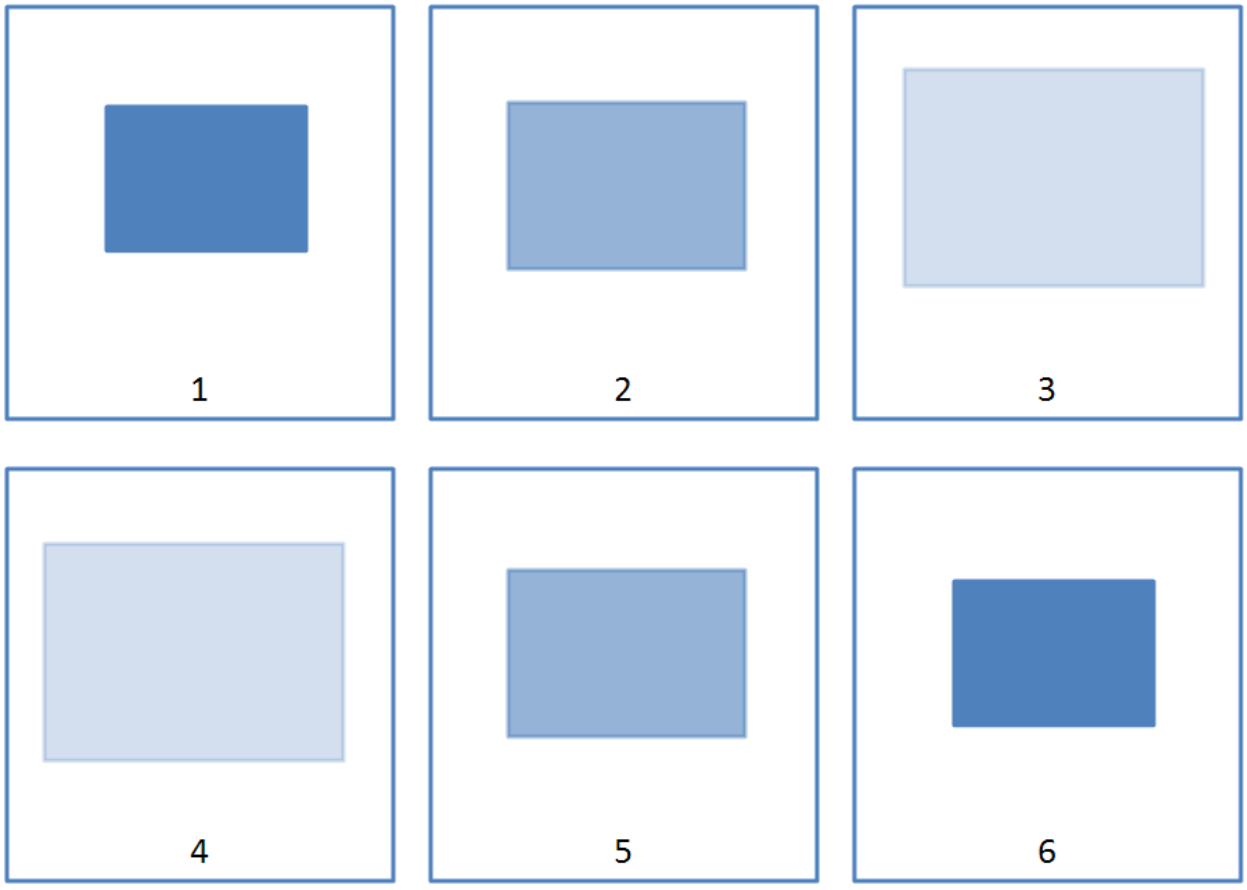
4.4.5 Animation Behaviors

All animation consists of the following basic animation behaviors:

Element	Name	Description
anim	Animate	The animate behavior introduces a generic attribute animation that requires no semantic understanding of the attribute being animated. It can animate numbers.
animColor	Animate Color	This behavior animates the color of a particular element.

Element	Name	Description
animEffect	Animate Effect	This behavior provides the ability to do image transform/filter effects on elements.
animMotion	Animate Motion	Animate motion provides an abstracted way to move positioned elements. It provides the ability to specify from/to/by type motion as well as to use more detailed path descriptions for motion over polylines or bezier curves.
animRotation	Animate Rotation	This behavior allows rotation of an element.
animScale	Animate Scale	Allows animation of the width and/or height of an element over time.

A time node can combine multiple animations for a range of effects. For example, the "flash bulb" animation which scales a shape larger while at the same time having it fade uses two animation behavior elements. An example is shown below:



The representation for this animation effect in the time node element appears like:

```
<p:par>
  <p:cTn id="5">
    <p:stCondLst>….
    <p:childTnLst>
      <p:animEffect transition="out" filter="fade"> …
      <p:animScale>
        <p:cBhvr>
          <p:cTn id="7" dur="500" autoRev="1" fill="hold"/>
          <p:tgtEl>
            <p:spTgt spid="9"/>
          </p:tgtEl>
        </p:cBhvr>
        <p:by x="105000" y="105000"/>
      </p:animScale>
    </p:childTnLst>
  </p:cTn>
</p:par>
```

In this time node, we have two animation effects. One is creating a "fade" effect on the shape using the animEffect element and the other is creating a "scale" effect using the animScale element. All animation behaviors have a cBhvr and cTn element, which stores properties for the animation. For example, we can give the animation behaviors an ID and attributes that set the duration of the animation. The spTgt specifies the target shape to which this animation effect will be applied.

4.4.6 Conditional Properties

Another important aspect of time nodes is conditional properties. There are four such conditions:

Element	Name	Description
stCondLst	Start Condition	Conditions that must be met for a time node to start.
prevCondLst	Previous Condition	Conditions that must be met for a time node to go back to the previous time node.
nextCondLst	Next Conditions	Conditions that must be met for a time node to advance to the next time node.
endCondLst	End Conditions	Conditions that must be met for a time node to end.

Conditional properties are useful for providing finer granularity as to exactly when a time node should be activated. For example, suppose we have a shape with an entrance appearance after five seconds. The stCondLst element should be used as follows:

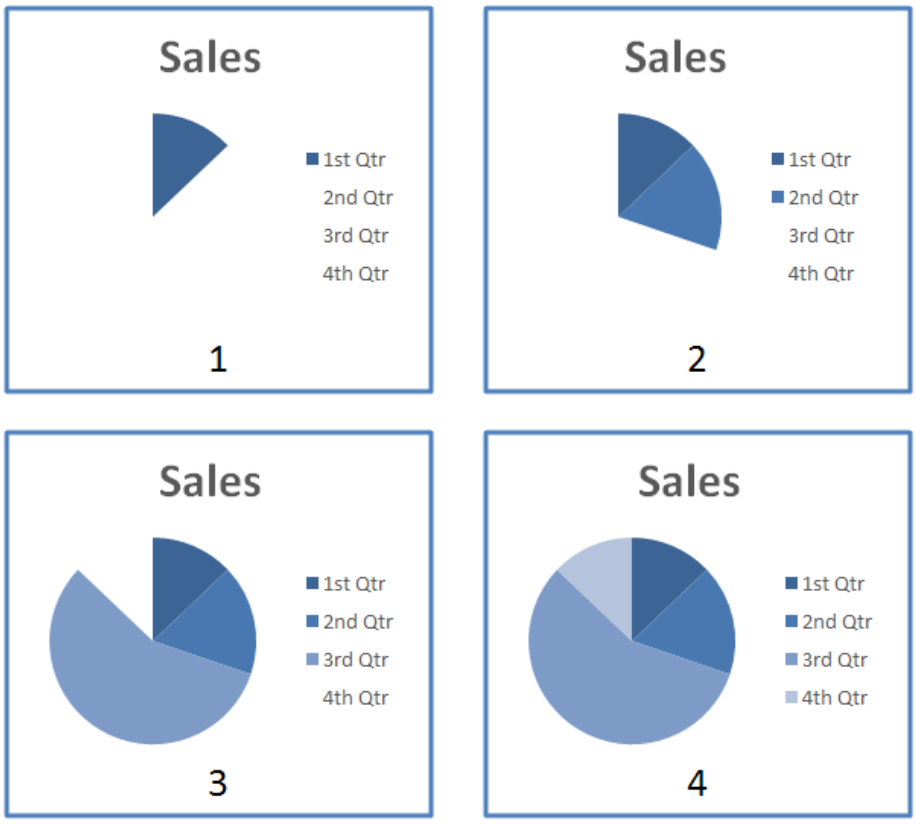
```
<p:par>
  <p:cTn id="5">
    <p:stCondLst>
      <p:cond delay="5000"/>
    </p:stCondLst>
  </p:cTn>
</p:par>
```

4.4.7 Build Animations

Another important aspect of animations is how they are built. This refers to how the different sub-shapes or sub-components of an object are displayed. The different objects that can have build properties are text, diagrams, and charts.

This is specified in the bldLst element.

For example, suppose we want to animate a pie chart, but based on category as shown below:



The representation of this in the slide XML looks like:

```
<p:bldLst>
  <p:bldGraphic spid="4" grpId="0">
    <p:bldSub>
      <a:bldChart bld="category"/>
    </p:bldSub>
  </p:bldGraphic>
</p:bldLst>
```

The bldLst element contains children elements that describe how the different objects should be built. In this case, there is only one graphic to be build, that with id 4. The bldGraphic element contains the bldSub element, which describes how the object should be built. This element then contains the bldChart element with the attribute bld set to `category`.

End of informative text.

4.5 Slide Synchronization

This subclause provides an overview of the p:SldSyncPr element in pml-slideSynchronizationData.xsd.

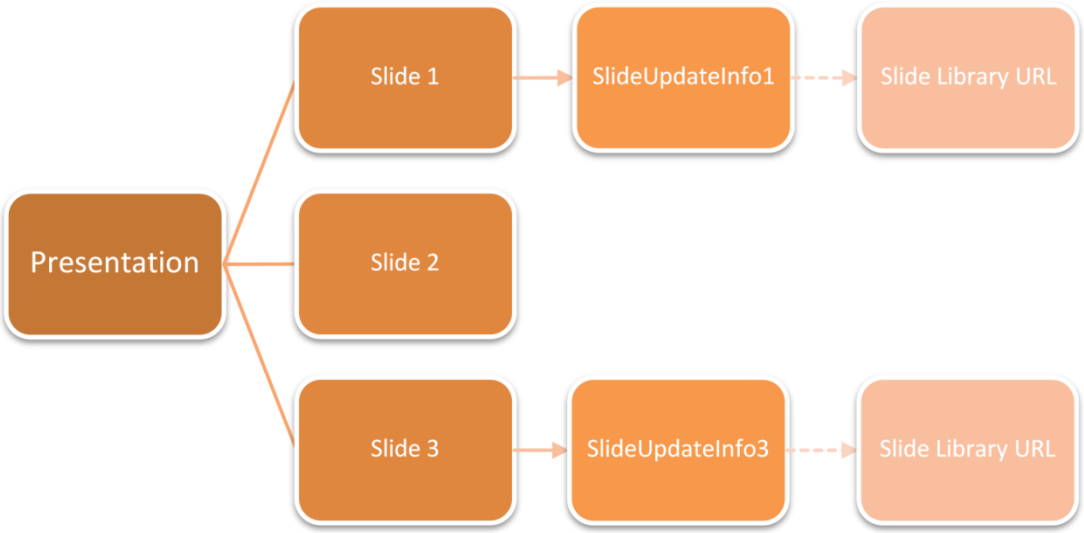
4.5.1 Introduction

A *Slide Library* is a library type in SharePoint Server that exclusively contains single-slide presentations. Users are able to publish and reuse slides to/from these libraries. Furthermore, when a user inserts a slide from the library into a presentation, she is able to create an update relationship so that she is notified when the original slide on the server changes.

It is worth noting that the SlideUpdateInfo part in itself does not define the complete slide update functionality. That part requires a SharePoint Slide Library or compatible server (e.g., a webdav server that emulates SharePoint SOAP methods).

4.5.2 Slide Update Info

For each slide in a presentation that has an update relationship with its counterpart in a Slide Library, a Slide Update Info part is created. The diagram below provides an overview of this relationship.



Each Slide Update Info part is stored under its own folder. For example:

```
/ppt/slideUpdateInfo/slideUpdateInfo1.xml
/ppt/slideUpdateInfo/slideUpdateInfo3.xml
```

The part is indentified for each slide by a relationship with the following characteristics:

```
Type: http://…/slideUpdateInfo
TargetMode: Internal
Target= "<Uri of the slideupdateinfo part for the slide>"
```

The content type of the update info part is application/vnd.openxmlformats-officedocument.presentationml.slideUpdateInfo+xml.

It contains:

- Modified time of the slide on the server when it was inserted (stored in ISO 8061 format).
- Time the slide was inserted into the presentation.
- Regular ID of the slide on the server (saved as a string)

These Slide Update Info parts themselves have an external relationship to the Slide Library Url from which the Slide was inserted.

```
Type: http://.../slidelibraryUrl
TargetMode: External
Target = “<Url of the Slide Library>”
```

Every Slide Update Info part should have exactly one occurrence of this relationship.

Samples:

slideupdateinfo1.xml

```
<p:sldUpdatePr … serverSldId="7991" serverSldModifiedTime="2006-03-08T18:48:33"
  clientInsertedTime="2006-03-10T06:02:33.975" />
```

slideupdateinfo1.xml.rels

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
  <Relationships xmlns="http://.../relationships">
  <Relationship Id="rId1" Type=http://.../slideUpdateUrl
    Target="http://content/slides" TargetMode="External" />
</Relationships>
```

End of informative text.

5. Introduction to DrawingML

This clause is informative.

This clause contains a detailed introduction to the components of DrawingML.

5.1 Basics

5.1.1 Introduction

This subclause provides a high-level overview of the content described in the dml-baseTypes.xsd, dml-compatibility.xsd and dml-lockedCanvas.xsd schemas. The aggregation of the elements within these schemas encompass what has been labeled the DrawingML – Basics sub-clause. That is, the elements contained here are considered to be commonly shared elements among the DrawingML framework.

5.1.2 Overview

This sub-clause is made up of four distinct pieces: Basic Elements, Colors, Compatibility, and Locked Canvas. Together these make up the common elements that are shared across the DrawingML framework. These elements are described below.

5.1.3 Basic Elements

When the common elements of the DrawingML framework are aggregated it can be seen that the most widely used elements are property elements. These reside within every object and allow for the setting of both visual and non-visual object-specific properties. The visual properties are those that affect the appearance of the object when it is rendered on the screen. The non-visual properties on the other hand, do not affect the object's appearance. Instead, these properties are used to store information normally hidden such as identification numbers, human readable names for the objects and specific behaviour that should be obeyed within the UI when manipulating the corresponding object.

5.1.4 Colors

The notion of color plays a vital role in the presentation of DrawingML objects within a document. Virtually all objects are specified to have a corresponding color or set of colors. The notion of color is a common one among graphically-inclined applications. Because of this and the fact that there are many different types of graphic objects available today, we introduce the notion of several different types of color models. The following color models can be used to specify color within a DrawingML based document.

1. RGB – Red, Green, Blue Color Model
2. HSL – Hue, Saturation, Luminance Color Model
3. Scheme – Scheme Based Color Model
4. Preset – Color Presets Color Model

5. System – Operating System Color Model

These different models allow document authors the choice as to which color model would be appropriate for their particular application. Each of these is detailed within the DrawingML Basics reference material.

5.1.5 Compatibility

Compatibility deals with the notion of legacy drawings. Legacy drawings are objects that were supported by previous versions of a generating application, but are no longer provided as an option. In order to store these drawing objects correctly, we introduce the notion of legacy drawing compatibility. This allows for the specification of information used to identify this legacy object and thus allow for full rendering support within current versions of the generating application.

5.1.6 Locked Canvas

Locked Canvas is a minor topic that is similar to compatibility in that it is used to render drawing objects that would otherwise not be recognized due to a lack of information. Locked Canvas, however, goes in the opposite direction from compatibility, and deals with objects that have been created and saved in the current version of a generating application and are being opened in a previous version of the generating application. The locked canvas element acts as a container for more advanced drawing objects. The notion of a locked canvas comes from the fact that the generating application opening the file cannot create this object and thus cannot perform edits either. Thus, the drawing object is locked from all UI adjustments that would normally take place.

5.2 Audio and Video

5.2.1 Introduction

DrawingML contains support for basic audio and video capabilities. The definitions for these structures can be found in dml-audioVideo.xsd.

5.2.2 Functional Overview

Presentation authors can specify that both audio and video can play while a slide is shown in slide show. When such media is inserted into a presentation, a presentation author can specify that the media is to play automatically (that is, in accordance with the slide's animation timeline) or in response to a mouse-click. In either case, media only plays for the duration of time specified, or until the slide changes, whichever is shorter.

Sources for audio content include CD-based files as well as the more traditional disc- or server-based files.

When inserting audio content stored on a CD, the author can specify a start and end track, as well as an index into each track. This information identifies the content from the CD to be played for the specified slide during a slide show.

In cases where the audio or video content is stored on a hard drive or server, the author can only specify the file itself; it will be played in its entirety, or until the slide changes.

5.2.3 DrawingML Syntax

In all three cases, the media objects themselves are stored on the slides using a picture shape. The picture shape uses a blipFill to show the media object on the slide's surface. In both audio cases, the picture used is the icon image, whereas in the video case, the picture used is the poster frame for the video file.

To express this information, the standard blipFill element is used to refer to the image file; and because this refers to a file within the package, a relationship ID is used:

```
<p:pic>
  <p:nvPicPr> …</p:nvPicPr>
  <p:blipFill>
    <a:blip r:embed="rId4" r:link="" />
    <a:stretch>
      <a:fillRect />
    </a:stretch>
  </p:blipFill>
  <p:spPr> … </p:spPr>
</p:pic>
```

As the media objects are related to the slide's timeline—in both the automatic and mouse-click cases—they must have interactivity information stored in the form of a hyperlink.

To express this information, a hyperlink is added to the non-visual shape properties; and because it is a hyperlink, it, too, uses a relationship ID:

```
<p:pic>
  <p:nvPicPr>
    <p:cNvPr id="15" name="Rectangle 15" descr="">
      <a:hlinkClick r:id="rId3" tgtFrame="" tooltip="" />
    </p:cNvPr>
  <p:cNvPicPr />
    <p:nvPr> … </p:nvPr>
  </p:nvPicPr>
  <p:blipFill> … </p:blipFill>
  <p:spPr> … </p:spPr>
</p:pic>
```

The final piece of information required for each media type is the source bits. In both file-based media cases, DrawingML needs to provide a mechanism to specify the location and file name for the media; this is in contrast to the CD-based audio where only track information is required. Regardless of the type of source information required, all of this is stored in application-specific non-visual properties. This is illustrated in the following three XML islands representing each of the three media cases:

CD-Audio	<p:pic> <p:nvPicPr> … <p:cNvPicPr /> <p:nvPr> <a:audioCd> <a:st track="2" time="50" /> <a:end track="3" time="22" /> </a:audioCd> </p:nvPr> … </p:pic>
File-Audio	<p:pic> <p:nvPicPr> … <p:cNvPicPr /> <p:nvPr> <a:audioFile r:embed="" r:link="rId1" /> </p:nvPr> </p:nvPicPr> … </p:pic>
File-Video	<p:pic> <p:nvPicPr> … <p:cNvPicPr /> <p:nvPr> <a:videoFile r:embed="" r:link="rId1" /> </p:nvPr> </p:nvPicPr> … </p:pic>

In the CD-Audio case, there is no capability to choose the particular CD drive that will contain the source. This is a functional limitation.

While the default case is that media is linked, file-based media can also have the source bits included in the package as a separate part. In this case, the relationships point not to an external file but, rather, to a part inside the package.

5.3 Styles

5.3.1 Introduction

This piece of DrawingML deals with the definition of the shared aspects contained within a document theme. The shared-style sheet defines an application-independent set of styling that can be applied to objects within a document and which affects the look of the document and the information and objects it contains. For example, in a presentation, shapes can have a certain look, whereas in an e-mail, all of the text can have certain properties, and headings are styled.

A second topic is the definition of a table style as used within DrawingML. A table style defines the look of a table regardless of the data present in that table.

5.3.2 Shared Style Sheet

The shared-style sheet within DrawingML is responsible for containing different formatting options and style options that can be used within a given document.

5.3.2.1 Theme

The theme is the root-level complex type associated with a shared-style sheet. This complex type holds all of the different formatting options available to a theme, and defines the overall look and feel of a document when themed objects are used within the document. In figure 1 below, we can see an example of two different themes applied to the same slide in a presentation.

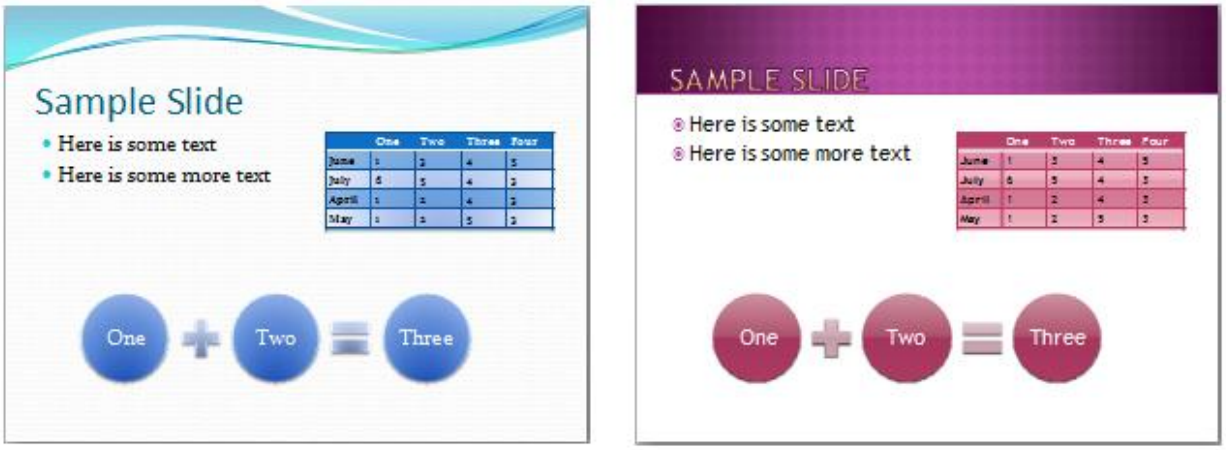


Figure 1: A theme applied to the same slide in a presentation. Not only does the font and colors change, but also the effects applied to the shapes and table.

A theme consists of four main parts, although the themeElements element is the piece that holds the main formatting defined within the theme. The other parts provide overrides, defaults, and additions to the information contained in themeElements. The complex type defining a theme, CT_OfficeStyleSheet, is defined in the following manner:

```
<complexType name="CT_OfficeStyleSheet">
  <sequence>
    <element name="themeElements" type="CT_BaseStyles" minOccurs="1"
      maxOccurs="1"/>
    <element name="objectDefaults" type="CT_ObjectStyleDefaults"
      minOccurs="0" maxOccurs="1"/>
    <element name="extraClrSchemeLst" type="CT_ColorSchemeList"
      minOccurs="0" maxOccurs="1"/>
    <element name="custClrLst" type="CT_CustomColorList" minOccurs="0"
      maxOccurs="1"/>
```

```
      <element name="extLst" type="CT_OfficeArtExtensionList"
        minOccurs="0" maxOccurs="1"/>
    </sequence>
    <attribute name="name" type="xsd:string" use="optional" default=""/>
  </complexType>
```

This complex type also holds a CT_OfficeArtExtensionList, which is used for future extensibility of this complex type.

5.3.2.2 Theme Elements

The complex type CT_BaseStyles defines the theme elements for a theme, and is the workhorse of the theme. The bulk of the shared theme information that is used by a given document is defined here. Within this complex type is defined a color scheme, a font scheme, and a style matrix (format scheme) that defines different formatting options for different pieces of a document. The complex type CT_BaseStyles is defined in the following manner:

```
  <complexType name="CT_BaseStyles">
    <sequence>
      <element name="clrScheme" type="CT_ColorScheme" minOccurs="1"
        maxOccurs="1"/>
      <element name="fontScheme" type="CT_FontScheme" minOccurs="1"
        maxOccurs="1"/>
      <element name="fmtScheme" type="CT_StyleMatrix" minOccurs="1"
        maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList"
        minOccurs="0" maxOccurs="1"/>
    </sequence>
  </complexType>
```

5.3.2.3 Color Scheme

The complex type CT_ColorScheme defines a set of colors for the theme. The set of colors consists of twelve color slots that can each hold a color of choice. The colors are organized in the following way:

- Dark 1 (dk1) – This represents a dark color, usually defined as a system text color
- Light 1 (lt1) – This represents a light color, usually defined as the system window color
- Dark 2 (dk2) – This represents a second dark color for use
- Light 2 (lt2) – This represents a second light color for use
- Accents 1 through 6 (accent1 through accent6) – These are six colors which can be used as accent colors in the theme
- Hyperlink (hlink) – The color of hyperlinks
- Followed Hyperlink (folHlink) – The color of a followed hyperlink

These colors define the theme colors that objects can utilize within a document. When an object uses a theme color, the color of the object can change when the theme is changed, but will always map to accent 1 if that

were the theme color used by the object. An example of theme colors defined and used can be seen in figure 2.

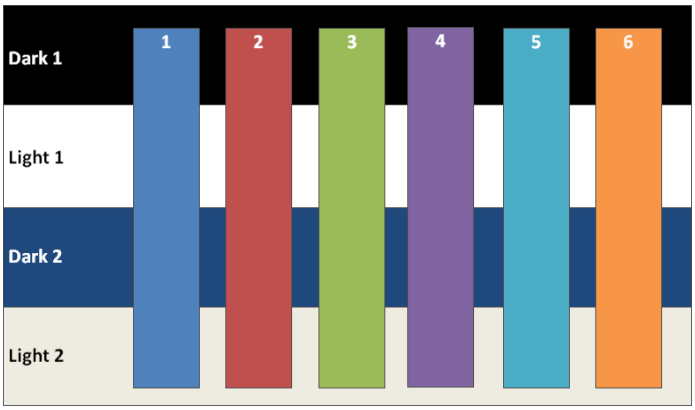


Figure 2: Sample colors defined and used for dark1/2, light1/2, and the six accent colors.

The complex type CT_ColorScheme is defined in the following manner:

```
<complexType name="CT_ColorScheme">
  <sequence>
    <element name="dk1" type="CT_Color" minOccurs="1" maxOccurs="1"/>
    <element name="lt1" type="CT_Color" minOccurs="1" maxOccurs="1"/>
    <element name="dk2" type="CT_Color" minOccurs="1" maxOccurs="1"/>
    <element name="lt2" type="CT_Color" minOccurs="1" maxOccurs="1"/>
    <element name="accent1" type="CT_Color" minOccurs="1"
      maxOccurs="1"/>
    <element name="accent2" type="CT_Color" minOccurs="1"
      maxOccurs="1"/>
    <element name="accent3" type="CT_Color" minOccurs="1"
      maxOccurs="1"/>
    <element name="accent4" type="CT_Color" minOccurs="1"
      maxOccurs="1"/>
    <element name="accent5" type="CT_Color" minOccurs="1"
      maxOccurs="1"/>
    <element name="accent6" type="CT_Color" minOccurs="1"
      maxOccurs="1"/>
    <element name="hlink" type="CT_Color" minOccurs="1" maxOccurs="1"/>
    <element name="folHlink" type="CT_Color" minOccurs="1"
      maxOccurs="1"/>
```

```
      <element name="extLst" type="CT_OfficeArtExtensionList"
        minOccurs="0" maxOccurs="1"/>
    </sequence>
    <attribute name="name" type="xsd:string" use="required"/>
  </complexType>
```

5.3.2.4 Font Scheme

The complex type CT_FontScheme defines a font pair. The pair consists of a major font and a minor font. An example of use would be the major font used in headings for a document and the minor font used for the paragraph parts of a document. The major and minor fonts are defined through a collection of font faces defined on a per-language basis. For example, one may define only a Latin-based font, or one can define many different fonts for different locals for a major or minor font. The font used in the document depends on the user's language.

The complex type CT_FontScheme is defined in the following manner:

```
  <complexType name="CT_FontScheme">
    <sequence>
      <element name="majorFont" type="CT_FontCollection" minOccurs="1"
        maxOccurs="1"/>
      <element name="minorFont" type="CT_FontCollection" minOccurs="1"
        maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList"
        minOccurs="0" maxOccurs="1"/>
    </sequence>
    <attribute name="name" type="xsd:string" use="required"/>
  </complexType>
```

5.3.2.5 Major and Minor Font (Font Collection)

The complex type CT_FontCollection defines a major and minor font which is used in the font scheme. A font collection consists of a font definition for Latin, East Asian, and complex script. On top of these three definitions, one may also define a font for use in a specific language or languages.

The complex type CT_FontCollection is defined in the following manner:

```
  <complexType name="CT_FontCollection">
    <sequence>
      <element name="latin" type="CT_TextFont" minOccurs="1"
        maxOccurs="1"/>
      <element name="ea" type="CT_TextFont" minOccurs="1" maxOccurs="1"/>
      <element name="cs" type="CT_TextFont" minOccurs="1" maxOccurs="1"/>
      <element name="font" type="CT_SupplementalFont" minOccurs="0"
        maxOccurs="unbounded"/>
```

```
      <element name="extLst" type="CT_OfficeArtExtensionList"
        minOccurs="0" maxOccurs="1"/>
    </sequence>
  </complexType>
```

5.3.2.6 Supplemental Font

The complex type CT_SupplementalFont defines an additional font that is used for language specific fonts in themes. For example, one can specify a font that gets used only within the Japanese language context.

The complex type CT_SupplementalFont is defined in the following manner:

```
  <complexType name="CT_SupplementalFont">
    <attribute name="script" type="xsd:string" use="required"/>
    <attribute name="typeface" type="ST_TextTypeface" use="required"/>
  </complexType>
```

5.3.2.7 Format Scheme (Style Matrix)

The complex type CT_StyleMatrix defines a set of formatting options, which can be referenced by documents that apply a certain style to a given part of an object. For example, in a given shape, say a rectangle, one can reference a themed line style, themed effect, and themed fill that would be theme specific and change when the theme is changed. All of these formatting options are defined within this style matrix. Background fills can also be contained within the style matrix. This is most useful to presentations (but not unique to presentations) which reference different background fills as slide backgrounds. Three sets of each type of formatting are defined, corresponding to subtle, moderate, and intense versions of each style. Combinations of styles are used to create, for example a shape style. An example of this would be a shape style utilizing a subtle fill, moderate line, and intense effect to define the overall look of a shape.

The complex type CT_StyleMatrix is defined in the following manner:

```
  <complexType name="CT_StyleMatrix">
    <sequence>
      <element name="fillStyleLst" type="CT_FillStyleList" minOccurs="1"
        maxOccurs="1"/>
      <element name="lnStyleLst" type="CT_LineStyleList" minOccurs="1"
        maxOccurs="1"/>
      <element name="effectStyleLst" type="CT_EffectStyleList"
        minOccurs="1" maxOccurs="1"/>
      <element name="bgFillStyleLst" type="CT_BackgroundFillStyleList"
        minOccurs="1" maxOccurs="1"/>
    </sequence>
    <attribute name="name" type="xsd:string" use="optional" default=""/>
  </complexType>
```

5.3.2.8 Fill Style List

The complex type CT_FillStyleList defines a set of three fill types. Currently, only three fill types are used, corresponding to subtle, moderate, and intense fills, but the number of fills that can be defined is unbounded. An example of three fills that could be present can be seen in figure 3. In this figure, we have a solid blue fill in the subtle slot, a gradient fill in the moderate slot, and an image fill in the intense slot.

Figure 3: Three different fills increasing in relative intensity.

The complex type CT_FillStyleList is defined in the following manner:

```
<complexType name="CT_FillStyleList">
  <sequence>
    <group ref="EG_FillProperties" minOccurs="3" maxOccurs="unbounded"/>
  </sequence>
</complexType>
```

5.3.2.9 Line Style List

The complex type CT_LineStyleList defines a set of three line styles. As with the fill style list, currently only three styles are utilized corresponding to a subtle line, moderate line, and intense line.

The complex type CT_LineStyleList is defined in the following manner:

```
<complexType name="CT_LineStyleList">
  <sequence>
    <element name="ln" type="CT_LineProperties" minOccurs="3"
      maxOccurs="unbounded"/>
  </sequence>
</complexType>
```

5.3.2.10 Effect Style List

The complex type CT_EffectStyleList defines a set of three effect styles. As with the previously mentioned style lists, three styles are currently utilized corresponding to subtle, moderate, and intense effect styles, but the list remains unbounded. In figure 4 we see subtle, moderate, and intense effects applied to a given shape with a blue fill. The subtle effect is, basically, no effect, whereas the moderate effect is a glow surrounding the shape, and the intense effect is a 3-D bevel along with a shadow applied to the shape.

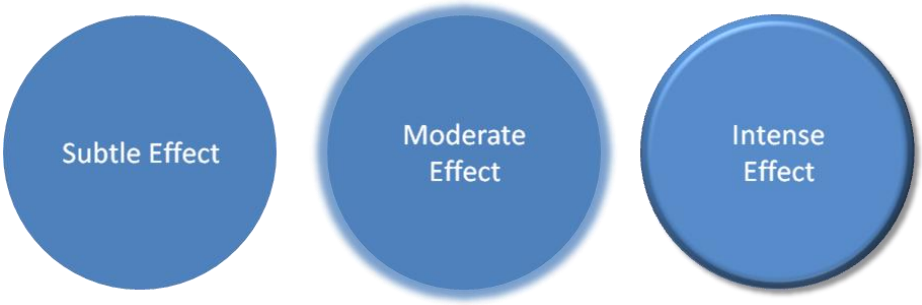


Figure 4: Subtle, moderate, and intense effects applied to a shape that has a blue fill.

The complex type CT_EffectStyleList is defined in the following manner:

```
<complexType name="CT_EffectStyleList">
  <sequence>
    <element name="effectStyle" type="CT_EffectStyleItem" minOccurs="3"
      maxOccurs="unbounded"/>
  </sequence>
</complexType>
```

5.3.2.11 Effect Style Item

The complex type CT_EffectStyleItem holds the properties for a given effect style. Within this complex type, one can define a list of effects (blur, shadow, reflection, etc.) along with any 3-D properties that are to be applied to an object. A basic example of how effects can be applied to a shape can be seen in figure 4.

The complex type CT_EffectStyleItem is defined in the following manner:

```
<complexType name="CT_EffectStyleItem">
  <sequence>
    <group ref="EG_EffectProperties" minOccurs="1" maxOccurs="1"/>
    <element name="scene3d" type="CT_Scene3D" minOccurs="0"
      maxOccurs="1"/>
    <element name="sp3d" type="CT_Shape3D" minOccurs="0" maxOccurs="1"/>
  </sequence>
</complexType>
```

5.3.2.12 Background Fill Style List

The complex type CT_BackgroundFillStyleList defines a set of three fill types similar to the fill style list. Again, they define three fill types corresponding to subtle, moderate, and intense background fills but the list itself is unbounded. The background-fills are meant, for example, to be applied to a slide background, or as the background fill in a shape or table.

The complex type CT_BackgroundFillStyleList is defined in the following manner:

```
<complexType name="CT_BackgroundFillStyleList">
  <sequence>
    <group ref="EG_FillProperties" minOccurs="3" maxOccurs="unbounded"/>
  </sequence>
</complexType>
```

5.3.2.13 Table Styles

Table styles are responsible for the rapid formatting that can be applied to a table. This rapid formatting takes different things into account, such as if the first row or last row should be emphasized, or if there is some type of banding present on the table. All of these different types of formatting can be defined within a table style. An example of different table styles in use on the same table can be seen in figure 1.

Figure 5: Different table styles in use.

The application of a table style to a table formats the table in its entirety. There are numerous complex types that make up a table style. The pieces of a table style will be discussed first, before defining the table style itself.

5.3.2.14 Cell 3D

The complex type CT_Cell3D defines all of the 3-D properties that an individual cell can hold. In the case of a table, these 3-D properties can be a bevel along with a material and a light rig for the cell. More explanation of these three pieces of a CT_Cell3D can be found in the document on 3-D. These properties are applied on a per-cell basis, rather than to the table as a whole. A CT_Cell3D is defined in the following manner:

```
<xsd:complexType name="CT_Cell3D">
  <xsd:sequence>
    <xsd:element name="bevel" type="CT_Bevel" minOccurs="1"
      maxOccurs="1" />
```

```
      <xsd:element name="lightRig" type="CT_LightRig" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="ext" type="CT_OfficeArtExtension" minOccurs="0"
        maxOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="prstMaterial" type="ST_PresetMaterialType"
      use="optional" default="plastic" />
  </xsd:complexType>
```

This complex type also holds a CT_OfficeArtExtension. This complex type is used for future extensibility and will be seen elsewhere throughout the tables area.

5.3.2.15 Themeable Styles

There are three groups and a complex type that account for style pieces that can be themed. These themed-aspects either pull from the style matrix, or they define an actual fill or effect for example. If they pull their style from the matrix, then an update to the document theme will also update the particular style dynamically. The three groups consist of the following groups:

```
  <xsd:group name="EG_ThemeableFillStyle">
    <xsd:choice>
      <xsd:element name="fill" type="CT_FillProperties" minOccurs="1"
        maxOccurs="1" />
      <xsd:element name="fillRef" type="CT_StyleMatrixReference"
        minOccurs="1" maxOccurs="1" />
    /xsd:choice>
  </xsd:group>
  <xsd:group name="EG_ThemeableEffectStyle">
    <xsd:choice>
      <xsd:element name="effect" type="CT_EffectProperties" minOccurs="1"
        maxOccurs="1" />
      <xsd:element name="effectRef" type="CT_StyleMatrixReference"
        minOccurs="1" maxOccurs="1" />
    </xsd:choice>
  </xsd:group>
  <xsd:group name="EG_ThemeableFontStyles">
    <xsd:choice>
      <xsd:element name="font" type="CT_FontCollection" minOccurs="1"
        maxOccurs="1" />
      <xsd:element name="fontRef" type="CT_FontReference" minOccurs="1"
        maxOccurs="1" />
    </xsd:choice>
  </xsd:group>
```

The three groups above all give a choice between using a themed style, or defining the style themselves. The last type in this group is a complex type used to perform the same task as the above three, only it deals with the lines in the table. The complex type CT_ThemeableLineStyle is defined as:

```
<xsd:complexType name="CT_ThemeableLineStyle">
  <xsd:choice>
    <xsd:element name="ln" type="CT_LineProperties" minOccurs="1"
      maxOccurs="1" />
    <xsd:element name="lnRef" type="CT_StyleMatrixReference"
      minOccurs="1" maxOccurs="1" />
  </xsd:choice>
</xsd:complexType>
```

5.3.2.16 On/Off Property Definition

The simple type ST_OnOffStyleType defines a type with values of on, off, or default. The default value means to follow the parent settings. This comes into play for a themed property, which means follow what the theme says. For an unthemed property, this means to follow the parent setting in the property inheritance chain.

5.3.2.17 Text Properties

The compelex type CT_TableStyleTextStyle defines the table text properties that can be styled. The text properties contain a reference to a themeable font style along with bold and italic being enabled or disabled. The CT_TableStyleTextStyle is defined in the following manner:

```
<xsd:complexType name="CT_TableStyleTextStyle">
  <xsd:sequence>
    <xsd:group ref="EG_ThemeableFontStyles" minOccurs="0"
      maxOccurs="1" />
    <xsd:group ref="EG_ColorChoice" minOccurs="0" maxOccurs="1" />
      <xsd:element name="ext" type="CT_OfficeArtExtension" minOccurs="0"
        maxOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="b" type="ST_OnOffStyleType" use="optional"
    default="def" />
  <xsd:attribute name="i" type="ST_OnOffStyleType" use="optional"
    default="def" />
</xsd:complexType>
```

5.3.2.18 Cell Border Properties

The complex type CT_TableCellBorderStyle defines the properties of the borders that can be styled in a table. The border styles can be applied to the following different types of borders in a table:

- left – left border
- right – right border
- top – top border

- bottom – bottom border
- insideH – inner horizontal borders
- insideV – inner vertical borders
- tl2br – diagonal border from top left corner to bottom right corner
- tr2bl – diagonal border from top right corner to bottom left corner

The complex type is defined in the following manner:

```
<xsd:complexType name="CT_TableCellBorderStyle">
  <xsd:sequence>
    <xsd:element name="left" type="CT_ThemeableLineStyle" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="right" type="CT_ThemeableLineStyle" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="top" type="CT_ThemeableLineStyle" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="bottom" type="CT_ThemeableLineStyle"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="insideH" type="CT_ThemeableLineStyle"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="insideV" type="CT_ThemeableLineStyle"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="tl2br" type="CT_ThemeableLineStyle" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="tr2bl" type="CT_ThemeableLineStyle" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="ext" type="CT_OfficeArtExtension" minOccurs="0"
      maxOccurs="1" />
  </xsd:sequence>
</xsd:complexType>
```

5.3.2.19 Cell Style Properties

The complex type CT_TableStyleCellStyle contains the definition for cell properties which can be styled. Within this complex type are held the border style, cell fill style, and the cell 3-D. The complex type is defined in the following manner:

```
<xsd:complexType name="CT_TableStyleCellStyle">
  <xsd:sequence>
    <xsd:element name="tcBdr" type="CT_TableCellBorderStyle"
      minOccurs="0" maxOccurs="1" />
```

```
      <xsd:group ref="EG_ThemeableFillStyle" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="cell3D" type="CT_Cell3D" minOccurs="0"
        maxOccurs="1" />
    </xsd:sequence>
  </xsd:complexType>
```

5.3.2.20 Table Background Style

The complex type CT_TableBackgroundStyle defines the style elements associated with the background of the table. The table background style can contain a fill and effect. The complex type is defined in the following manner:

```
  <xsd:complexType name="CT_TableBackgroundStyle">
    <xsd:sequence>
      <xsd:group ref="EG_ThemeableFillStyle" minOccurs="0"
        maxOccurs="1" />
      <xsd:group ref="EG_ThemeableEffectStyle" minOccurs="0"
        maxOccurs="1" />
    </xsd:sequence>
  </xsd:complexType>
```

5.3.2.21 Table Part Style

The complex type CT_TablePartStyle defines a structure for holding the style information for a single part of the table. The table is broken up in 13 different parts, which are explained in the next subclause of this document. A table part contains a text style and a cell style and is defined in the following manner:

```
  <xsd:complexType name="CT_TablePartStyle">
    <xsd:sequence>
      <xsd:element name="tcTxStyle" type="CT_TableStyleTextStyle"
        minOccurs="0" maxOccurs="1" />
      <xsd:element name="tcStyle" type="CT_TableStyleCellStyle"
        minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
  </xsd:complexType>
```

5.3.2.22 Table Style

The complex type CT_TableStyle defines the actual table style. Apart from the table background, 13 different parts that can be defined in a table style. These parts work together to define the styling for a table, given the 6 combinations of on/off states for the first row, first column, last row, last column, row banding, and column banding options. The different parts of a table that make up a table style are:

- tableBg – table background (this is not a CT_TablePartStyle)
- wholeTable – formatting for the entire table

- band1Horizontal – applied when row banding is enabled, this is the first row style, which alternates with band2Horiztonal
- band2Horizontal – applied when row banding is enabled, this is the second row style, which alternates with band1Horizontal
- band1Vertical – applied when column banding is enabled, this is the first column style, which alternates with band2Vertial
- band2Vertical – applied when column banding is enabled, this is the second column style, which alternates with band1Vertical
- lastCol – formatting applied to the last column when last column formatting is enabled
- firstCol – formatting applied to the first column when first column formatting is enabled
- lastRow – formatting applied to the last row when last row formatting is enabled
- firstRow – formatting applied to the first row when first row formatting is enabled
- seCell – formatting applied to the cell in the southeast corner of the table when last column and last row are enabled
- swCell – formatting applied to the cell in the southwest corner of the table when first column and last row are enabled
- neCell – formatting applied to the cell in the northeast corner of the table when the last column and first row are enabled
- nwCell – formatting applied to the cell in the northwest corner of the table when the first column and first row are enabled

The table style is defined in the following manner:

```
  <xsd:complexType name="CT_TableStyle">
    <xsd:sequence>
      <xsd:element name="tblBg" type="CT_TableBackgroundStyle"
        minOccurs="0" maxOccurs="1" />
      <xsd:element name="wholeTbl" type="CT_TablePartStyle" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="band1H" type="CT_TablePartStyle" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="band2H" type="CT_TablePartStyle" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="band1V" type="CT_TablePartStyle" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="band2V" type="CT_TablePartStyle" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="lastCol" type="CT_TablePartStyle" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="firstCol" type="CT_TablePartStyle" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="lastRow" type="CT_TablePartStyle" minOccurs="0"
        maxOccurs="1" />
```

```
      <xsd:element name="seCell" type="CT_TablePartStyle" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="swCell" type="CT_TablePartStyle" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="firstRow" type="CT_TablePartStyle" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="neCell" type="CT_TablePartStyle" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="nwCell" type="CT_TablePartStyle" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="ext" type="CT_OfficeArtExtension" minOccurs="0"
        maxOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="styleId" type="ST_Guid" use="required" />
    <xsd:attribute name="styleName" type="xsd:string" use="required" />"
  </xsd:complexType>
```

Also contained within the table style are an ID and a name. The name shows up as the name for the table style, and the ID is the unique id (GUID) that is associated with the table style.

5.3.2.23 Table Style List

The final complex type dealing with table styles is simply a list of table styles. Also contained in this list is the default style which gets applied to the table when the a default is to be used. The complex type CT_TableStyleList is defined in the following manner:

```
  <xsd:complexType name="CT_TableStyleList">
    <xsd:sequence>
      <xsd:element name="tblStyle" type="CT_TableStyle" minOccurs="0"
        maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="def" type="ST_Guid" use="required" />
  </xsd:complexType>
```

5.4 Text

5.4.1 Introduction

This subclause provides a high-level overview of the content described in the following schemas: dml-text.xsd, dml-textParagraph.xsd, dml-textRun.xsd, dml-textCharacter.xsd and dml-textBullet.xsd.

The best way to understand these schemas as they relate to one another is to learn about the DrawingML Text file format in the following order.

- Text Overview
- Body Level Properties

- Paragraph Level Properties
- Run and Character Level Properties

Companion schemas build on the ones discussed in this document. As these are encountered below, pointers to them are provided.

This subclause provides a structured breakdown of the text portion of the DrawingML file format. Other subclauses build on this foundation and explain more about topics such as text frame, text styles, text fields, and embedded fonts.

Note that DrawingML Text described within this document is distinct from WordprocessingML Text in that the file framework surrounding it has been optimized for use in a graphics-centric, presentation-like manner. As a contrast, WordprocessingML Text allows for text to be stored in a format that is optimized for layout of printed documents but not for art-based text as is found within the <a:> namespace.

Note that the use of the "p" namespace within this document references to the PresentationML-specific schemas while the use of the "a" namespace within this document references to the DrawingML-specific schemas.

5.4.2 Overview

Consider an XML tree that has the following basic structure:

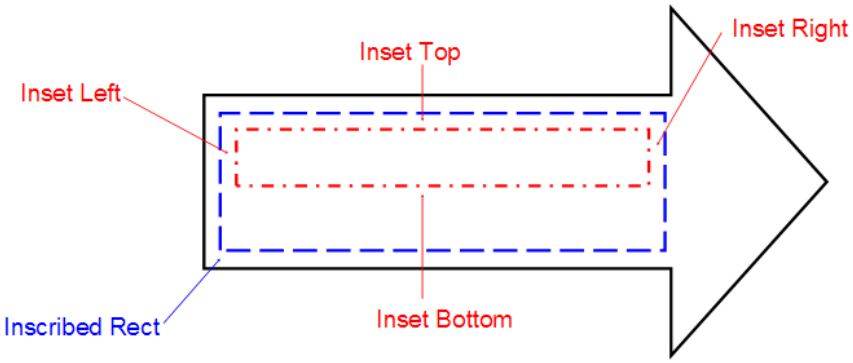
```
<p:sld>
  <p:cSld>
    <p:spTree>
      <p:sp>
        <p:txBody>
          Your text here!
   (not really, there are other elements needed within here)
        </p:txBody>
      </p:sp>
      <p:sp>
      </p:sp>
      …
    </p:spTree>
  </p:cSld>
</p:sld>
```

In the structure above, we are interested in the content contained within the matching p:txBody tags. The understanding of this tag in relation to the basic slide structure above encompasses the schema background needed to digest effectively the remainder of this description.

Note that shapes are the standard container within which all text resides. Usually, the shape does not have any visual properties attached to it and thus no visible shape is rendered; nonetheless, a shape is still present and does house any content text.

Each shape contains an inset rectangle that houses any text attached to that shape. The shape has margins or *insets* that buffer this rectangle on all four sides (top, bottom, left, and right) just like margins on a page. When thinking about text within a shape, it is useful to keep these inset properties in mind.

An illustration of this is provided below.



Let's look at the different element tags contained within p:txBody. Listed below are only those tags discussed here. (Note that this sample framework is a skeleton and does not fully show all elements and attributes needed.)

```
<p:txBody>
  <a:bodyPr />          required, only listed once.
  <a:lstStyle />        optional, if present only listed once.
  <a:p>                 required, no limit on the number of instances.
    <a:pPr />           optional, if present only listed once.
    <a:r>               required, no limit on the number of instances.
      <a:rPr />         optional, if present only listed once.
      <a:t>Your text here!</a:t>
                        Actual text for this run is contained here.
    </a:r>
    <a:endParaRPr />    optional, if present only listed once.
  </a:p>
</p:txBody>
```

Element	Purpose	Description
a:bodyPr	Body Properties	Describes text anchor points, shape autofit, number of columns, text warping, and 3D scenes and lighting effects. See §5.4.3.
a:lstStyle	List Style	Used to define style properties for the paragraph and its nine list levels.

a:p	Single Paragraph	Houses a single paragraph and its corresponding paragraph-level properties. Contained within here are also all the text runs that comprise this paragraph. See §5.4.3.4.
a:pPr	Paragraph Properties	Describes the format and style with which the corresponding paragraph is presented. Some possible settings that can be utilized within this space include, but are not limited to, the following: spacing, margins, and alignment. See §5.4.3.4.
a:r	Single Run	Specifies the existence of a run of text within a paragraph. A run represents the most granular form of text that can be represented in the file format. See §5.4.3.8.
a:rPr	Run Properties	Allows the attachment of properties to the run of text specified by its parent a:r element. These properties include, but are not limited to, the following: underline, strikethrough, and text caps. See §5.4.3.8.
a:t	Actual text	Allows for the storage of the specific text that all these body, paragraph and run level properties are describing. This tag is the most important as it gives context to all the other elements and attributes that have come before it.
a:endParaRPr	Persistent Run Properties	Specifies the properties that are to persist should the user begin to type additional text after this paragraph. This property should only be set when the style that should follow this paragraph is different from the paragraph itself.

5.4.3 Body Level Properties

Schemas represented here: dml-text.xsd

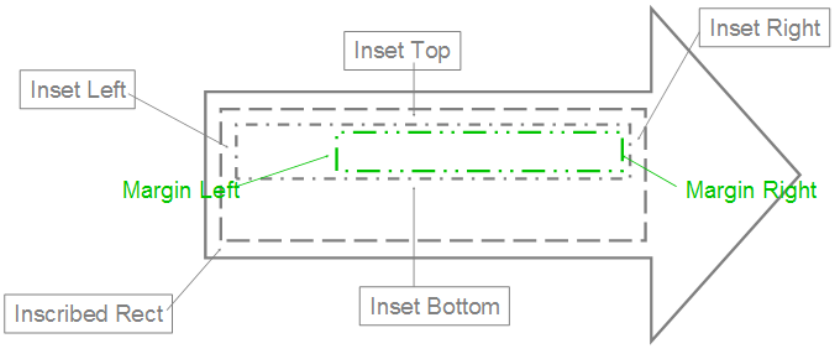
In this subclause, we'll explore the sorts of properties that can be attached to the body as a whole. As shown in the sample XML above, there are three essential property levels available. The body-level properties are to the broadest of these. Note that some of these body-level properties are applied as attributes to the body property tag while others are expressed as child elements. The specific method by which each property is applied can be found in the schemas listed above.

```
<p:txBody>
  <a:bodyPr /> ← Main element covered in this subclause
  <a:lstStyle />
  <a:p>
    <a:pPr />
    <a:r>
      <a:rPr />
      <a:t>Your text here!</a:t>
    </a:r>
  </a:p>
</p:txBody>
```

5.4.3.1 Setting Up the Text Area

Let's start with how a text area might be initially described. This area is the container within which all the child text for this body resides. First, it is useful to understand the inset properties; specifically, the top, bottom, left and right inset properties that are also known as internal margins for the text body. The anchor attribute allows us to specify where the text area should be anchored within its bounding rectangle.

An illustration of this bounding rectangle is highlighted below by the inner green box. Notice here that the bounding rectangle is anchored to the right.



Here's how the text will appear inside. Attribute AutoFit allows for three basic scenarios:

- No AutoFit: The text is allowed to flow outside the container.
- Normal AutoFit: The text is resized using defined constraints to fit inside the container area. (This is used when the text is too large or long to fit in the text container.)
- Shape AutoFit: The actual text container is resized to contain all the text. (This is the only option that can cause the container to have its dimensions changed.)

The term *flow* is used to describe the way in which text moves around inside this text area, and to describe how each of the body properties affects the text within the text area.

One way that text can flow is from one line to the next. This can be done automatically by using the text-wrapping attribute. Another way is to use columns. The XML framework allows for the specification of a

number of columns into which the text is to be automatically broken. This feature also allows for the specifying of the spacing of columns and a right-to-left layout instead of the default left-to-right. Another way that text can flow is vertical instead of horizontal. For this, there are many different types of vertical text that can be described: from text that appears rotated to text where the characters are truly stacked. The text can even be made to flow differently when an East Asian font is specified.

When looking at the flow it is useful to discuss the potential for *overflow*. That is, the text must flow outside the text area because it is too large to fit inside. For this, there are two common types: vertical and horizontal. The vertical overflow can be handled in three ways:

- overflow: This allows the text to flow outside the text area.

- ellipsis: This crops the text that overflows and adds "…" to denote that there is hidden text.

- clip: This crops the text just as ellipsis but does not insert "…", so the user has no indication that there is hidden text.

- Horizontal overflow works exactly like vertical, but with only two options: overflow and clip, which both operate as described above.

5.4.3.2 Manipulating the Text

Let's look at the ways in which the text can be further enhanced at the text body level. Note that the properties that follow apply only to the text body as a whole and thus cannot be applied to a specific paragraph or run within the text body. Theses properties are as follows:

Text Warping: Text within the text area is made to distort itself according to a predefined shape. This shape resides within the bounding box described earlier. This effect is known as *text warping* and has its preset shapes defined further within dml-shapeGeometry.xsd

- 3D Text: Text can be described with respect to a 3D scene. Using this tag provides three basic options:
  - The text resides within a 3D scene, but as planar text.
  - The text is allowed to reside within the 3D scene and has 3D effects (such as bevel or extrusion) applied to it.
  - The text resides on top of a 3D scene. The 3D scene properties are defined dml-shape3DStyles.xsd and dml-shape3DScene.xsd schemas.
- Rotated and Upright Text: A particular rotation can be specified that is applied to the text within the text area. Note that this is different from the rotation that is applied to the shape within which the text area resides. If this attribute is not specifically set then the rotation of the container shape is used.

5.4.3.3 Backwards and Forwards Compatibility

The following areas are of interest when considering support for both past design and future innovation.

- From WordArt: This is specific to dealing with previous WordArt text. Now that text is described as simply a shape, there is no need for a WordArt-specific description. There is, however, the need to identify which pieces of text were from the old WordArt styles in case there is the need to write them back out in their old format.
- Future Extensions: The ability for future extensions has been provided to the body property tag via the ext tag. This can be used the widest way possible as it is a complex type and can thus describe the most complex future properties. Note that each of the schema subclauses below have their own ext tag.

5.4.3.4 Paragraph-Level Properties

Schemas represented here: dml-textParagraph.xsd, dml-textBullet.xsd.

In this subclause, we will explore what sorts of properties can be attached to a paragraph as a whole. Paragraph-level properties allow for a more granular description of the text than the properties of the body tag described earlier. Keep in mind that the properties that can be applied at this level are not duplicates of the body or run levels, but unique only to the paragraph element. Once again, it should be noted that some of these paragraph-level properties are applied as attributes to the paragraph property tag while others are expressed as child elements. The specific method by which each property is applied can be found in the schemas listed above.

```
<p:txBody>
```

```
      <a:bodyPr />
      <a:lstStyle />
      <a:p>
        <a:pPr />  ← Main element covered in this subclause
        <a:r>
          <a:rPr />
          <a:t>Your text here!</a:t>
        </a:r>
      </a:p>
    </p:txBody>
```

5.4.3.5 Spacing, Alignment, and Direction

The XML file format allows for the specifying of spacing both between lines in the form of line spacing, and also outside the paragraph via margins and special before/after spacing. In addition to this, there is also the ability to specify indent spacing for the beginning of the paragraph.

The standard alignment options include left-aligned, right-aligned, centered, justified, and distributed. Justified alignment causes each line of text to be stretched out to a certain point. To ensure that short lines remain readable, they are not stretched. Distributed alignment is quite similar, but stretches every line, regardless of that line's length.

Text direction is specified as either left-to-right (the default) or right-to-left using the specific rtl tag.

5.4.3.6 Tabs and Line Breaks

When the default tabs are not sufficient for the paragraph in question there is the option of including custom tab stops in the XML file format. The information required for this is both a default tab size attribute and a full tab stop list showing all tab stop positions that apply to this paragraph. Keep in mind that if tab stops are not explicitly stated in the file format that the business logic of the application must use its own default positions if tabs are needed.

Line break is a tag that informs the application as to whether it should break up a string of text onto multiple lines based on Latin grammar rules or East Asian grammar rules. The East Asian option uses the Kinsoku settings to determine whether a word is allowed to begin or end a line of text.

5.4.3.7 Adding Bullets

Bullets are specified per paragraph, so bullets can be mixed and matched within a single text body to appear as a coherent text group. The types of bullets available are:

- Character Bullets: Uses a font character to denote a bullet and can be set to appear in any size (percentage of text), color (all available including theme colors), and font. *The properties are Bullet Color, Bullet Size, Bullet Font Typeface, Bullet Character (represents the actual bullet)*

- g Bullet 1
  - g Bullet 2
- g Bullet 3

- Auto-Numbered Bullets: Uses the application logic to assign a series of numbers/characters to a specific bulleted item using just a bullet scheme and a starting number. (When a starting number is used, all bulleted paragraphs listed after the start number are automatically numbered based on this last known start number. The scope of this auto-numbering is only within its current text body, no start at number would ever carry over to a different text body.) The properties are Start At number, Bullet Scheme (letters, roman numerals, etc.), Bullet Color, Bullet Size, and Bullet Font Typeface.
  1. Bullet 1
     1. Bullet 2
  2. Bullet 3

- Blip Bullets: Uses a picture to denote a bulleted item. The only additional property available with this type of applied bullet is the size (percentage of text). If the graphic is not in the applications standard set of graphics then the attached graphic is converted to a PNG format, placed in the document container and is given a relationship id that is used later to reference the image. The properties are Embed id (corresponds to a bullet graphic) and Bullet Size.
  - Bullet 1
    - Bullet 2
  - Bullet 3

5.4.3.8 Run- and Character-Level Properties

Schemas represented here: dml-textRun.xsd, dml-textCharacter.xsd

In this subclause, we'll explore the most granular text properties available in this XML framework, namely those described at the text run and character level. This level is usually the level in which text is broken up into differently formatted parts, because the most commonly used text properties almost all reside at this level. This allows for some very detailed formatting to be represented. Again, it should be noted that for consistency that some of these run and character level properties are applied as attributes to the run property tag while others are expressed as child elements. The specific method by which each property is applied can be found in the schemas listed above.

```
<p:txBody>
  <a:bodyPr />
  <a:lstStyle />
  <a:p>
    <a:pPr />
    <a:r>
      <a:rPr />    ← Main element covered in this subclause
      <a:t>Your text here!</a:t>
    </a:r>
  </a:p>
</p:txBody>
```

5.4.3.9 Visual Properties

When looking to format a run of text the first property that one might need to specify would be the font typeface. The XML file format allows for the specification of not only Latin Fonts but also East Asian, Complex Script, and Symbol fonts as well. These four font buckets give the application additional information that is used to layout text in a manner fitting for the specific font. Along with the actual font being used, comes the size of the font. To specify this simply use the *sz* attribute and along with a value that is 1/100th of the size in points.

Other common formatting properties allowed in the XML framework are bold, italic, underline and strikethrough. The use of both the bold and italic properties is simply via a Boolean value of 0 or 1. The usage of the underline and strikethrough, however, allow a more specific selection to be made. There are 17 values for underline, which range from a single line to wavy double lines. In addition to specifying the style of underline that is to be used, the framework can also specify fill properties for the underline. These are solid color, multi-color gradient, and texture fill. For strikethrough, there are two options: single and double strike through.

When standard formatting isn't adequate, more complex effects can be defined for a specific run of text. The basic breakdown for these is line properties, fill properties and effect properties. Encapsulated within each of these areas is a wide range of customizable effects. A quick look at line properties, for example, reveals the ability to specify a color, gradient, or pattern fill, along with a width and style applied. Along these lines fill properties allows for transparent fill, solid fill, gradient fill, texture fill and even picture fill. While these features alone give the XML file format plenty of robustness in describing text, other features are also available. Because text is treated the same as a shape, a run of text can have virtually all shape effects applied to it just as if it were a shape. These effects include shadow, glow, and reflection, and are placed in an effect list under the run properties tag. An example of what these lines, fills and effects may look like is provided below. More information on these effects can be found in either dml-shapeLineProperties.xsd or dml-shapeEffects.xsd.

```
<a:rPr>
  <a:ln>
    <a:solidFill … />     ← Line properties here
  </a:ln>
  <a:gradFill>
    <a:gsLst … />         ← Fill properties here
  </a:gradFill>
  <a:effectLst>
    <a:reflection … />    ← Effect properties here
  </a:effectLst>
</a:rPr>
```

A few additional properties are worth noting:

- Minimum kerning size: This specifies the smallest font size at which kerning still occurs. When no tag is present for this the default value is 0, allowing kerning at any text size.
- Spacing between characters: The units here are the same as are used for font size. Along the lines of specifying horizontal spacing, vertical spacing can be specified via the baseline tag. This is typically used for subscript and superscript text and is specified in the same units as font size.
- Capitalization and Normalize: Capitalization sets the case of the character to either all small caps or all large caps. For this property there are only these two settings aside from the "none" setting at which point this property is ignored. Normalize height takes all shorter characters and adjusts their height up so that they are the same as taller characters. This property is set via a Boolean value.

5.4.3.10 Properties for Interactivity

Hyperlinks: The XML file format allows for the inputting of hyperlinks that are activated by either click or mouse over. These two tags are HyperlinkClick and HyperlinkMouseOver, respectfully. They both allow for the specifying of a link to another resource very much like those found on a common website.

Spelling and Smart Tags: Although spelling is very much an application-specific part of text editing and is most likely to be done within the application itself there are a few ways that spelling settings and preferences can be persisted within the file format. One way is through the spelling error bit, which simply saves whether there is a known spelling mistake. The next is the spelling dirty bit. This gets set whenever the user has entered new text and the application has not had a chance to check for spelling errors on this piece of text. Lastly, in this realm we actually have a user preference of no proofing that is persisted for the next time a document is opened. This allows the user to specify a word that they do not want to have checked for spelling. Along with spell-checking comes the notion of smart tags which must be checked for just like spelling mistakes. For this there are two related tags. The first is the smart tag clean, which allows for a boolean value to be set determining if this portion of text has been checked for the presence of new smart tags. The next is the actual smart tag id. Once a piece of text has been determined to be a smart tag then a smart tag id is assigned which points to the actual smart tag information.

5.4.3.11 International Language Support

There exists the notion of the language id, which is simply a value that assists the application in laying out the text. The tags that help with this are the language id tag and the alternate language id tag. Together, these allow the file format to be robust and handle multiple languages for a single run of text. In addition, there is also the kumimoji tag, which aids with the layout of East Asian text by specifying whether numbers appear vertically with text (default) or horizontally. An illustration of a run of text with kumimoji applied is provided below.

5.5 Tables

5.5.1 Introduction

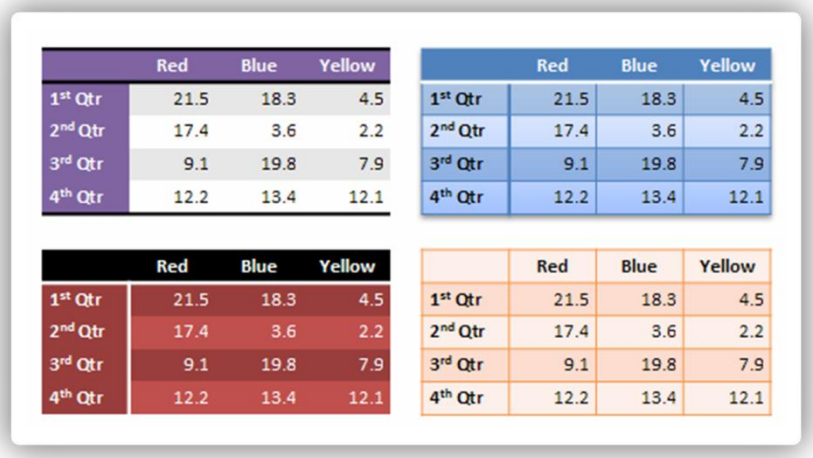
This document provides a high-level overview of the content described in the schemas dml-table.xsd and dml-tableStyle.xsd.

This aspect of DrawingML deals with the definition of a table and the associated styling information. The first part describes the table styles aspect, while the second part describes the definition of a table within DrawingML. The above-mentioned schemas fall into the following groupings:

Table Styles	Table Definition
dml-tableStyle.xsd	dml-table.xsd

5.5.2 Table Styles

Table styles are responsible for the rapid formatting that can be applied to a table. This rapid formatting takes different things into account, such as if the first row or last row should be emphasized, or if there is some type of banding present on the table. All of these different types of formatting can be defined within a table style. An example of different table styles in use on the same table can be seen in figure 1.



	Red	Blue	Yellow
1st Qtr	21.5	18.3	4.5
2nd Qtr	17.4	3.6	2.2
3rd Qtr	9.1	19.8	7.9
4th Qtr	12.2	13.4	12.1

	Red	Blue	Yellow
1st Qtr	21.5	18.3	4.5
2nd Qtr	17.4	3.6	2.2
3rd Qtr	9.1	19.8	7.9
4th Qtr	12.2	13.4	12.1

	Red	Blue	Yellow
1st Qtr	21.5	18.3	4.5
2nd Qtr	17.4	3.6	2.2
3rd Qtr	9.1	19.8	7.9
4th Qtr	12.2	13.4	12.1

	Red	Blue	Yellow
1st Qtr	21.5	18.3	4.5
2nd Qtr	17.4	3.6	2.2
3rd Qtr	9.1	19.8	7.9
4th Qtr	12.2	13.4	12.1

Figure 6: Different table styles in use.

The application of a table style to a table formats the table in its entirety. A number of complex types make up a table style. The pieces of a table style are discussed first, before the table style itself is defined.

5.5.2.1 Cell 3-D

The complex type, CT_Cell3D, defines all of the 3-D properties that an individual cell can hold. In the case of a table, these 3-D properties can be a bevel along with a material and a light rig for the cell. (More explanation of these three pieces of a CT_Cell3D can be found in §5.6.) These properties are applied on a per-cell basis, rather than to the table as a whole. A CT_Cell3D is defined in the following manner:

```
<xsd:complexType name="CT_Cell3D">
  <xsd:sequence>
    <xsd:element name="bevel" type="CT_Bevel" minOccurs="1"
      maxOccurs="1" />
    <xsd:element name="lightRig" type="CT_LightRig" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="extLst" type="CT_OfficeArtExtensionList"
      minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="prstMaterial" type="ST_PresetMaterialType"
    use="optional" default="plastic" />
</xsd:complexType>
```

This complex type also holds a CT_OfficeArtExtensionList. This complex type is used for future extensibility and will be seen elsewhere throughout the tables area.

5.5.2.2 Themeable Styles

Three groups and a complex type account for style pieces that can be themed. These themed aspects either pull from the style matrix, or they define an actual fill or effect for example. If they pull their style from the matrix, then an update to the document theme will also update the particular style dynamically. The three groups consist of the following groups:

```
<xsd:group name="EG_ThemeableFillStyle">
  <xsd:choice>
    <xsd:element name="fill" type="CT_FillProperties" minOccurs="1"
      maxOccurs="1" />
    <xsd:element name="fillRef" type="CT_StyleMatrixReference"
      minOccurs="1" maxOccurs="1" />
  </xsd:choice>
</xsd:group>
<xsd:group name="EG_ThemeableEffectStyle">
  <xsd:choice>
    <xsd:element name="effect" type="CT_EffectProperties"
      minOccurs="1" maxOccurs="1" />
    <xsd:element name="effectRef" type="CT_StyleMatrixReference"
      minOccurs="1" maxOccurs="1" />
  </xsd:choice>
</xsd:group>
<xsd:group name="EG_ThemeableFontStyles">
  <xsd:choice>
    <xsd:element name="font" type="CT_FontCollection" minOccurs="1"
      maxOccurs="1" />
    <xsd:element name="fontRef" type="CT_FontReference" minOccurs="1"
      maxOccurs="1" />
  </xsd:choice>
</xsd:group>
```

The three groups above all give a choice between using a themed style or defining the style themselves. The last type in this group is a complex type used to perform the same task as the above three, only it deals with the lines in the table. The complex type, CT_ThemeableLineStyle, is defined as:

```
<xsd:complexType name="CT_ThemeableLineStyle">
  <xsd:choice>
    <xsd:element name="ln" type="CT_LineProperties" minOccurs="1"
      maxOccurs="1" />
    <xsd:element name="lnRef" type="CT_StyleMatrixReference"
      minOccurs="1" maxOccurs="1" />
  </xsd:choice>
</xsd:complexType>
```

5.5.2.3 On/Off Property Definition

The simple type, ST_OnOffStyleType, defines a type with values of on, off, or default. A value of default indicates that parent settings should be used. Thus, for a themed property, default indicates that the theme properties should be followed. For an unthemed property, default means that the parent setting in the property inheritance chain should be followed.

5.5.2.4 Text Properties

The complex type, CT_TableStyleTextStyle, defines the table text properties that can be styled. The text properties contains a reference to a themeable font style along with bold and italic being enabled or disabled. The CT_TableStyleTextStyle is defined in the following manner:

```
<xsd:complexType name="CT_TableStyleTextStyle">
  <xsd:sequence>
    <xsd:group ref="EG_ThemeableFontStyles" minOccurs="0"
      maxOccurs="1" />
    <xsd:group ref="EG_ColorChoice" minOccurs="0" maxOccurs="1" />
      <xsd:element name="extLst" type="CT_OfficeArtExtensionList"
      minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="b" type="ST_OnOffStyleType" use="optional"
    default="def" />
  <xsd:attribute name="i" type="ST_OnOffStyleType" use="optional"
    default="def" />
</xsd:complexType>
```

5.5.2.5 Cell Border Properties

The complex type, CT_TableCellBorderStyle, defines the properties of the borders that can be styled in a table. The border styles can be applied to the following different types of borders in a table:

- `left` – left border
- `right` – right border
- `top` – top border
- `bottom` – bottom border
- `insideH` – inner horizontal borders
- `insideV` – inner vertical borders
- `tl2br` – diagonal border from top left corner to bottom right corner
- `tr2bl` – diagonal border from top right corner to bottom left corner

The complex type is defined in the following manner:

```
<xsd:complexType name="CT_TableCellBorderStyle">
  <xsd:sequence>
    <xsd:element name="left" type="CT_ThemeableLineStyle"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="right" type="CT_ThemeableLineStyle"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="top" type="CT_ThemeableLineStyle"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="bottom" type="CT_ThemeableLineStyle"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="insideH" type="CT_ThemeableLineStyle"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="insideV" type="CT_ThemeableLineStyle"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="tl2br" type="CT_ThemeableLineStyle"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="tr2bl" type="CT_ThemeableLineStyle"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="extLst" type="CT_OfficeArtExtensionList"
      minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
</xsd:complexType>
```

5.5.2.6 Cell Style Properties

The complex type, CT_TableStyleCellStyle, contains the definition for cell properties that can be styled. Within this complex type are held the border style, cell fill style, and the cell 3-D. The complex type is defined in the following manner:

```
<xsd:complexType name="CT_TableStyleCellStyle">
  <xsd:sequence>
    <xsd:element name="tcBdr" type="CT_TableCellBorderStyle"
      minOccurs="0" maxOccurs="1" />
    <xsd:group ref="EG_ThemeableFillStyle" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="cell3D" type="CT_Cell3D" minOccurs="0"
      maxOccurs="1" />
  </xsd:sequence>
</xsd:complexType>
```

5.5.2.7 Table Background Style

The complex type, CT_TableBackgroundStyle, defines the style elements associated with the background of the table. The table background style can contain a fill and effect. The complex type is defined in the following manner:

```
<xsd:complexType name="CT_TableBackgroundStyle">
  <xsd:sequence>
    <xsd:group ref="EG_ThemeableFillStyle" minOccurs="0"
      maxOccurs="1" />
    <xsd:group ref="EG_ThemeableEffectStyle" minOccurs="0"
      maxOccurs="1" />
  </xsd:sequence>
</xsd:complexType>
```

5.5.2.8 Table Part Style

The complex type, CT_TablePartStyle, defines a structure for holding the style information for a single part of the table. The table is broken up in 13 different parts which will be explained in the next subclause of this document. A table part contains a text style and a cell style and is defined in the following manner:

```
<xsd:complexType name="CT_TablePartStyle">
  <xsd:sequence>
    <xsd:element name="tcTxStyle" type="CT_TableStyleTextStyle"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="tcStyle" type="CT_TableStyleCellStyle"
      minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
</xsd:complexType>
```

5.5.2.9 Table Style

The complex type, CT_TableStyle, defines the actual table style. There are thirteen different parts (outside of the table background) that can be defined in a table style. These parts work together to define the styling for a table, given the six combinations of on/off states for the first row, first column, last row, last column, row banding, and column banding options. The different parts of a table that make up a table style are:

- tableBg – table background (this is not a CT_TablePartStyle)
- wholeTable – formatting for the entire table
- band1Horizontal – applied when row banding is enabled, this is the first row style, which alternates with band2Horiztonal
- band2Horizontal – applied when row banding is enabled, this is the second row style, which alternates with band1Horizontal
- band1Vertical – applied when column banding is enabled, this is the first column style, which alternates with band2Vertial
- band2Vertical – applied when column banding is enabled, this is the second column style, which alternates with band1Vertical
- lastCol – formatting applied to the last column when last column formatting is enabled
- firstCol – formatting applied to the first column when first column formatting is enabled
- lastRow – formatting applied to the last row when last row formatting is enabled
- firstRow – formatting applied to the first row when first row formatting is enabled

- seCell – formatting applied to the cell in the southeast corner of the table when last column and last row are enabled
- swCell – formatting applied to the cell in the southwest corner of the table when first column and last row are enabled
- neCell – formatting applied to the cell in the northeast corner of the table when the last column and first row are enabled
- nwCell – formatting applied to the cell in the northwest corner of the table when the first column and first row are enabled

The table style is defined in the following manner:

```
<xsd:complexType name="CT_TableStyle">
  <xsd:sequence>
    <xsd:element name="tblBg" type="CT_TableBackgroundStyle"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="wholeTbl" type="CT_TablePartStyle"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="band1H" type="CT_TablePartStyle" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="band2H" type="CT_TablePartStyle" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="band1V" type="CT_TablePartStyle" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="band2V" type="CT_TablePartStyle" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="lastCol" type="CT_TablePartStyle"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="firstCol" type="CT_TablePartStyle"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="lastRow" type="CT_TablePartStyle"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="seCell" type="CT_TablePartStyle" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="swCell" type="CT_TablePartStyle" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="firstRow" type="CT_TablePartStyle"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="neCell" type="CT_TablePartStyle" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="nwCell" type="CT_TablePartStyle" minOccurs="0"
      maxOccurs="1" />
```

```
      <xsd:element name="extLst" type="CT_OfficeArtExtensionList"
        minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="styleId" type="ST_Guid" use="required" />
    <xsd:attribute name="styleName" type="xsd:string" use="required" />
  </xsd:complexType>
```

Also contained within the table style is an ID and a name. The name shows up as the name for the table style and the ID is the unique id (GUID) that is associated with the table style.

5.5.2.10 Table Style List

The final complex type dealing with table styles is simply a list of table styles. Also contained in this list is the default style which is applied to the table when the a default is to be used. The complex type, CT_TableStyleList, is defined in the following manner:

```
  <xsd:complexType name="CT_TableStyleList">
    <xsd:sequence>
      <xsd:element name="tblStyle" type="CT_TableStyle" minOccurs="0"
        maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="def" type="ST_Guid" use="required" />
  </xsd:complexType>
```

5.5.3 Table Definition

In this subclause, the focus is on the actual definition of a table and the data contained within the table. There are not as many complex types in this subclause as with the table style subclause, but they are organized in the same way as the table style section.

5.5.3.1 Cell Properties

The complex type, CT_TableCellProperties, holds all the information that deals with the properties of a given cell. The cell properties contain a section for the different line properties (ln*), the cell fill properties, the 3-D properties, cell margin information (mar*), anchoring information (anchor and anchorCtr), a vertical text type, and finally an attribute which defines the behavior of horizontal text overflow (horzOverflow). As with many other types defined in this document, CT_TableCellProperties contains an element reserved for future extensibility. The complex type is defined in the following manner:

```
  <xsd:complexType name="CT_TableCellProperties">
    <xsd:sequence>
      <xsd:element name="lnL" type="CT_LineProperties" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="lnR" type="CT_LineProperties" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="lnT" type="CT_LineProperties" minOccurs="0"
        maxOccurs="1" />
```

```
      <xsd:element name="lnB" type="CT_LineProperties" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="lnTlToBr" type="CT_LineProperties"
        minOccurs="0" maxOccurs="1" />
      <xsd:element name="lnBlToTr" type="CT_LineProperties"
        minOccurs="0" maxOccurs="1" />
      <xsd:element name="cell3D" type="CT_Cell3D" minOccurs="0"
        maxOccurs="1" />
      <xsd:group ref="EG_FillProperties" minOccurs="0" maxOccurs="1" />
      <xsd:element name="extLst" type="CT_OfficeArtExtensionList"
        minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="marL" type="ST_Coordinate32" use="optional"
      default="91440" />
    <xsd:attribute name="marR" type="ST_Coordinate32" use="optional"
      default="91440" />
    <xsd:attribute name="marT" type="ST_Coordinate32" use="optional"
      default="45720" />
    <xsd:attribute name="marB" type="ST_Coordinate32" use="optional"
      default="45720" />
    <xsd:attribute name="vert" type="ST_TextVerticalType" use="optional"
      default="horz" />
    <xsd:attribute name="anchor" type="ST_TextAnchoringType"
      use="optional" default="t" />
    <xsd:attribute name="anchorCtr" type="xsd:boolean" use="optional"
      default="false" />
    <xsd:attribute name="horzOverflow" type="ST_TextHorzOverflowType"
      use="optional" default="clip" />
  </xsd:complexType>
```

5.5.3.2 Column

The complex type, CT_TableCol, defines a table column element. The table column element simply holds the width for a given column in a table along with an element reserved for future extensibility. The complex type is defined as:

```
  <xsd:complexType name="CT_TableCol">
    <xsd:sequence>
      <xsd:element name="extLst" type="CT_OfficeArtExtensionList"
        minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="w" type="ST_Coordinate" use="required" />
  </xsd:complexType>
```

5.5.3.3 Table Grid

The complex type, CT_TableGrid, defines a list of table column elements, or rather CT_TableCol complex types. The CT_TableGrid should contain a CT_TableCol for each column in the table and it is defined in the following manner:

```
<xsd:complexType name="CT_TableGrid">
  <xsd:sequence>
    <xsd:element name="gridCol" type="CT_TableCol" minOccurs="0"
      maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>
```

5.5.3.4 Cell

The complex type, CT_TableCell, defines a cell in a table. Within this complex type lies a text body which holds the data of the cell along with any formatting applied to the text. This complex type also holds a table cell property complex type which has already been defined. The rowSpan and gridSpan attributes are available along with hMerge and vMerge attributes. The hMerge and vMerge attributes define if the current cell is supposed to be merged with the previous cell horizontally or vertically. This is how the table is parsed and created. The complex type, CT_TableCell, is defined as:

```
<xsd:complexType name="CT_TableCell">
  <xsd:sequence>
    <xsd:element name="txBody" type="CT_TextBody" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="tcPr" type="CT_TableCellProperties"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="extLst" type="CT_OfficeArtExtensionList"
      minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="rowSpan" type="xsd:int" use="optional"
    default="1" />
  <xsd:attribute name="gridSpan" type="xsd:int" use="optional"
    default="1" />
  <xsd:attribute name="hMerge" type="xsd:boolean" use="optional"
    default="false" />
  <xsd:attribute name="vMerge" type="xsd:boolean" use="optional"
    default="false" />
</xsd:complexType>
```

5.5.3.5 Row

The complex type, CT_TableRow, defines a table row. This complex type is somewhat more complex than the similar table column complex type in that it holds a sequence of CT_TableCell structures along with a height for the row. The complex type is defined in the following way:

```
<xsd:complexType name="CT_TableRow">
  <xsd:sequence>
    <xsd:element name="tc" type="CT_TableCell" minOccurs="0"
      maxOccurs="unbounded" />
    <xsd:element name="extLst" type="CT_OfficeArtExtensionList"
      minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="h" type="ST_Coordinate" use="required" />
</xsd:complexType>
```

5.5.3.6 Table Properties

The complex type, CT_TableProperties, defines the properties for a table as a whole. Within this complex type is a definition for a table style that is currently applied to the table, or the GUID for the built in table style that is applied to the table. Also in this complex type are right-to-left settings, the effects applied to the table (shadow, reflection, etc), background fill information, and the states for the different on/off table style options. The complex type is defined as:

```
<xsd:complexType name="CT_TableProperties" >
  <xsd:sequence>
    <xsd:group ref="EG_FillProperties" minOccurs="0" maxOccurs="1" />
    <xsd:group ref="EG_EffectProperties" minOccurs="0"
      maxOccurs="1" />
    <xsd:choice oxsd:cname="TableStyleOrLink" minOccurs="0"
      maxOccurs="1" >
      <xsd:element name="tableStyle" type="CT_TableStyle" />
      <xsd:element name="tableStyleId" type="ST_Guid" />
    </xsd:choice>
    <xsd:element name="extLst" type="CT_OfficeArtExtensionList"
      minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="rtl" type="xsd:boolean" use="optional"
    default="false" />
  <xsd:attribute name="firstRow" type="xsd:boolean" use="optional"
    default="false" />
  <xsd:attribute name="firstCol" type="xsd:boolean" use="optional"
    default="false" />
  <xsd:attribute name="lastRow" type="xsd:boolean" use="optional"
    default="false" />
  <xsd:attribute name="lastCol" type="xsd:boolean" use="optional"
    default="false" />
  <xsd:attribute name="bandRow" type="xsd:boolean" use="optional"
    default="false" />
```

```
    <xsd:attribute name="bandCol" oxsdtype="xsd:boolean" use="optional"
      default="false" />
  </xsd:complexType>
```

5.5.3.7 Table

The final complex type, CT_Table, is the root element for a table. This complex type holds all the information that is needed to create a table within DrawingML. Within CT_Table are the table properties, a table grid, and a table row. A CT_Table is defined in the following manner:

```
  <xsd:complexType name="CT_Table">
    <xsd:sequence>
      <xsd:element name="tblPr" type="CT_TableProperties" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="tblGrid" type="CT_TableGrid" minOccurs="1"
        maxOccurs="1" />
      <xsd:element name="tr" type="CT_TableRow" minOccurs="0"
        maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
```

5.6 3D Aspects

5.6.1 Introduction

This subclause provides a high-level overview of the content described in the following schemas: dml-shape3DStyles.xsd, dml-shape3DScene.xsd, dml-shape3DScenePlane.xsd, dml-shape3DLighting.xsd, and dml-shape3DCamera.xsd.

This aspect of DrawingML deals mainly with the 3-D aspects, and can be broken down into two topics: 3-D properties associated with an object, and the styling information associated with an object. The above-mentioned schemas fall into the grouping in the following way:

3-D	Styles
dml-shape3DScene.xsd dml-shape3DScenePlane.xsd dml-shape3DLighting.xsd dml-shape3DCamera.xsd	dml-shape3DStyles.xsd

5.6.2 3-D

Here we'll explain the 3-D definitions contained in DrawingML. The goal here is to define a 3-D scene so that lighting calculations can be made on the geometry within the scene.

5.6.2.1 3-D Scene

Every 3-D scene consists of a camera, a light, and a backdrop, that define the associated properties of the scene. The complex type, CT_Scene3D, defines the scene as follows:

```
<xsd:complexType name="CT_Scene3D" oxsd:cname="Scene3D">
  <xsd:sequence>
    <xsd:element name="camera" type="CT_Camera" minOccurs="1"
      maxOccurs="1" />
    <xsd:element name="lightRig" type="CT_LightRig" minOccurs="1"
      maxOccurs="1/>
    <xsd:element name="backdrop" type="CT_Backdrop" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="ext" type="CT_OfficeArtExtension" minOccurs="0"
      maxOccurs="1" />
  </xsd:sequence>
</xsd:complexType>
```

As was stated above, the complex type, CT_Scene3D, contains a camera, a set of lights (the light rig), and a backdrop. Those familiar with 3-D rendering techniques understand the usage of a camera and set of lights, or light rig. The backdrop, however, is a special structure (which will be defined below) that allows for a special plane to render certain effects that need to be rendered together in a single plane. The final element of a CT_Scene3D is the ext element. This is a DrawingML structure used for future extensibility. This element will be seen in other complex types dealing with the 3-D scene as well.

5.6.2.2 Camera

The complex type, CT_Camera, defines a camera within the 3-D scene. A camera is based on a preset, with an optional rotation, field-of-view, and zoom, which all act as overrides for the preset values. A camera is defined in the following way:

```
<xsd:complexType name="CT_Camera">
  <xsd:sequence>
    <xsd:element name="rot" type="CT_SphereCoords" minOccurs="0"
      maxOccurs="1" oxsd:dataStructure="optional" />
  </xsd:sequence>
  <xsd:attribute name="prst" type="ST_PresetCameraType" use="required"
    />
  <xsd:attribute name="fov" type="ST_FOVAngle" use="optional" />
  <xsd:attribute name="zoom" type="ST_PositivePercentage"
    use="optional" default="100000" />
</xsd:complexType>
```

The only complex type contained in the camera, CT_SphereCoords, is a type defined elsewhere within the DrawingML. There are three simple types associated with a camera:

- ST_FOVAngle (field of view angle), which is a positive angle between 0 and 180 in 60,000th of a degree.
- ST_PositivePercentage (zoom), which is defined as a percentage in 1,000th of a percent.
- ST_PresentCameraType (preset camera)

Figure 1 below shows some different presets applied to a shape.

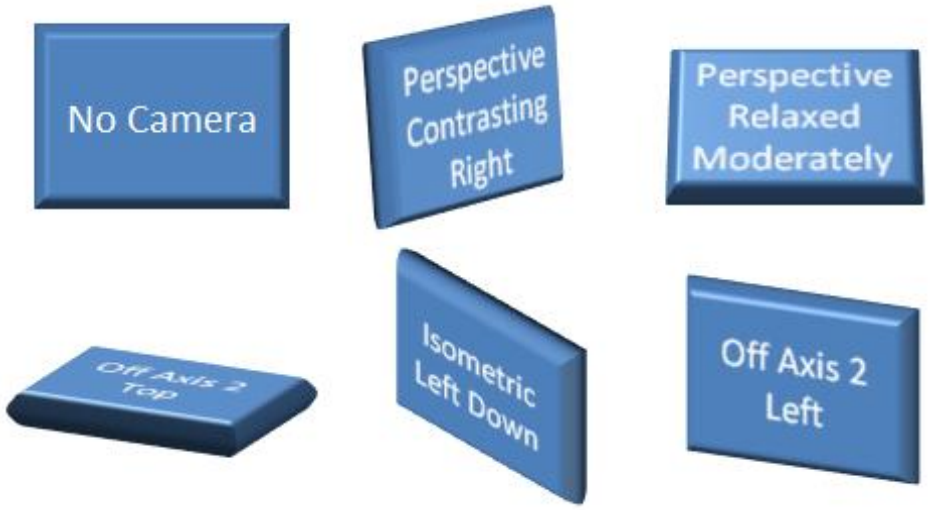


Figure 7: Different default cameras applied to a shape

The available options for ST_PresetCameraType are as follows:

legacyObliqueTopLeft
legacyObliqueTop
legacyObliqueLeft
legacyObliqueFront
legacyObliqueRight

legacyObliqueBottomLeft
legacyObliqueBottom
legacyObliqueBottomRight
legacyPerspectiveTopLeft
legacyPerspectiveTop

legacyPerspectiveTopRight
legacyPerspectiveLeft
legacyPerspectiveFront
legacyPerspectiveRight
legacyPerspectiveBottomLeft

legacyPerspectiveBottom
legacyPerspectiveBottomRight

orthographicFront
isomentricTopUp
isometricTopDown

isomentricBottomDown
isometricLeftUp
isometricLeftDown
isometricRightUp
isometricRightDown

isometricOffAxis1Left
isometricOffAxis1Right
isometricOffAxis1Top
isometricOffAxis2Left
isometricOffAxis2Right

isometricOffAxis2Top
isometricOffAxis3Left
isometricOffAxis3Right
isometricOffAxis3Bottom
isometricOffAxis4Left

isometricOffAxis4Right
isometricOffAxis4Bottom
obliqueTopLeft
obliqueTopRight
obliqueLeft

obliqueRight
obliqueBottomLeft
obliqueBottom
obliqueBottomRight
perspectiveFront

perspectiveLeft
perspectiveRight
perspectiveAbove
perspectiveBelow
perspectiveAboveLeftFacing

perspectiveAboveRightFacing
perspectiveContrastingRightFacing
perspectiveContrastingLeftFacing
perspectiveHeroicLeftFacing
perspectiveHeroicRightFacing

perspectiveHeroicExtremeLeftFacing
perspectiveHeroicExtremeRightFacing
perspectiveRelaxed
perspectiveRelaxedModerately

5.6.2.3 Light

The complex type, CT_LightRig, defines the lighting of the scene. A light rig consists of a preset direction, preset rig type, and a rotation that serves as an override for the direction. The complex type is defined as:

```
<xsd:complexType name="CT_LightRig">
  <xsd:sequence>
    <xsd:element name="rot" type="CT_SphereCoords" minOccurs="0"
      maxOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="rig" type="ST_LightRigType" use="required" />
  <xsd:attribute name="dir" type="ST_LightRigDirection" use="required"}
    />
</xsd:complexType>
```

Just as with the camera, the complex type, CT_SphereCoords, is defined elsewhere in the DrawingML. This element, however, serves as an override for the default light right direction. Figure 2 below shows some of the different preset lights applied to a shape.

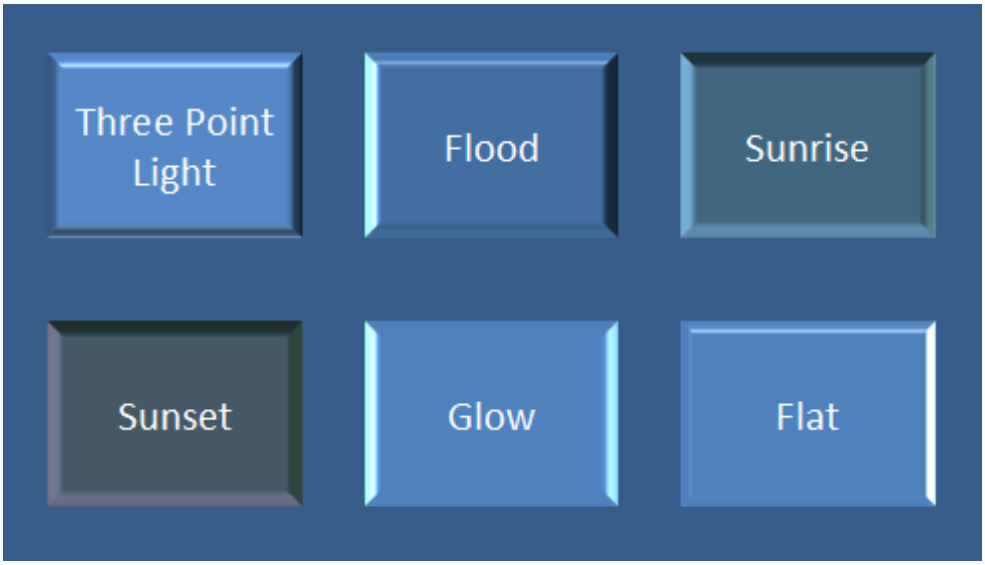


Figure 8: Some preset lights applied to a shape.

The types of available light rigs are:

legacyFlat1
legacyFlat2
legacyFlat3

legacyFlat4
legacyNormal1

legacyNormal2
legacyNormal3
legacyNormal4
legacyHarsh1
legacyHarsh2

legacyHarsh3
legacyHarsh4
threePoint
balanced
soft

harsh
flood
contrasting
morning
sunrise

sunset
chilly
freezing
flat
twoPoint
glow
brightRoom

The types of available present directions are:

tl – top left
t – top
tr – top right
l – left

r – right
bl – bottom left
b – bottom
br – bottom right

5.6.2.4 Backdrop

The complex type, CT_Backdrop, defines a unique place in the 3-D scene. The backdrop is a flat 2-D plane that can hold effects, such as shadows, oriented in 3-D space. The points and vectors contained within the backdrop are relative to world space. The complex type is defined as:

```
<xsd:complexType name="CT_Backdrop">
  <xsd:sequence>
    <xsd:element name="anchor" type="CT_Point3D" minOccurs="1"
      maxOccurs="1" />
    <xsd:element name="norm" type="CT_Vector3D" minOccurs="1"
      maxOccurs="1" />
    <xsd:element name="up" type="CT_Vector3D" minOccurs="1"
      maxOccurs="1"/>
    <xsd:element name="ext" type="CT_OfficeArtExtension" minOccurs="0"
      maxOccurs="1" />
  </xsd:sequence>
</xsd:complexType>
```

All of the complex types defined within this backdrop are defined elsewhere in DrawingML. As with other complex types, the backdrop also contains an element reserved for future extensibility.

5.6.3 Styles

The 3-D styles section encompasses the properties for 3-D shapes. These properties are those that get applied to the 3-D shape, such as a bevel or a contour, and they define the look of the shape in 3-D. A number of simple types used within the complex types of this group are defined below.

5.6.3.1 Simple Types

The simple types defined here outline the different presets available to the user. These presets are applied to the shapes through the complex type definitions that are outlined later.

5.6.3.1.1 Bevel Type

The simple type, ST_BevelPresetType, defines a preset bevel for a shape and some examples can be seen in figure 3.

Figure 9: Different bevel types applied to a shape

The different types of bevels available are:

relaxedInset
circle
slope
cross
angle
softRound

convex
coolSlant
divot
riblet
hardedge
artDeco

5.6.3.1.2 Preset Material Type

The simple type, ST_PresetMaterialType, defines a material for the shape. The material properties describe the surface appearance of the shape, and are used in lighting calculations to define exactly how the light interacts with the shape. Some example material types can be seen in figure 4.

Figure 10: Different material types on a shape.

The different preset material types are:

legacyMatte
legacyPlastic
legacyMetal
legacyWireframe
matte

plastic
metal
warmMatte
translucentPowder
powder

dkEdge
softEdge
clear
flat
softMetal

5.6.3.2 Complex Types

The complex types in this area define the actual 3-D properties that get applied to a shape. These properties work together in order to define the geometry of a shape along with the scene related properties that define the look of the geometry of the shape.

5.6.3.2.1 Bevel

The complex type, CT_Bevel, defines a bevel for a shape. The bevel consists of a width and a height value, along with a preset bevel. The complex type is defined in the following manner:

```
<xsd:complexType name="CT_Bevel">
  <xsd:attribute name="w" type="ST_PositiveCoordinate" use="optional"
    default="76200" />
  <xsd:attribute name="h" type="ST_PositiveCoordinate" use="optional"
    default="76200" />
  <xsd:attribute name="prst" type="ST_BevelPresetType" use="optional"
    default="circle" />
</xsd:complexType>
```

5.6.3.2.2 Shape 3-D

The complex type, CT_Shape3D, defines all of the 3-D properties associated with an individual shape. A shape can have two bevels, one on the top and one on the bottom. An extrusion color also defined, which, when applied, applies a color to the surface of the extrusion. There is also an extrusion width, which defines the width of the extrusion. A contour color and width can be defined for the shape. A z-axis anchor is defined within the complex type and is the anchor relative to the shape's top face. The shape 3-D complex type also holds a present material. Finally the shape 3-D contains another element just as in previous complex types, which is used for future extensibility. The CT_Shape3D complex type is defined in the following manner:

```
<xsd:complexType name="CT_Shape3D">
  <xsd:sequence>
    <xsd:element name="bevelT" type="CT_Bevel" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="bevelB" type="CT_Bevel" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="extrusionClr" type="CT_Color" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="contourClr" type="CT_Color" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="ext" type="CT_OfficeArtExtension" minOccurs="0"
      maxOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="z" type="ST_Coordinate" use="optional"
    default="0" />
  <xsd:attribute name="extrusionH" type="ST_PositiveCoordinate"
    use="optional" default="0" />
  <xsd:attribute name="contourW" type="ST_PositiveCoordinate"
    use="optional" default="0" />
  <xsd:attribute name="prstMaterial" type="ST_PresetMaterialType"
    use="optional" default="warmMatte" />
</xsd:complexType>
```

5.6.3.2.3 Flat Text

The complex type, CT_FlatText, defines a text object in a 3-D scene that should be rendered as a normal, flat, text overlay outside of the 3-D scene. The complex type is defined in the following manner:

```
<xsd:complexType name="CT_FlatText">
  <xsd:attribute name="z" type="ST_Coordinate" use="optional"
    default="0" />
</xsd:complexType>
```

5.6.3.2.4 Group, Text 3-D

The final structure to be defined is a group, EG_Text3D, which describes how text should be applied in the 3-D scene. If the text object is a member of the 3-D scene, then there are three different ways it can be displayed:

- If no EG_Text3D choice is provided, the text will be rendered in a scene coherent manner and will be rendered in perspective inside of the 3D scene as a planar shape inside the 3-D.
- If CT_Shape3D is provided then the text will be scene coherent and fully 3-D.
- If CT_FlatText is provided then the text will be drawn as normal 2-D text rendered on top of the 3-D scene.

An EG_Text3D is defined in the following manner:

```
<xsd:group name="EG_Text3D">
  <xsd:choice oxsd:cname="Text3DChoice"
    oxsd:cnameMember="text3DChoice">
    <xsd:element name="sp3dtype="CT_Shape3D" minOccurs="1"
      maxOccurs="1/>
    <xsd:element name="flatTx" type="CT_FlatText" minOccurs="1"
      maxOccurs="1" />
  </xsd:choice>
</xsd:group>
```

5.7 Coordinate Systems and Transformations

5.7.1 Introduction

This document provides an overview of the transformation elements for shapes and groups, represented by <a:xfrm>in dml-baseTypes.xsd. These schemas are for the representation of scaling and rotation on individual shapes and groups.

§5.7.2, §5.7.3, and §5.7.4 provide a qualitative overview of the transformation pipeline. §5.7.6 provides mathematical details.

5.7.2 Coordinate System

All DrawingML shapes are located on a 2-D Cartesian coordinate space with the origin (0,0) in the upper left-hand corner of the canvas. The x-axis coordinates grow positively as one moves from left to right, and the y-axis coordinates grow positively as one moves from top to bottom.

Coordinates are measured in EMUs (914400 EMUs per inch), and can be positive or negative.

5.7.3 Shape Transformations

In this subclause, we describe the transformation pipeline for a shape. To summarize, the *shape transformation* for a shape is defined as the following sequence of operations:

1. The translation and scaling required to transform its original bounding box to a rectangle specified by the offset and extents.
2. A flip across the center of the bounding box according to flipH and flipV.
3. A rotation about the center of the bounding box according to the rot attribute.

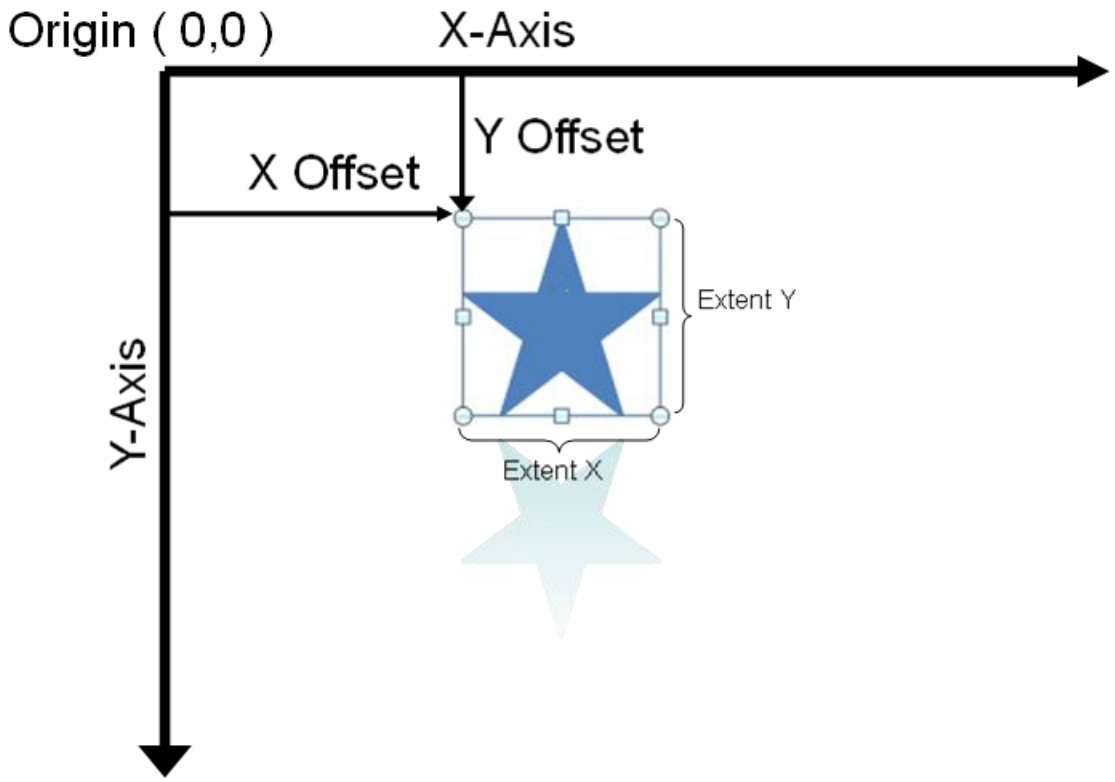
To render a shape that is not inside a group (§5.7.4), the renderer simply applies the shape transformation to the original shape.

5.7.3.1 Scaling and Translating a Shape

The shape is scaled horizontally, scaled vertically, and translated in both dimensions, to fill a given bounding box. The bounding box is represented by specifying an offset in x and y (attributes x and y of a:off) and extents in x and y (attributes cx and cy of a:ext, both of which must be greater than or equal to zero). The upper left

corner of the bounding box is located at the offset, and the lower right corner of the bounding box is located at the offset plus extent.

If the starting shape has zero width (e.g., it is a vertical line), then the cx attribute of a:ext is ignored and the horizontal scaling is skipped. Similarly, if the starting shape has zero height, then the cy attribute of a:ext is ignored and the vertical scaling is skipped.



The following XML fragment represents the offset and extents for the star shape above:

```
<a:xfrm>
  <a:off x="1866680" y="990600"/>
  <a:ext cx="1371600" cy="1371600"/>
</a:xfrm>
```

Notice that as demonstrated with the example above, any effects attached to the shape are disregarded when scaling and translating the shape to fill the given bounding box.

This example illustrates that no additional parameters are needed to represent the scaling of a shape. The bounding-box parameters are sufficient to represent scaling. The following XML Fragments represent the offset and extents for a star shape, before and after scaling. In this particular example, the bounding boxes have been chosen to have the same upper-left corner, i.e., the same offset.

Before scaling (small star):

```
<a:xfrm>
  <a:off x="1066800" y="990600"/>
  <a:ext cx="1371600" cy="1371600"/>
</a:xfrm>
```

After scaling (large star):

```
<a:xfrm>
  <a:off x="1066800" y="990600"/>
  <a:ext cx="2438400" cy="2133600"/>
</a:xfrm>
```

5.7.3.2 Rotating a Shape

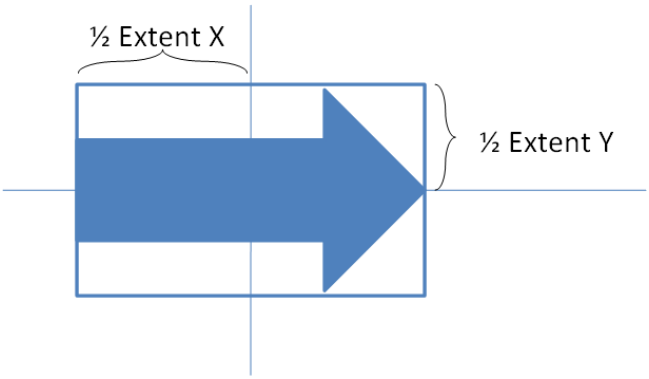
Rotation is represented with the rot attribute. The shape is rotated clockwise about the bounding-box center, by the amount specified in the rot attribute. Each unit of rotation is 1/1,000 of an arc minute (1/60,000 of a degree).

This example represents the small star from above, with a subsequent 45-degree rotation clockwise. Since the y axis points down, a clockwise rotation is positive.

```
<a:xfrm rot="2700000">
  <a:off x="1066800" y="990600"/>
  <a:ext cx="1371600" cy="1371600"/>
</a:xfrm>
```

5.7.3.3 Flipping a Shape

A flip is a reflection across a vertical or horizontal line that intersects the center of the bounding box. The optional flipH and flipV attributes control horizontal and vertical flipping, respectively. Each is absent or equal to `0` if no flipping is to be performed, and equal to `1` if flipping is to be performed.



The following XML fragment illustrates a shape that has been flipped both horizontally and vertically.

```
<a:xfrm flipH="1" flipV="1">
  <a:off x="3964937" y="2652643"/>
  <a:ext cx="168838" cy="1219199"/>
</a:xfrm>
```

5.7.4 Group Transformations

A group is composed of zero to many shapes. Because a group is a shape, this composition relationship can nest recursively. (A group with zero shapes is degenerate; it produces no user-visible output. A group with one shape is also degenerate; it has no representational power beyond that of the one shape.)

The definition of a group transformation is identical to that of a shape transformation, except that in place of the pre-transform bounding box of a shape, we use the union of all of its children prior to their individual rotations. To summarize, a *group transformation* is the following sequence of operations:

1. The translation and scaling required to transform the union of the children's bounding boxes to a rectangle defined by the group's offset and extent attributes.
2. A flipped about that bounding box according to the flipH and flipV attributes.
3. A rotation about the center of that bounding box according to the rot attribute.

To render a simple shape that is inside a group hierarchy, the renderer does not simply apply the shape transformation and all parent group transformations to the original shape. Instead (see §5.7.5), it applies the transformation equal to the following sequence of operations:

1. Horizontal scaling and flipping by a factor equal to the product of the horizontal scalings and flips in its own transformation and those of its parents.
2. Vertical scaling and flipping by a factor equal to the product of the vertical scalings and flips in its own transformation and those of its parents.
3. Rotation by an amount equal to the sum of the rotations in its own transformation and those of its parents.
4. Translation such that its center coincides with the point obtained by applying the shape transformation and all parent group transformations to the shape's original center.

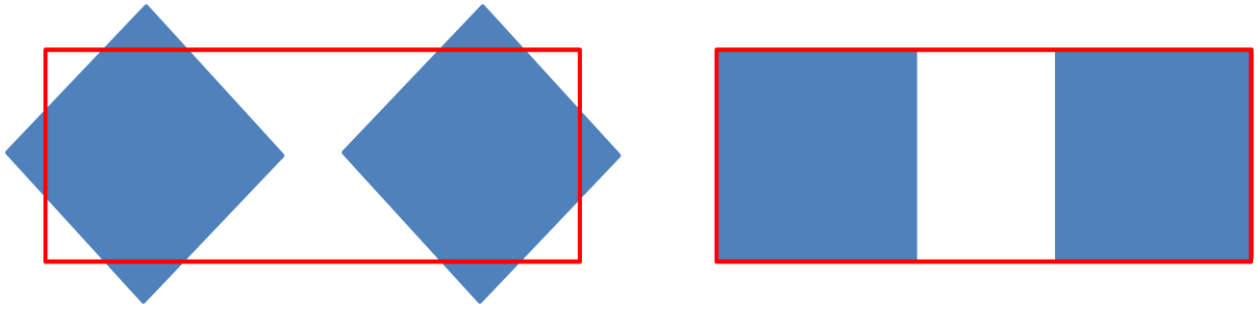
Because of the similarity with the transformation pipeline for a shape, the forthcoming subsections primarily cover illustrative examples.

5.7.4.1 Scaling and Translating a Group

The group is scaled horizontally, scaled vertically, and translated in both dimensions. The parameters are chosen to transform the child bounding box (specified by a:chOff and a:chExt) to the group bounding box (specified by a:off and a:ext). The *child bounding box* is defined as the bounding box around the group's children as they would have been had their rot attributes been absent.

It is possible for the child bounding box to have a zero value for cx or cy in a:chExt, e.g., because the starting shape is a horizontal or vertical line, or because the starting shape was scaled to have zero width or height. Such a case is handled in the same way as previously described for simple shapes.

This example demonstrates the definition of the child bounding box. The two shapes on the left, rotated squares, are grouped. The two shapes on the right, non-rotated squares, are also grouped.



The red bands are not part of the drawing; each represents the child bounding box of a group. In the XML fragments, the child bounding boxes have identical y values, illustrating that they are computed based on the bounding boxes of the squares prior to their rotation.

For the left-hand group:

```
<a:xfrm>
  <a:off x="762000" y="1828800" />
  <a:ext cx="3327400" cy="1219200" />
  <a:chOff x="762000" y="1828800" />
  <a:chExt cx="3327400" cy="1219200" />
</a:xfrm>
```

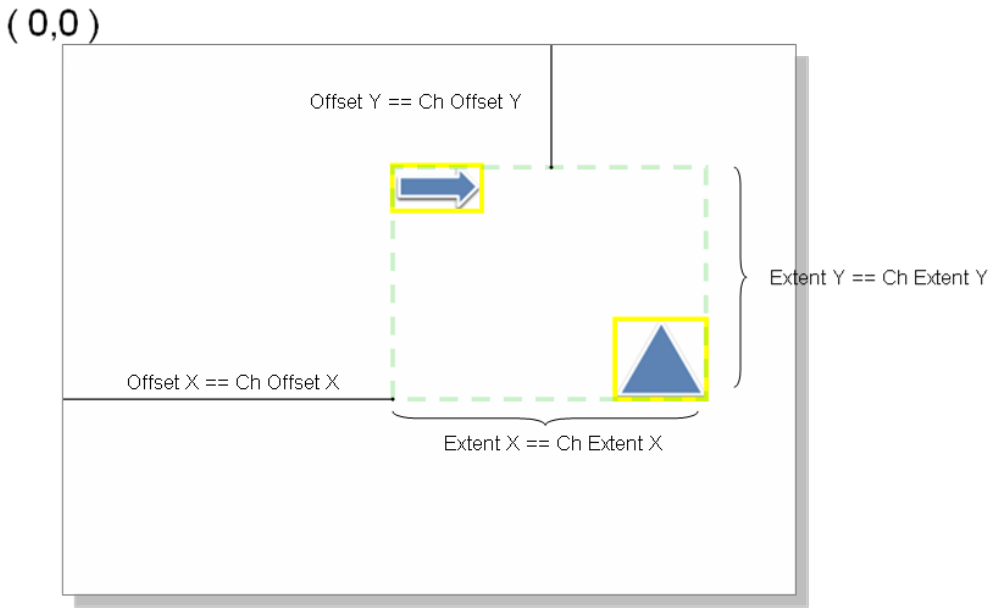
For the right-hand group:

```
<a:xfrm>
  <a:off x="4978400" y="1828800" />
  <a:ext cx="3327400" cy="1219200" />
  <a:chOff x="4978400" y="1828800" />
  <a:chExt cx="3327400" cy="1219200" />
</a:xfrm>
```

The remainder of the examples in this subsection illustrate translation and scaling of a group.

In this situation, two shapes are grouped: an arrow and a triangle. No further translation, scaling, rotation, or flipping is applied.

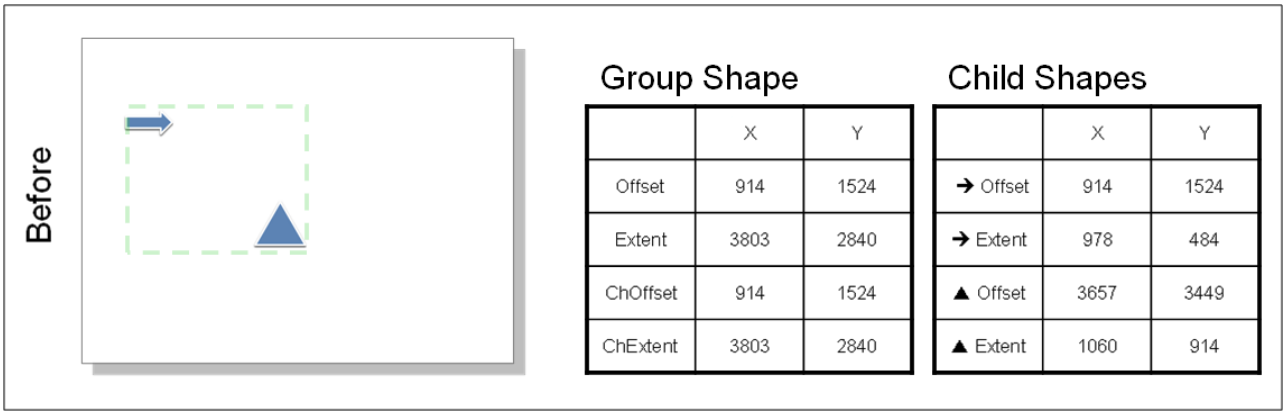
To represent this situation, the child bounding box is the bounding box around both of these shapes; and because no further transformation is applied, the group bounding box is equal to the child bounding box.



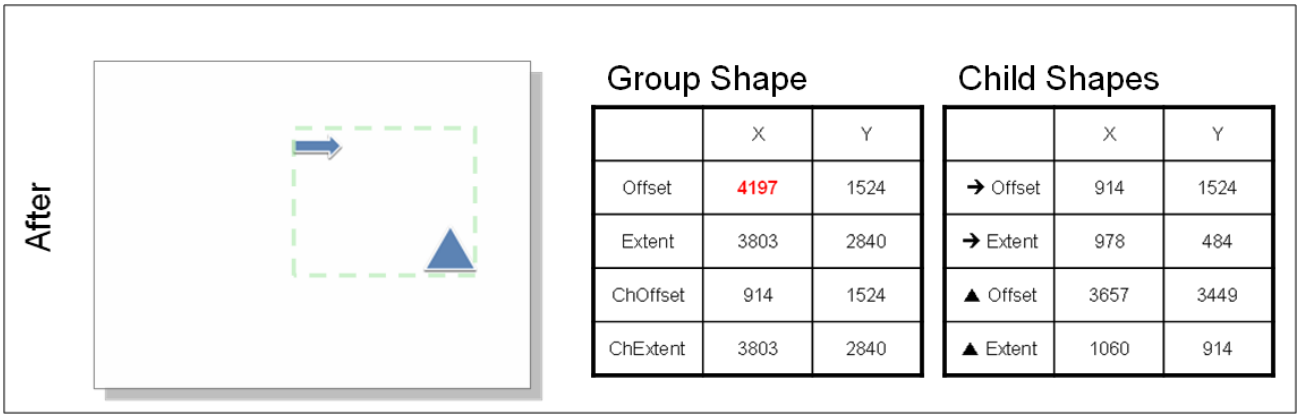
The following is an XML snippet representing the transform variables of the group.

```
<p:grpSpPr>
  <a:xfrm>
    <a:off x="2209800" y="2514600"/>
    <a:ext cx="4038600" cy="2286000"/>
    <a:chOff x="2209800" y="2514600"/>
    <a:chExt cx="4038600" cy="2286000"/>
  </a:xfrm>
</p:grpSpPr>
```

This example illustrates that no additional parameters are needed to represent the translation of a group. A group is moved to the right. The following diagram shows the starting state, prior to the translation. Note that offset==ChOffset and extent==ChExtent.



Increasing the x component of the offset moves the group to the right.

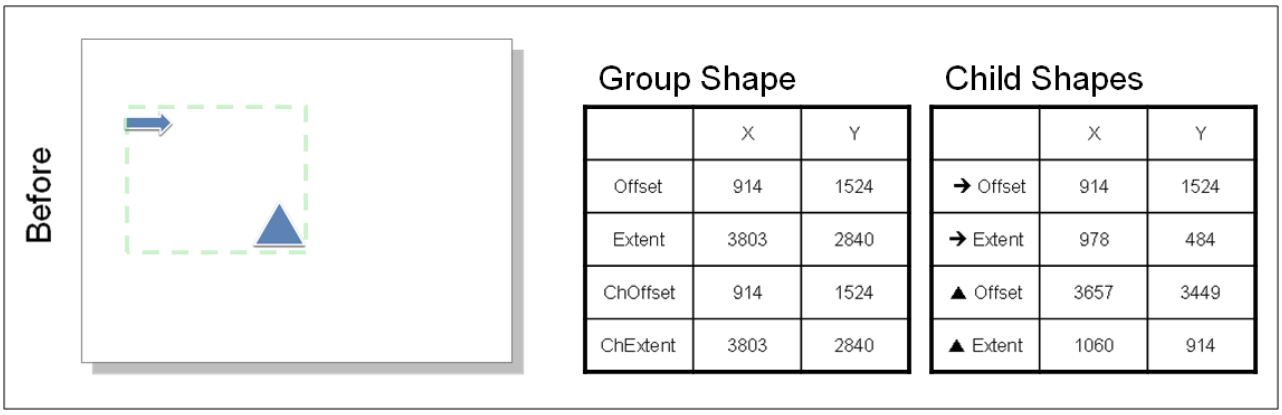


Similarly, scaling can be performed by adjusting the group bounding box.

5.7.4.2 Rotating a Group

Group rotation is identical to shape rotation. The group is rotated clockwise about the bounding-box center, by the amount specified in the rot attribute.

In this example, group is rotated 90 degrees clockwise. The following diagram shows the starting state, prior to the rotation.



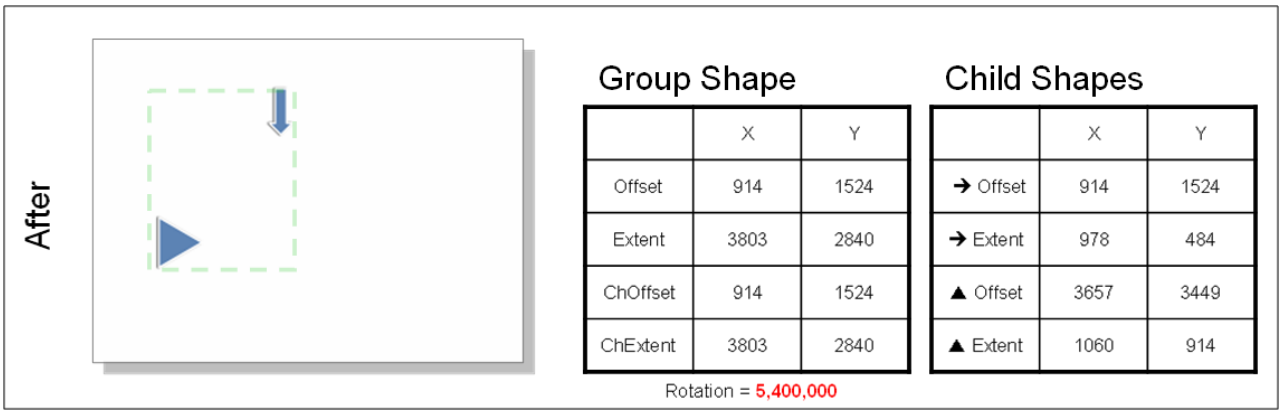
Group Shape

	X	Y
Offset	914	1524
Extent	3803	2840
ChOffset	914	1524
ChExtent	3803	2840

Child Shapes

	X	Y
→ Offset	914	1524
→ Extent	978	484
▲ Offset	3657	3449
▲ Extent	1060	914

Setting the rotation attribute to 5,400,000 rotates the group clockwise 90 degrees.



Group Shape

	X	Y
Offset	914	1524
Extent	3803	2840
ChOffset	914	1524
ChExtent	3803	2840

Child Shapes

	X	Y
→ Offset	914	1524
→ Extent	978	484
▲ Offset	3657	3449
▲ Extent	1060	914

Rotation = 5,400,000

5.7.5 Nesting Transformations

The following example illustrates the rendering procedure described at the end of the introduction to §5.7.4, which differs from a conventional transformation pipeline in any case where a scaled group contains a rotated child.

[Example: In this example, the diagram on the left side is a group that comprises a rotated red square centered inside a non-rotated blue square. The diagram on the right side is the same group, scaled horizontally. The red square scales along an axis parallel to its own edges instead of an axis parallel to the edges of the blue square.

5.7.6 Transformation Matrices

The preceding sections fully define the transformation pipeline and its parameters. This section assists developers in implementing the pipeline by describing it mathematically. It is generalized to describe either the shape transformation pipeline or the group transformation pipeline.

5.7.6.1 Symbol Definitions

Let the following symbols represent parameters described in the preceding sections.

(B_x, B_y)	For a shape: upper left corner of untransformed shape. For a group: upper left corner of child bounding box (a:chOff).
(D_x, D_y)	For a shape: (width,height) of untransformed shape. For a group: (width,height) of child bounding box (a:chExt).
(B'_x, B'_y)	Upper left corner of bounding box (a:off), prior to rotation and flip.
(D'_x, D'_y)	(width,height) of bounding box (a:ext), prior to rotation and flip.
θ	Clockwise rotation (from the attribute rot, which uses thousandths of an arc minute).
F_x	−1 if flipH is true; +1 otherwise.
F_y	−1 if flipV is true; +1 otherwise.

We use homogeneous coordinates, in which a point p is represented in the form

$$p = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} .$$

We use the convention in which transformations are applied by left-hand multiplication. Thus, to obtain the point p' by applying transformation T to point p we write:

$$p' = T\,p$$

5.7.6.2 Transformation Pipeline

The entire transformation pipeline that defines either a shape transformation or a group transformation is represented by the matrix T, defined at the end of this subsection.

Scaling and translation are produced by the following matrix:

$$T_{st} = \begin{bmatrix} \frac{D'_x}{D_x} & 0 & B'_x - \left(\frac{D'_x}{D_x}\right) B_x \\ 0 & \frac{D'_y}{D_y} & B'_y - \left(\frac{D'_y}{D_y}\right) B_y \\ 0 & 0 & 1 \end{bmatrix} .$$

The following matrix translates the bounding box to the origin in preparation for rotation and flipping:

$$U = \begin{bmatrix} 1 & 0 & -\left(B'_x + \frac{D'_x}{2}\right) \\ 0 & 1 & -\left(B'_y + \frac{D'_y}{2}\right) \\ 0 & 0 & 1 \end{bmatrix} .$$

Rotation and flipping are produced by the following matrix:

$$T_{rf} = U^{-1} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_x & 0 & 0 \\ 0 & F_y & 0 \\ 0 & 0 & 1 \end{bmatrix} U .$$

The entire transformation pipeline for one step in the group hierarchy is represented by the matrix

$$T = T_{rf} T_{st}$$

5.8 Shape Properties and Effects

5.8.1 Introduction

This subclause provides a high-level overview of the Fill Properties, Line Properties, and Effect Properties described in the following schemas: dml-shapeLineProperties.xsd, dml-shapeEffects.xsd, dml-baseTypes.xsd.

Color Models are also covered, as Fills, Lines, and Effects all reference color model schemas to represent color.

5.8.2 Color Models

There are several methods of expressing color: scrgbClr, srgbClr, hslClr, sysClr, schemeClr, and prstClr. Although srgbClr is the most commonly used model, the rationale for having various equivalent color models stems from a desire to have different ways of naturally expressing a color choice.

5.8.2.1 scrgbClr

scrgbClr is a legacy form of expressing Red, Green, Blue color. Values are expressed in Percentages. r, g, and b are all required and correspond to red, green, and blue, respectively.

```
<a:scrgbClr r="10000" g="20000" b="30000">
```

5.8.2.2 srgbClr

srgbClr is similar to scrgbClr with the exception that instead of expressing the values as percentages, they are specified using two hex digits per color, in the order RGB.

```
<a:srgbClr val="FFFF00">
```

5.8.2.3 hslClr

hslClr represents a color using the Hue, Saturation, and Luminescence color model. Values are expressed in Percentages. h, s, and l are all required, and correspond to hue, saturation, and luminescence respectively. A perceptual gamma of 2.2 is assumed.

```
<a:hslClr h="10000" s="20000" l="30000">
```

5.8.2.4 sysClr

sysClr represents a system color, and introduces a level of indirection. For example, specifying:

```
<a:sysClr val="windowText">
```

binds the color to be the color chosen in the system for "Window Text". The possible values are:

```
scrollBar
background
activeCaption
inactiveCaption
menu
window
windowFrame
menuText
windowText
captionText
activeBorder
inactiveBorder
appWorkspace
highlight
highlightText
btnFace
btnShadow
grayText
btnText
inactiveCaptionText
btnHighlight
3dDkShadow
3dLight
infoText
infoBk
hotLight
gradientActiveCaption
gradientInactiveCaption
menuHighlight
menuBar
```

5.8.2.5 schemeClr

schemeClr represents a color from a theme. The color changes if theme bindings change. For example, specifying:

```
<a:schemeClr val="lt1">
```

binds the color to be Light 1 color of the current theme. The possible values are:

bg1	semantic background color
bg1	semantic background color
tx1	semantic text color
bg2	semantic additional background color
tx2	semantic additional text color
accent1	extra scheme color 1
accent2	extra scheme color 2
accent3	extra scheme color 3
accent4	extra scheme color 4
accent5	extra scheme color 5
accent6	extra scheme color 6
hlink	hyperlink color
folHlink	followed hyperlink color
dk1	main dark color 1
lt1	Main light color 1
phClr	a color used in theme definitions which means "use the color of the style"
dk2	main dark color 2
lt2	main light color 2

5.8.2.6 prstClr

prstClr represents a preset color. This is a legacy definition of colors which is no longer currently used. A preset color is a choice from among several presets provided in older versions of Office.

```
<a:prstClr val="black">
```

The selected color is "black". Valid values for this setting are:

```
aliceBlue
antiqueWhite
aqua
aquamarine
azure
beige
bisque
black
blanchedAlmond
blue
blueViolet
brown
burlyWood
cadetBlue
chartreuse
chocolate
coral
cornflowerBlue
cornsilk
crimson
cyan
dkBlue
dkCyan
dkGoldenrod
dkGray
dkGreen
dkKhaki
dkMagenta
dkOliveGreen
dkOrange
dkOrchid
dkRed
dkSalmon
dkSeaGreen
dkSlateBlue
dkSlateGray
dkTurquoise
dkViolet
deepPink
deepSkyBlue
```

```
dimGray
dodgerBlue
firebrick
floralWhite
forestGreen
fuchsia
gainsboro
ghostWhite
gold
goldenrod
gray
green
greenYellow
honeydew
hotPink
indianRed
indigo
ivory
khaki
lavender
lavenderBlush
lawnGreen
lemonChiffon
ltBlue
ltCoral
ltCyan
ltGoldenrodYellow
ltGray
ltGreen
ltPink
ltSalmon
ltSeaGreen
ltSkyBlue
ltSlateGray
ltSteelBlue
ltYellow
lime
limeGreen
linen
magenta
```

```
maroon
medAquamarine
medBlue
medOrchid
medPurple
medSeaGreen
medSlateBlue
medSpringGreen
medTurquoise
medVioletRed
midnightBlue
mintCream
mistyRose
moccasin
navajoWhite
navy
oldLace
olive
oliveDrab
orange
orangeRed
orchid
paleGoldenrod
paleGreen
paleTurquoise
paleVioletRed
papayaWhip
peachPuff
peru
pink
plum
powderBlue
purple
red
rosyBrown
royalBlue
saddleBrown
salmon
sandyBrown
seaGreen
```

```
seaShell
sienna
silver
skyBlue
slateBlue
slateGray
snow
springGreen
steelBlue
tan
teal
thistle
tomato
transparent
turquoise
violet
wheat
white
whiteSmoke
yellow
yellowGreen
```

5.8.3 Color Transforms

A color transform is a modification to related properties of an underlying color. For example, transparency is a property that is related to color. Color transforms are specified as child tags off any color model's tag.

```
<a:solidFill>
  <a:srgbClr val="00B050">
    <a:alpha val="51000"/>
  </a:srgbClr>
</a:solidFill>
```

The following are the allowed color transforms and descriptions of the transformations they apply:

- tint: Yields a lighter version of its input color. A 10% tint is 10% of the input color combined with 90% white.
- shade: Yields a darker version of its input color. A 10% shade is 10% of the input color combined with 90% black.
- comp: Yields the complement of its input color. For example, the complement of red is green.
- inv: Yields the inverse of its input color. For example, the inverse of red (1,0,0) is cyan (0,1,1).
- gray: Yields a grayscale of its input color, taking into relative intensities of the red, green, and blue primaries.
- alpha: Yields its input color with the specified opacity, but with its color unchanged.

- alphaOff: Yields a more or less opaque version of its input color. An alpha offset never increases the alpha beyond 100% or decreases below 0%; i.e., the result of the transform pins the alpha to the range of [0%,100%]. A 10% alpha offset increases a 50% opacity to 60%. A -10% alpha offset decreases a 50% opacity to 40%.
- alphaMod: Yields a more or less opaque version of its input color. An alpha modulate never increases the alpha beyond 100%. A 200% alpha modulate makes a input color twice as opaque as before. A 50% alpha modulate makes a input color half as opaque as before.
- hue: Yields the input color with the specified hue, but with its saturation and luminance unchanged.
- hueOff: Yields the input color with its hue shifted, but with its saturation and luminance unchanged.
- hueMod: Yields the input color with its hue modulated by the given percentage.
- sat: Yields the input color with the specified saturation, but with its hue and luminance unchanged. Typically saturation values fall in the range [0%, 100%].
- satOff: Yields the input color with its saturation shifted, but with its hue and luminance unchanged.
- satMod: Yields the input color with its saturation modulated by the given percentage. A 50% saturation modulate will reduce the saturation by half. A 200% saturation modulate will double the saturation.
- lum: Yields the input color with the specified luminance, but with its hue and saturation unchanged. Typically, luminance values fall in the range [0%,100%].
- lumOff: Yields the input color with its luminance shifted, but with its hue and saturation unchanged.
- lumMod: Yields the input color with its luminance modulated by the given percentage. A 50% luminance modulate will reduce the luminance by half. A 200% luminance modulate will double the luminance.
- red: Yields the input color with the specified red component, but with its green and blue components unchanged.
- redOff: Yields the input color with its red component shifted, but with its green and blue components unchanged.
- redMod: Yields the input color with its red component modulated by the given percentage. A 50% red modulate will reduce the red component by half. A 200% red modulate will double the red component.
- green: Yields the input color with the specified green component, but with its red and blue components unchanged.
- greenOff: Yields the input color with its green component shifted, but with its red and blue components unchanged.
- greenMod: Yields the input color with its green component modulated by the given percentage. A 50% green modulate will reduce the green component by half. A 200% green modulate will double the green component.
- blue: Yields the input color with the specified blue component, but with its red and green components unchanged.
- blueOff: Yields the input color with its blue component shifted, but with its red and green components unchanged.

- blueMod: Yields the input color with its blue component modulated by the given percentage. A 50% blue modulate will reduce the blue component by half. A 200% blue modulate will double the blue component.
- gamma: Yields the sRGB gamma shift of its input color.
- invGamma: Yields the inverse sRGB gamma shift of its input color.

5.8.4 Fills

There are six types of fills:

- No Fill
- Solid Fill
- Gradient Fill
- Blip Fill
- Pattern Fill
- Group Fill

The se types describe the general structure of all fills; however, not all fills are permitted in all locations. For example, Blip Fills and Group Fills are not permitted on lines.

5.8.4.1 Solid Fills

```
<p:sp>
  <p:nvSpPr>...
  <p:spPr>
    <a:xfrm>
      <a:off x="5410200" y="2438400"/>
      <a:ext cx="2895600" cy="304800"/>
    </a:xfrm>
    <a:prstGeom prst="rect">
      <a:avLst/>
    </a:prstGeom>
    <a:solidFill>
      <a:srgbClr val="FFFF00"/>
    </a:solidFill>
  </p:spPr>
  <p:style>...
  <p:txBody>...
</p:sp>
```

A solid fill specifies a single color, using any valid color model

5.8.4.2 Gradient Fills

```
<a:gradFill>
  <a:gsLst>
    <a:gs pos="69000">
      <a:schemeClr val="accent1"/>
    </a:gs>
    <a:gs pos="0">
      <a:scrgbClr r="0" g="0" b="0"/>
    </a:gs>
  </a:gsLst>
  <a:lin ang="2700000" scaled="1"/>
</a:gradFill>
```

Gradient Stop List

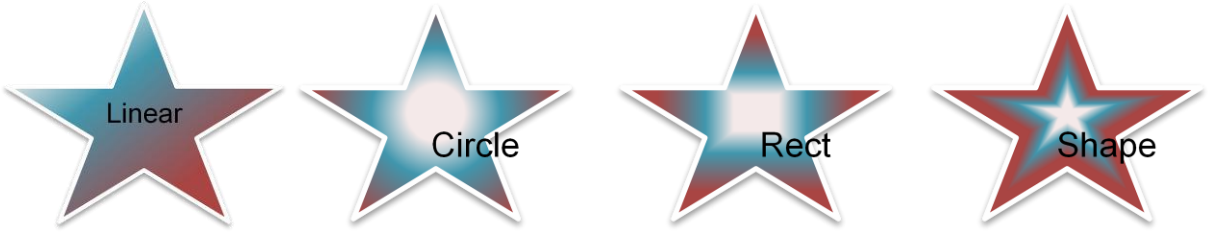
Gradient Fills consist of three elements: a list of gradient stops, a shading specification, as well as some attributes.

Two attributes are available on gradient fills.

- flip specifies how to flip a tile region when using it to fill a larger fill region.
- rotWithShape specifies whether the fill rotates along with a shape when the shape is rotated.

A gradient stop list is a list of locations and colors that make up the gradient fill. Positions are specified as percentages.

The shading specification specifies the two possible kinds of gradient fills: linear, or path based. A linear fill follows a straight-line direction as specified by the angle of the line. A path-based fill follows the contours of a well-defined path (such as a shape, circle, or rectangle).



5.8.4.3 Blip Fills

```
<p:blipFill>
  <a:blip r:embed="rId4" r:link=""/>
  <a:srcRect l="11000" t="14000"
             r="20000" b="28667"/>
  <a:stretch>
    <a:fillRect/>
  </a:stretch>
</p:blipFill>
```

BLIPs refer to Binary Large Image or Pictures. Blip Fills are made up of several components: a Blip Reference, a Source Rectangle, and a Fill Mode.

The Blip reference, a:blip, is the main reference to the blip content itself. A reference ID serves as the main link with an attribute allowed to specify compression level of the blip (one of: email, screen, print, hqprint, none). A Blip Effect may optionally be specified to indicate a modification of the raw blip content. Valid Blip Effects are:

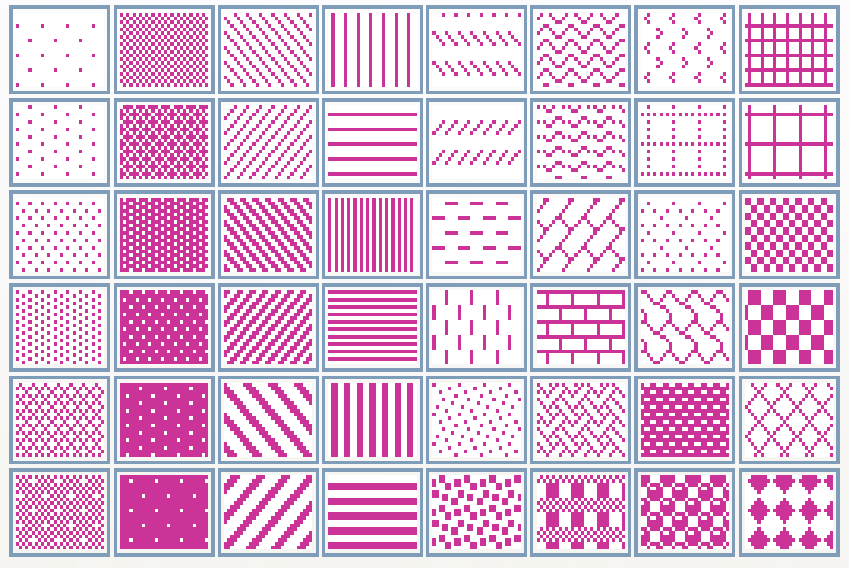
- alphaBiLevel
- alphaCeiling
- alphaFloor
- alphaInv
- alphaMod
- alphaModFix
- alphaRepl
- biLevel
- blur
- clrChange
- clrRepl
- duotone
- fillOverlay
- grayscl
- hsl
- lum
- tint

The blip effects mirror the color transformations (see the descriptions in the color transformations subclause for descriptions of Blip Effects).

A Source Rectangle, a:srcRect, is used to implement image cropping, and indicates the rectangular window of content which is of interest.

Finally, two fill modes are possible, tiling, and stretching. This indicates the behavior to be performed when the user resizes an image to an area larger than the source rectangle. Tiling 'tiles' an image so that image content is simply duplicated while stretching scales the source rectangle content to fill the bounds of the fillRect (bounding box of blip filled shape).

5.8.4.4 Pattern Fills



Pattern Fills are legacy Office 11 fills which consist of a Foreground Color, a Background Color and a preset pattern value. Possible pattern values are:

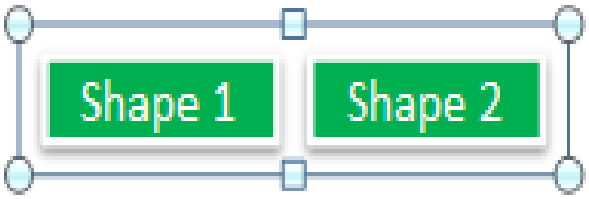
```
pct5
pct10
pct20
pct25
pct30
pct40
pct50
pct60
pct70
pct75
pct80
pct90
horz
vert
ltHorz
```

```
ltVert
dkHorz
dkVert
narHorz
narVert
dashHorz
dashVert
cross
dnDiag
upDiag
ltDnDiag
ltUpDiag
dkDnDiag
dkUpDiag
wdDnDiag
wdUpDiag
dashDnDiag
dashUpDiag
diagCross
smCheck
lgCheck
smGrid
lgGrid
dotGrid
smConfetti
lgConfetti
horzBrick
diagBrick
solidDmnd
openDmnd
dotDmnd
plaid
sphere
weave
divot
shingle
wave
trellis
zigZag
```

5.8.4.5 Group Fills

When objects are grouped together, a group fill is a convenient structure for indicating that the fill properties of any individual element inherits from the fill properties of parent group.

```
<p:grpSp>
  <p:nvGrpSpPr>...
  <p:grpSpPr>
    <a:xfrm>...
    <a:solidFill>
      <a:srgbClr val="00B050"/>
    </a:solidFill>
  </p:grpSpPr>
  <p:sp>
    <p:nvSpPr>...
    <p:spPr>
      <a:xfrm>...
      <a:prstGeom prst="rect">...
      <a:grpFill/>
    </p:spPr>
    <p:style>...
    <p:txBody>...
  </p:sp>
  <p:sp>
    <p:nvSpPr>...
    <p:spPr>
      <a:xfrm>...
      <a:prstGeom prst="rect">...
      <a:grpFill/>
    </p:spPr>
    <p:style>...
    <p:txBody>...
  </p:sp>
</p:grpSp>
```



5.8.5 Line Properties

While it is obvious that line properties, a:ln are used to represent properties for lines, what may be less obvious is where this structure can appear. Lines aren't just for lines-- just about any object can have a line property-- usually referring to the outlines that are possible on shapes, pictures, or text. Lines used in this context also yield additional characteristics we wish to persist-- like what happens when line segments meet (i.e., line joins). So when understanding this section on line properties, it's important to visualize two possible cases-- a single line segment and the properties of that segment, and the case of multiple line segments (e.g., an outline of an autoshape). By considering both cases, the meaning of most properties becomes intuitively clear.

Line properties consist of several sections: line fill properties, line dash properties, line join properties, head/tail properties, as well as a few attributes.

5.8.5.1 Line Fill Properties

Line fill properties are a proper subset of general fill properties. One of the following can be used: noFill, solidFill, gradFill, or pattFill. Blip fills and group fills are not permitted for line fill properties.

5.8.5.2 Line Dash Properties

Line Dash properties may we either one of the presets, a:prstDash, or a custom dashing scheme, a:custDash. For the presets, the following options are available:

- solid: Solid (continuous) pen.
- dot: Dot style. [- - - - - - - - - - - -]
- dash: Short dash style. [---- ---- ---- ---- ---- ----]
- lgDash: Long dash style. [-------- -------- -------- --------]
- dashDot" Short dash followed by dot. [---- - ---- - ---- - ---- -]
- lgDashDot: Long dash followed by dot. [-------- - -------- - -------- -]
- lgDashDotDot: Long dash followed by two dots. [-------- - - -------- - -]
- sysDash: System short dash style (PS_DASH). [--- --- --- --- --- --- --- --- --- --- ---]
- sysDot: System dot style (PS_DOT). [-]
- sysDashDot: System short dash and one dot (PS_DASHDOT) [--- - --- - --- - --- - --- - --- - --- -]
- sysDashDotDot: System short dash and two dots (PS_DASHDOTDOT) [--- - - --- - - --- - - --- - - --- - - --- - -]

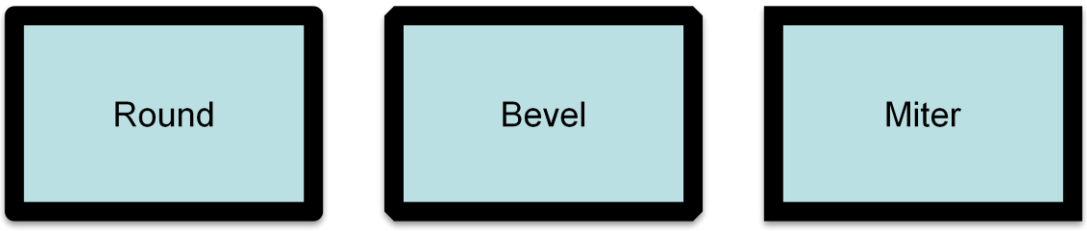
5.8.5.2.1 Custom Dashes

Custom dashes allow full flexibility in expressing any dashing scheme. Custom dashes are also known as dash stop lists, a:ds, due to the way the custom dashes are expressed. An element of the list specifies two attributes: d for the length of the dash relative to line width, and sp for length of the space relative to line width. Any number of elements may be combined into a dash stop list for full generality in expressing dashing schemes.

5.8.5.3 Line Join Properties

```
<a:ln w="38100" cap="sq" cmpd="thickThin" algn="ctr">
  <a:solidFill>
    <a:schemeClr val="lt1"/>
  </a:solidFill>
  <a:prstDash val="sysDot"/>
  <a:bevel/>
</a:ln>
```

Line join properties are for expressing the visual appearance of what happens when line segments meet. They can be round, beveled, or mitered. Notice the corners of the following rectangles, which illustrate the effect line join properties have.

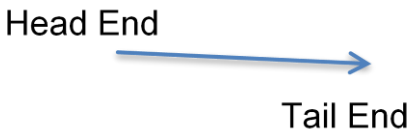


The only attribute of line join properties is lim. This attribute limits the amount by which lines can be extended to form a join. Normally, this is a relatively infrequent occurrence, but in the case of nearly parallel lines, this attribute comes into play.

5.8.5.4 Head/Tail End Properties

Head/Tail End properties specify whether there are any special attachments to the head or the tail of a line. All parameters are specified in attributes: a type, a w (width of line end to width of line), and a len (length of the end relative to the line width). By default, no head/tail end properties are applied. The type can be one of: none, triangle, stealth, diamond, oval, arrow. w and len can be one of sm, med and lg corresponding to small, medium, and large respectively.

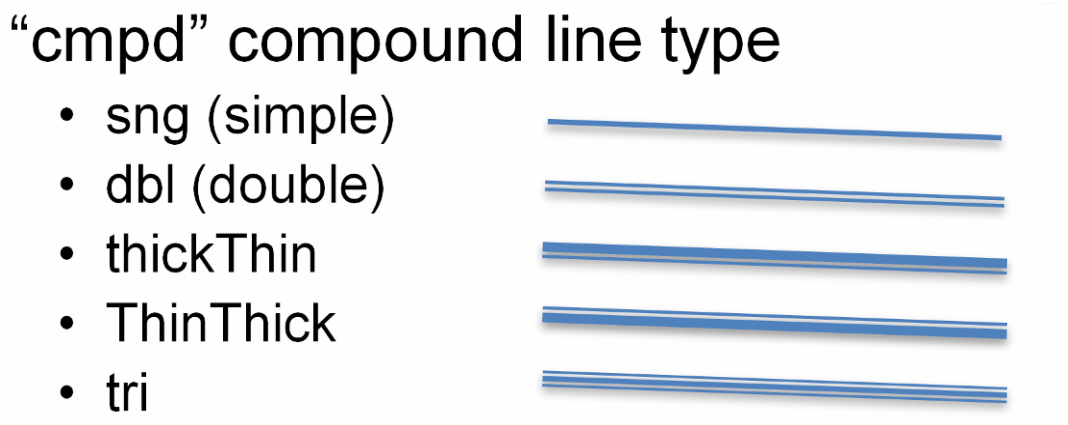
```
<a:ln w="25400" cap="rnd"
      cmpd="sng" algn="ctr">
  <a:solidFill>
    <a:srgbClr val="4F81BD"/>
  </a:solidFill>
  <a:prstDash val="solid"/>
  <a:tailEnd type="arrow" w="lg" len="lg"/>
</a:ln>
```



5.8.5.5 Line Attributes

Line Properties, a:ln, takes several attributes: w specifies the line width. cap specifies whether the line ends are round (value rnd), square (sq), or flat (flat).

cmpd specifies a compound line type. Its permitted values are shown below:



5.8.6 Effects

Effects are most naturally applied to shapes, but like fill properties and line properties, they can apply to shapes, pictures, and text. Effects are represented two ways: via an Effect List, a:effectLst, or an effects container, a:effectDag.

5.8.6.1 Effects Lists

```
<p:sp>
  <p:nvSpPr>...
  <p:spPr>
    <a:xfrm>...
    <a:prstGeom prst="star5">...
    <a:effectLst>
      <a:glow rad="190500">
        <a:schemeClr val="accent5">
          <a:alpha val="80000"/>
        </a:schemeClr>
      </a:glow>
      <a:outerShdw blurRad="50800" dist="50800"
                   dir="2700000" algn="tl"
                   rotWithShape="0">
        <a:srgbClr val="000000">
          <a:alpha val="43137"/>
        </a:srgbClr>
      </a:outerShdw>
      <a:reflection stA="75000" endA="10000"
                    dist="101600" dir="5400000"
                    sy="-100000" algn="bl"
                    rotWithShape="0"/>
      <a:softEdge rad="31750"/>
    </a:effectLst>
    <a:scene3d>...
    <a:sp3d>...
  </p:spPr>
  <p:style>...
  <p:txBody>...
</p:sp>
```

An effect list is made up of one or more primitive effects that can be applied one after another. The primitives are:

- Blur
- fillOverlay
- glow
- innerShdw
- outerShdw
- prstShdw
- reflection
- softEdge

5.8.6.2 Blur

Blur blurs all color channels, including alpha. Two attributes, rad (radius of blur) and grow (boolean), apply here. grow specifies if the bounds should grow as a result of the blurring.

5.8.6.3 Inner Shadow

```
<a:effectLst>
  <a:innerShdw blurRad="317500"
               dist="293171"
               dir="13500000">
    <a:srgbClr val="000000">
      <a:alpha val="43000"/>
    </a:srgbClr>
  </a:innerShdw>
</a:effectLst>
```

Inner Shadows contain a color choice, as well as three attributes:

- blurRad: blur radius
- dist: how far to offset the shadow
- dir: direction to offset the shadow

5.8.6.4 Outer Shadow

```
<a:effectLst>
  <a:outerShdw blurRad="50800" dist="50800"
               dir="2700000"
               sx="106000" sy="106000"
               algn="tl" rotWithShape="0">
    <a:srgbClr val="000000">
      <a:alpha val="43137"/>
    </a:srgbClr>
  </a:outerShdw>
</a:effectLst>
```

Outer shadows contain a color choice as well as several attributes:

- blurRad: blur radius
- dist: how far to offset the shadow
- dir: direction to offset the shadow
- sx, sy: horizontal/vertical scale factors
- kx, ky: horizontal/vertical skew angles

- algn: shadow alignment. Alignment happens first and effectively sets the origin for scale, skew, and offset
- rotWithShape: (boolean) Rotate shadow with shape

5.8.6.5 Preset Shadows

Preset shadows consist of a color choice, and a preset shadow:

shdw1
shdw2
shdw3
shdw4
shdw5
shdw6
shdw7
shdw8
shdw9
shdw10
shdw11
shdw12
shdw13
shdw14
shdw15
shdw16
shdw17
shdw18
shdw19
shdw20

The attributes for Preset Shadows are:

- dist: how far to offset the shadow
- dir: direction to offset the shadow

5.8.6.6 Reflection Effects

```
<a:effectLst>
  <a:reflection blurRad="12700" stA="50000" endPos="75000"
                dist="12700" dir="5400000" sy="-100000"
                algn="bl" rotWithShape="0"/>
</a:effectLst>
```

Reflections are represented entirely through attributes:

- blurRad: Blur Radius
- stA: (Start Alpha) starting reflection opacity
- stPos: start position along gradient ramp of start alpha value
- endA: (End Alpha) ending reflection opacity
- endPos: end position along gradient, ramp of end alpha value
- dist: how far to offset reflection
- dir: Direction to offset reflection
- fadeDir: direction of alpha gradient, ramp relative to shape itself
- sx, sy: horizontal/vertical scale factors
- kx, ky: horizontal/vertical skew angles
- algn: reflection alignment
- rotWithShape: (boolean)
- rotate: reflection with shape

5.8.6.7 Soft Edge Effects

```
<a:effectLst>
  <a:softEdge rad="127000"/>
</a:effectLst>
```

Soft Edge blurs the edges of the applied object subject to the specified blur radius rad.

5.8.6.8 Glow Effects

```
<a:effectLst>
  <a:glow rad="101600">
    <a:schemeClr val="accent2">
      <a:alpha val="60000"/>
    </a:schemeClr>
  </a:glow>
</a:effectLst>
```

A glow effect is very similar to a soft edge effect, but differs in that it permits a color specification in addition to rad. Basically, a glow is a soft edge effect, except with the color specified used instead of the object's color.

5.9 Shape Definitions and Attributes

5.9.1 Introduction

This document provides a high-level overview of the content described in the dml-shapeGeometry.xsd schema.

This aspect of DrawingML deals mainly with the shapes and their attributes, and is broken down into two topics:

- Working with preset shapes

- Defining custom shapes and their properties

5.9.2 The Coordinate Systems

To specify a shape there are a few high level systems that must first be understood, namely the coordinate systems that will be used. These are the document, shape and path coordinate systems, described in the following sub clauses.

5.9.2.1 The Document Coordinate System

To first specify a shape within a document the document coordinate system must be understood. This system has both an x and y component and starts with a value of (0,0) in the upper left corner of the document. As the x-coordinate increases, the point will move to the right. As the y-coordinate increases, the point will move downwards. The units of measurement within the document coordinate system are EMUs (91440 EMUs/U.S. inch, 36000 EMUs/cm). In addition to specifying a position for the shape, you must also specify the width and height of the shape, which is called the extent of the shape. This value is again measured in EMUs. To specify these two values, the following transform would be used.

```
<p:sp>
  <p:spPr>
    <a:xfrm>
      <a:off x="3200400" y="1600200"/>
      <a:ext cx="1200000" cy="1000000"/>
    </a:xfrm>
  </p:spPr>
</p:sp>
```

Here we can see that this new shape will be placed at x = 3200400 and y = 1600200 within the document coordinate system. In addition, we see that this shape will have a width of 1200000 EMUs and a height of 1000000 EMUs.

The width and height set the bounding box within which the entire shape will be contained.

5.9.2.2 The Shape Coordinate System

Now that we have a width and height specified, we can now move into the explanation of the shape coordinate system. The shape coordinate system has both an x and y component and starts with a value of (0,0) in the upper left corner of the shape. The width and height of this coordinate system are specified by the extent of the shape, which was recently specified above, and the units are once again EMUs. This coordinate system is used to define the locations of many of the shape attributes.

5.9.2.3 The Path Coordinate System

The final coordinate system is the path coordinate system which also has both an x and y component and starts with a value of (0,0) in the upper left corner of the shape. Now it must be known that this coordinate system is a unique one in that its units are relative to the specified width and height of the coordinate space. The path coordinate system has exactly the same EMU dimensions as the shape coordinate system but

different units. While the shape coordinate system uses EMUs, the path coordinate system uses (1/width) as the x units and (1/height) as the y units. That is if the path was specified to have a width of 2 and a height of 1, then the path coordinate (1,1) would be equivalent to (600000,1000000) in the shape coordinate system. The path coordinate system will be better understood later, once the path element is described.

Note that all dimensions and coordinates must be specified using whole numbers.

5.9.3 Specifying a Preset Shape

Within the Shape Definitions and Attributes section of DrawingML there are many pre-defined shapes that can be used, 187 to be exact. Of course, if the user does not wish to use a preset shape there is always the option of specifying a custom shape that will be described further in §5.9.4.

5.9.3.1 Defining a Preset Shape

It is quite easy to specify a preset shape as that is the whole notion around presets. They are meant to solve the most common cases of shape definition.

To specify a heart shape for instance the following DrawingML code can be used.

```
<p:sp>
  <p:spPr>
    <a:xfrm>
      <a:off x="1981200" y="533400"/>
      <a:ext cx="1143000" cy="1066800"/>
    </a:xfrm>
    <a:prstGeom prst="heart">
    </a:prstGeom>
  </p:spPr>
</p:sp>
```

This heart is rendered by the generating application using the custom shape code for this shape, which is fully documented within ST_ShapeTypes located in the reference documentation. Thus, we see that the user need on specify the preset name to place a shape within their document.

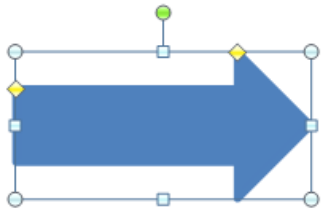
5.9.3.2 Adjusting a Preset Shape

While specifying a preset shape is convenient and looks good most of the time. There may also be the need for the user to adjust this preset to more closely suit the needs of their document. For this we introduce the notion of adjust values. The preset shape is built using lines, curves and calculations, just as a custom shape

would be defined. To allow for the adjusting of these preset shapes we have based certain properties of shapes on adjust values rather than concrete dimensions. This means that they can be modified which will in turn modify the geometry of the shape.

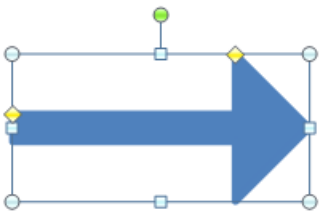
A simple arrow would be specified using the following DrawingML code.

```
<p:sp>
  <p:spPr>
    <a:xfrm>
      <a:off x="3276600" y="990600"/>
      <a:ext cx="978408" cy="484632"/>
    </a:xfrm>
    <a:prstGeom prst="rightArrow">
      <a:avLst>
        <a:gd name="adj1" fmla="val 50000"/>
        <a:gd name="adj2" fmla="val 50000"/>
      </a:avLst>
    </a:prstGeom>
  </p:spPr>
</p:sp>
```



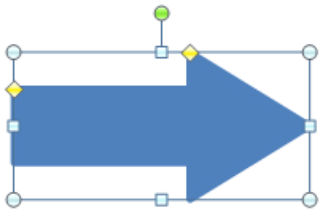
This will specify the basic arrow shown above which might be sufficient for the document needs of the user but it also may not. If this standard arrow is not sufficient then the two adjust values for this shape may be adjusted. For instance, if the body of the arrow is too large then the value for adj1 can be decreased. The following DrawingML code would specify such a case.

```
<a:gd name="adj1" fmla="val 18553"/>
```



Similarly, if the arrow head itself was too short then the value of adj2 can be increased. The following DrawingML code would specify such a case.

```
<a:gd name="adj2" fmla="val 81447"/>
```



Thus, it can be seen that while each preset is indeed a preset with a pre-defined geometry, it can be modified. Through the use of adjust values, the user is able to custom fit a preset shape to their document needs without having to specify an entirely custom shape.

Note that the values used here for adjust values have no real units as they are simply input parameters into the equations that make up the shape geometry. More on these equations and their parameters will be discussed in §5.9.4.2.

5.9.4 Specifying a Custom Shape

In addition to the specifying of a preset shape there is also the possibility of a specifying a custom shape. This is accomplished by defining a geometry from a set of construction methods and applying various shape properties to this geometry. This compliments preset shapes, giving the user the opportunity to specify a complete shape with any custom properties that are deemed necessary.

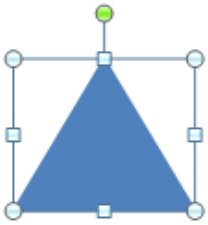
5.9.4.1 Defining the Geometry

Just like a preset shape, a custom shape has a position and a shape bounding box that is specified by the offset and extent transform values. The shape coordinate system is defined by these values as was described in section 1.2 above. The path coordinate system is also partially defined by these in that it has it's width and height set by these values. The units of the path system however are determined by the specified width and height of the path.

A custom shape with a single path can be specified using the following DrawingML code.

```
<p:sp>
  <p:spPr>
    <a:xfrm>
      <a:off x="3200400" y="1600200"/>
      <a:ext cx="1200000" cy="1000000"/>
    </a:xfrm>
    <a:custGeom>
      <a:pathLst>
        <a:path w="2" h="2">
          <a:moveTo>
            <a:pt x="0" y="2"/>
          </a:moveTo>
```

```
            <a:lnTo>
              <a:pt x="2" y="2"/>
            </a:lnTo>
            <a:lnTo>
              <a:pt x="1" y="0"/>
            </a:lnTo>
            <a:close/>
          </a:path>
        </a:pathLst>
      </a:custGeom>
    </p:spPr>
  </p:sp>
```



As can be seen in the above code, the path has a width and height of 2. This means that the path coordinate space will have units of (1/2 * shape width) for x-coordinate and (1/2 * shape height) for the y-coordinate. Thus we see that a coordinate of (2,2) in the path coordinate system will be the same as (1200000,1000000) within the shape coordinate system.

To define the shape path above we can see that there are a few different parts to defining this custom shape. The first is to define the first path in what is called the path list. It should be noted that the path list can have multiple paths in it, some filled, some not, some outlined, some not. To define the path we must specify the width, height and thus units for this path via the following DrawingML.

```
  <a:path w="2" h="2">
```

This sets up the path coordinate system for this path as was previously described. Next we need to move the drawing cursor to the point in this path coordinate system that we wish to start drawing our shape from. The following DrawingML does just that.

```
  <a:moveTo>
    <a:pt x="0" y="2"/>
  </a:moveTo>
```

This will move the drawing cursor to the bottom left position (0,2) which is equivalent to (0,1000000) in the shape coordinate system. Following this we can now start by drawing the first line in the shape via the following line.

```
<a:lnTo>
  <a:pt x="2" y="2"/>
</a:lnTo>
```

This will draw a line from the current drawing cursor position of (0,2) to (2,2) which is the bottom right corner of the path coordinate system. This is equivalent to drawing a line from (0,1000000) to (1200000,1000000) in the shape coordinate system. Now that we have the bottom edge of the triangle drawn we can continue to the final edge in the shape via the following.

```
<a:lnTo>
  <a:pt x="1" y="0"/>
</a:lnTo>
```

This will draw the final line that will be drawn from the current drawing cursor position of (2,2) to (1,0) which is in the top middle of the path coordinate system. This is equivalent to drawing a line from (1200000,1000000) to (600000,1000000) in the shape coordinate system. Now that most of the triangle has been drawn. It should be noted that since the <close/> element is specified at the end of the path that a line will be drawn from the last point in the path back to the first point in the path. This explains a bit of the existence of the following final element.

```
<a:close/>
```

This finalizes the edges of the shape path being specified. Since the fill of this path is set to normal, this path will have a fill no matter if this close tag is specified or not. However, the fact that it is specified determines that there will be a final edge drawn between the final drawing cursor point and the path starting point. Now that the path has been fully specified, this shape can be filled and thus be considered finished.

5.9.4.2 Adjusting the Geometry

Now that we have shown how a custom shape can be specified we can look at how it might be adjusted. This adjusting is different from the typical resizing that can happen by using the shape transform elements. Using these shape adjusting methods, a shape can be made to have many different resize/adjustment characteristics.

5.9.4.3 Geometry Guides

A guide within a shape is essentially an equation with a set number of inputs and a single output. A guide is used to calculate construction values for a shape and thus can be manipulated to govern the shape's overall geometry.

An example of this can be seen in the following DrawingML.

```
<gdLst>
  <gd name=”y1” fmla=”*/ h adj1 100”/>
</gdLst>
```

This guide will calculate it's output based on 3 input parameters and assign this output to a guide named y1. The formula that will be used in the calculation here is the multiply divide formula. The result for this guide will be calculated in the following manner: y1 = ((h * adj1) / 100). After the result here is calculated, the guide y1 can be used later within the <gdlst> or <path> to calculate further values. That is it can be used as an input for calculating another guide value. These guides then allow for a path to be based off of series of equations rather than static path coordinate values. To use a guide in the defining of a path we would simply specify the following within the path list.

```
<a:lnTo>
  <a:pt x="2" y="y1"/>
</a:lnTo>
```

This would draw a line to the point (2,y1) where y1 is the calculated result of the guide equation shown above. The drawing of this line will then change based on the input parameters of h and adj1 which are previously calculated guides as well.

Note that while h is a previously calculated guide. It is not calculated for each shape, rather it is a built-in guide that the generating application makes available to the shape.

5.9.4.3.1 Adjust Handles

To allow for the adjusting UI of a shape we introduce the notion of an adjust handle. This adjust handle will be linked to adjust values that will then be used as input to the guide equations defined previously. The numerical chain described here will thus directly change the geometry of the related shape. There are two types of adjust handles that can be specified. An XY adjust handle acts in the horizontal/vertical direction and has two related guides, both a horizontal and a vertical respectively. A polar adjust handle acts in a polar manner and has two related guides as well. One guide for the radial width and the other for the radial angle. An adjust handle is specified to have an x and y coordinate as well as these adjust handles. This adjust handle can then be moved around in a generating application's UI to adjust a pair of guides which will in turn adjust the shape being rendered.

An adjust handle can be specified by the following DrawingML.

```
<ahXY gdRefX="adj1" minX="-2147483647" maxX="2147483647" gdRefY="adj2"
  minY="-2147483647" maxY="2147483647">
  <pos x=”x1” y=”y1”/>
</ahXY>
```

Above is an XY adjust handle that has two guide references, a min and max allowed position for both the x and y coordinates as well as a position within the shape coordinate system where this adjust handle should be placed.

5.9.4.4 Additional Properties

In addition to specifying the geometry for a shape and all the associated adjustments for it there are also a few other properties that are of special significance. These properties do not act on the geometry of the shape but instead enhance a shape so that it may be used for a more specialized task.

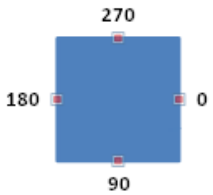
5.9.4.4.1 Connection Sites

As one may have experienced when trying to draw a diagram with shapes and connections between those shapes, it is quite difficult to move a part of your diagram without entirely redrawing the connections between shapes. For this, there is the notion of connection sites that allow for the specification of specific points within a shape to attach connection shapes to. This allows a user to build a diagram from a set of shapes and connect them together using connection shapes. A connection site is specified within the connection list and consists of an x-coordinate, y-coordinate and an attachment angle.

The following DrawingML code defines two connection sites, one at each edge of this triangle.

```
<a:cxnLst>
  <a:cxn ang="10800000">
    <a:pos x="0" y="679622"/>
  </a:cxn>
  <a:cxn ang="0">
    <a:pos x="1705233" y="679622"/>
  </a:cxn>
</a:cxnLst>
```

The attachment angle works by specifying an angle in 60,000ths of a degree that a connector should attach to. The diagram below shows an actual connection point and the attachment angles that correspond to the sides of this point. This information along with the geometry of the shape is used by the generating applications connector routing algorithm to correctly route connectors around connected shapes.



5.9.4.4.2 Text Rectangle

Within each shape is a text box that allows for the attaching of text to any given shape. The text rectangle defines where text will reside within the shape. Depending on Auto-fit options that are selected for the body

of text attached to this shape the text may intentionally flow outside this text rectangle. It must also be pointed out that this text rectangle will also be the bounding box that is used to compute the geometry of a <prstTxWarp>. The EMU dimensions of this text rectangle will be used to compute this geometry just like the transform extent element is used to compute the actual shape.

The following DrawingML specifies a text rectangle within a shape.

```
<a:rect l="0" t="0" r="1200000" b="1000000"/>
```

The text rectangle shown above will have a left edge of 0 x-coordinate, top edge of 0 y-coordinate, right edge of 1200000 x-coordinate and a bottom edge of 1000000 y-coordinate. This will effectively specify a space that is 1200000 EMUs in width and 1000000 EMUs in height.

Note that the edges of this text rectangle can be set so as to allow text to be placed outside the actual geometry of the shape.

5.10 Pictures

5.10.1 Introduction

This subclause provides a high-level overview of the content described in the dml-picture.xsd schema.

The DrawingML Picture file format is broken down into the following subjects:

- Specifying a basic picture
- Attaching properties to this picture
- Transforming this picture

The best way to understand the above subjects will be to cover them in the ordering above.

5.10.2 Specifying a Basic Picture

A picture can be inserted into a presentation slide by use of the picture element, pic, which is similar to the shape element but contains some key differences that enable more complete storage of picture information. This basic picture element should contain a blipfill and some basic non-visual picture properties.

```
<p:pic>
  <p:nvPicPr>
    <p:cNvPr id="4" name="St_Patrick's_Day.jpg"/>
    <p:cNvPicPr>
      <a:picLocks noChangeAspect="1"/>
    </p:cNvPicPr>
    <p:nvPr/>
  </p:nvPicPr>
  <p:blipFill>
    <a:blip r:embed="rId2"/>
    <a:stretch>
      <a:fillRect/>
    </a:stretch>
  </p:blipFill>
  <p:spPr>
    <a:xfrm>
      <a:off x="1346200" y="914400"/>
      <a:ext cx="3657600" cy="2743200"/>
    </a:xfrm>
    <a:prstGeom prst="rect">
      <a:avLst/>
    </a:prstGeom>
    <a:noFill/>
    <a:ln>
      <a:noFill/>
    </a:ln>
  </p:spPr>
</p:pic>
```

5.10.3 Attaching Properties to this Picture

Now that the base picture has been specified, we can move on to more complicated properties, such as recolor options and picture descriptions. In the picture below, notice that the picture that was once green has been re-colored in a purple hue. This can be done by utilizing the duotone element, which allows for the setting of two base colors to use for re-coloring the entire picture. The first is used to act upon the darker regions of the picture and the second is used to act upon the lighter regions. This we can see below that black (#000000) is indeed used below for the darker regions while accent4 (purple in this case) is used for the lighter areas.

```
<p:pic>
  <p:nvPicPr>
    <p:cNvPr id="4" name="St_Patrick's_Day.jpg"
      descr="This is a Saint Patrick's day picture"/>
    <p:cNvPicPr>
      <a:picLocks noChangeAspect="1"/>
    </p:cNvPicPr>
    <p:nvPr/>
  </p:nvPicPr>
  <p:blipFill>
    <a:blip r:embed="rId2">
      <a:duotone>
        <a:srgbClr val="000000"/>
        <a:schemeClr val="accent4"/>
      </a:duotone>
    </a:blip>
    <a:stretch>
      <a:fillRect/>
    </a:stretch>
  </p:blipFill>
```

```
    <p:spPr>
      <a:xfrm>
        <a:off x="1346200" y="914400"/>
        <a:ext cx="3657600" cy="2743200"/>
      </a:xfrm>
      <a:prstGeom prst="rect">
        <a:avLst/>
      </a:prstGeom>
      <a:noFill/>
      <a:ln>
        <a:noFill/>
      </a:ln>
    </p:spPr>
  </p:pic>
```

5.10.4 Transforming this Picture

Now that both basic properties and additional picture properties have been specified, we can begin incorporating shape properties. Below is the same picture as described above, with 3D camera perspective applied along with a simple shadow and a white outline. These shape properties are the same that can be applied to a shape element. One picture-specific difference can be seen here with the border around the picture. Instead of the border growing both inward and outward, it only grows outward.

```
  <p:pic>
    <p:nvPicPr>
      <p:cNvPr id="4" name="St_Patrick's_Day.jpg"
        descr="This is a Saint Patrick's day picture"/>
      <p:cNvPicPr>
        <a:picLocks noChangeAspect="1"/>
      </p:cNvPicPr>
      <p:nvPr/>
    </p:nvPicPr>
```

```
      <p:blipFill>
        <a:blip r:embed="rId2">
          <a:duotone>
            <a:srgbClr val="000000"/>
            <a:schemeClr val="accent4"/>
          </a:duotone>
        </a:blip>
        <a:stretch>
          <a:fillRect/>
        </a:stretch>
      </p:blipFill>
      <p:spPr>
        <a:xfrm>
          <a:off x="1346200" y="914400"/>
          <a:ext cx="3657600" cy="2743200"/>
        </a:xfrm>
        <a:prstGeom prst="rect">
          <a:avLst/>
        </a:prstGeom>
        <a:noFill/>
        <a:ln w="57150">
          <a:solidFill>
          <a:schemeClr val="bg1"/>
          </a:solidFill>
        </a:ln>
        <a:effectLst>
          <a:outerShdw blurRad="50800" dist="50800" dir="2700000" algn="tl"
            rotWithShape="0">
            <a:srgbClr val="7D7D7D">
              <a:alpha val="65000"/>
            </a:srgbClr>
          </a:outerShdw>
        </a:effectLst>
        <a:scene3d>
          <a:camera prst="perspectiveRelaxedModerately"/>
          <a:lightRig rig="threePt" dir="t">
            <a:rot lat="0" lon="0" rev="18900000"/>
          </a:lightRig>
        </a:scene3d>
      </p:spPr>
    </p:pic>
```

End of informative text.

5.11 WordprocessingML Drawing

Within a WordprocessingML document, it is possible to include graphical DrawingML objects:

- Charts
- Diagrams
- Locked Canvases
- Pictures

When these objects are present in a word processing document, it is necessary to include information that specifies how the objects are to be positioned relative to the paginated document.

The WordprocessingML Drawing namespace acts in this capacity, specifying all information necessary to anchor and display DrawingML objects within a word processing document.

Consider a DrawingML picture that is to displayed in the center of the printed page on which it appears, modifying the flow of text as necessary. This object would be specified as follows:

```
<w:r>
  <w:drawing>
    <wp:anchor relativeHeight="10" allowOverlap="true">
      <wp:positionH relativeFrom="margin">
        <wp:align>center</wp:align>
      </wp:positionH>
      <wp:positionV relativeFrom="margin">
        <wp:align>center</wp:align>
      </wp:positionV>
      <wp:extent cx="2441542" cy="1828800"/>
      <wp:wrapSquare wrapText="bothSides"/>
      <a:graphic>
        …
      </a:graphic>
    </wp:anchor>
  </w:drawing>
</w:r>
```

The anchor element specifies that this object is not positioned in-line with text, and its child elements specify that the object is centered on the page horizontally and vertically, and that text can wrap around it in a square.

5.11.1 Object Anchoring

When the WordprocessingML Drawing namespace is used to anchor a DrawingML object within a document, that object can be anchored in one of two ways:

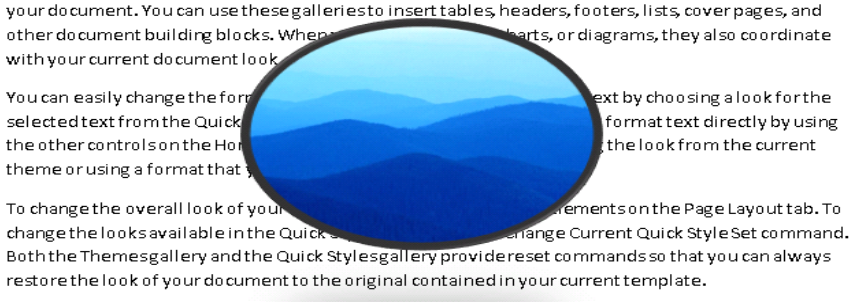
- In line with text - The object is displayed within the regular text stream (modifying line height and so on to accommodate it).

- Floating – The object is positioned absolutely or relatively within the document and text flow is modified as needed around it.

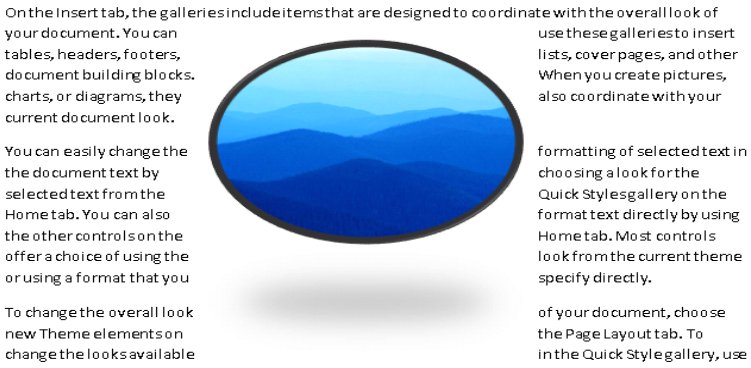
5.11.2 Text Wrapping

Aside from positioning data, WordprocessingML Drawing also needs to specify how text flows around the object. There are five different types of text wrapping which can be applied to floating objects present in WordprocessingML documents:

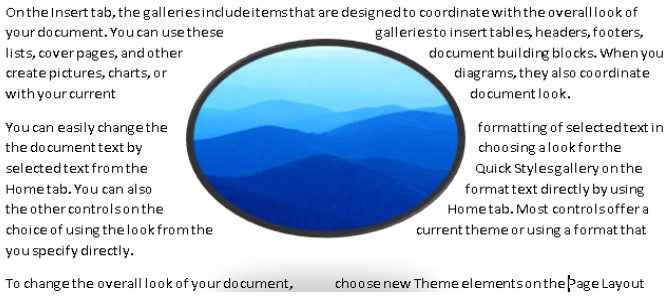
- In Front/Behind Text - In this type of text wrapping, the drawing object is positioned on the document and text is not displaced around it.



- Square Wrapping - In this type of text wrapping, the drawing object is positioned on the document and a rectangle is stored within the file format to determine the wrapping extents.

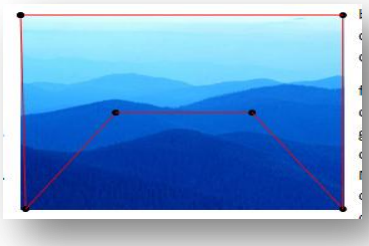


- Tight Wrapping - In this type of text wrapping, a wrapping polygon is created and stored in the WordprocessingML document, and this polygon determines how text wraps around the left and right sides of the drawing object.



- Through Wrapping - In this type of text wrapping, a wrapping polygon is created just like with tight wrapping, but any indents in the wrap polygon can be filled with text in this case.

  If the wrapping polygon looks like the following:

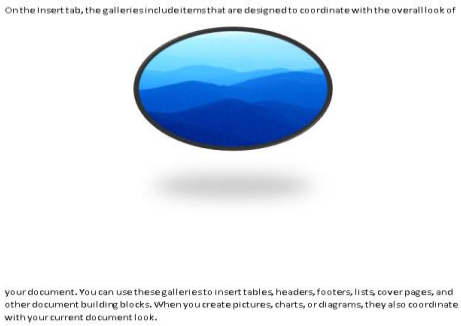


  Tight wrapping would look like this:

  While through wrapping would look like this:

  In the latter case, notice that text fills in the 'indentation' within the wrapping polygon.

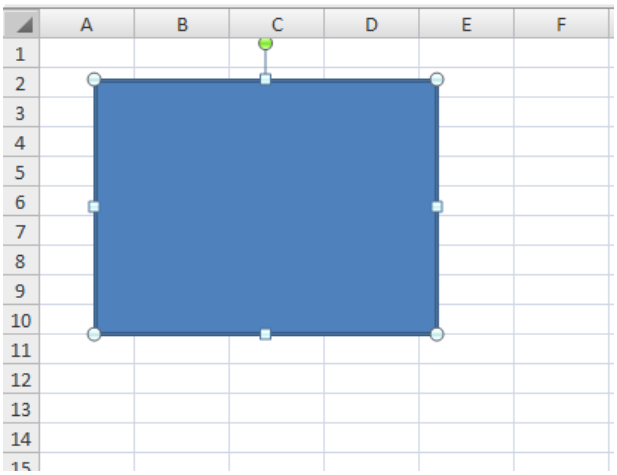
- Top and Bottom Wrapping - In this type of text wrapping, text cannot wrap around either side of the object, and shall only restart below the bottom edge of the document.



5.12 SpreadsheetML Drawing

5.12.1 Introduction

This subclause provides a high-level overview of the content described in the dml-spreadsheetDrawing.xsd schema. The elements in this schema specify how drawing elements are to be described within a spreadsheet. For example, suppose we want to specify a rectangle drawing shape within a worksheet to look like the following:



5.12.2 Overview

The elements that specify the drawing objects within a worksheet are all located within its respective drawing XML file. This file is located under the "drawings" folder inside the spreadsheet file. For example, if the drawing element is located on Worksheet 1, then the specifications for the said element would be located in the file \xl\drawings\drawing1.xml.

5.12.3 Worksheet Drawings

Within the drawing*.xml file is contained a single worksheet drawing wsDr element, which is the parent element for all the drawing elements. Its child specifies the anchoring properties of the drawing elements. It is

within this element that the main specifications for the drawing elements are located. For example in the above screenshot with a simple shape located on the worksheet, the XML for this would look like:

```
<xdr:wsDr>
  <xdr:absoluteAnchor>
    <xdr:pos x="2162175" y="1743075"/>
    <xdr:ext cx="1238250" cy="1314450"/>
    …
    </xdr:sp>
  </xdr:absoluteAnchor>
</xdr:wsDr>
```

In this SpreadsheetDrawingML code there is a single drawing specified almost exactly as it would within the regular DrawingML framework. However the SpreadsheetDrawingML wrapper that is used allows for the specifying of spreadsheet specific properties in addition to the normal drawing properties.

5.12.3.1 Anchoring Types

To define a drawing within a spreadsheet an anchoring type must be chosen. There are three different anchoring types allowed for use within a spreadsheet: Absolute Anchoring, One Cell Anchoring, and Two Cell Anchoring. Each of these types is described in the following subclauses.

5.12.3.1.1 Absolute Anchoring

Absolute Anchoring describes the placement of the drawing within the spreadsheet based upon absolute coordinates. This positioning information includes both position coordinates and extent coordinates. The absoluteAnchor element is what specifies this anchoring behavior and a sample usage is shown below.

```
<xdr:absoluteAnchor>
  <xdr:pos x="2162175" y="1552575"/>
  <xdr:ext cx="1238250" cy="1123950"/>
  …
  </xdr:sp>
</xdr:absoluteAnchor>
```

In this example, there is a single shape specified using absolute anchoring as its anchoring method.

5.12.3.1.2 One Cell Anchoring

One Cell Anchoring describes the placement of the drawing within the spreadsheet based upon offsets as well as a specified column and row. The offset is always in reference to the specified anchor cell and acts to offset the shape object from being exactly on top of the anchor cell. The offset information determines the actual placement of the drawing within the spreadsheet while the row and column are used to specify to which cell the drawing should be anchored. Thus, if the anchor cell changes positions then the drawing can be moved as well. The oneCellAnchor element is what specifies this anchoring behavior and a sample usage is shown below.

```
<xdr:oneCellAnchor>
  <xdr:from>
    <xdr:col>3</xdr:col>
    <xdr:colOff>333375</xdr:colOff>
    <xdr:row>8</xdr:row>
    <xdr:rowOff>28575</xdr:rowOff>
  </xdr:from>
  <xdr:ext cx="1238250" cy="1123950"/>
  …
  </xdr:sp>
</xdr:oneCellAnchor>
```

In this example, there is a single shape specified using one cell anchoring as its anchoring method.

5.12.3.1.3 Two Cell Anchoring

Two Cell Anchoring describes the placement of the drawing within the spreadsheet based upon offsets as well as a specified columns and rows. The offset is always in reference to the specified anchor cell and acts to offset the shape object from being exactly on top of the anchor cell. The offset information determines the actual placement of the drawing within the spreadsheet while the rows and columns are used to specify the cells to which the drawing should be anchored and upon which the resized is based. For instance, if the anchor cell changes positions then the drawing can be moved. Likewise, if the anchor cells behind the shape grow, then the shape can grow as well. The twoCellAnchor element is what specifies this anchoring behavior and a sample usage is shown below.

```
<xdr:twoCellAnchor>
  <xdr:from>
    <xdr:col>3</xdr:col>
    <xdr:colOff>447675</xdr:colOff>
    <xdr:row>8</xdr:row>
    <xdr:rowOff>28575</xdr:rowOff>
  </xdr:from>
  <xdr:to>
    <xdr:col>5</xdr:col>
    <xdr:colOff>466725</xdr:colOff>
    <xdr:row>14</xdr:row>
    <xdr:rowOff>9525</xdr:rowOff>
  </xdr:to>
  …
  </xdr:sp>
</xdr:twoCellAnchor>
```

In this example, there is a single shape specified using two cell anchoring as its anchoring method.

5.13 Charts

5.13.1 Overview

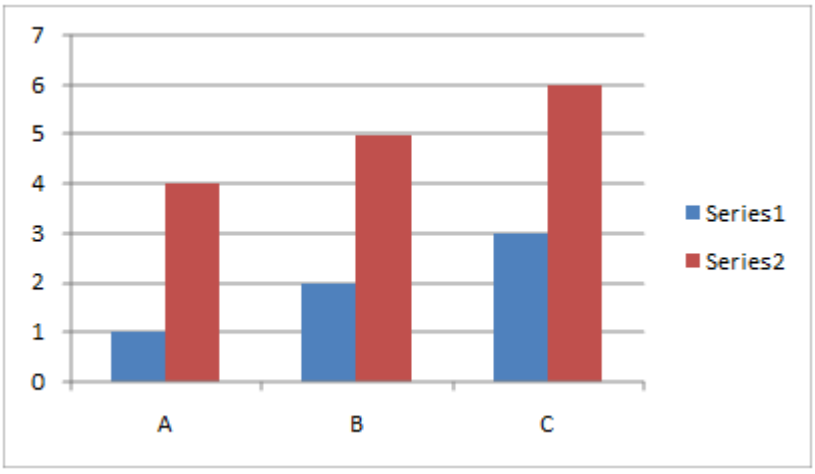
Charts provide a great way to visualize information by displaying a graphical representation of the data. The chart XML files can be reused and shared among different applications, such as a spreadsheet, presentation, and word processing.

Charts come in many different types and flavors, and this document provides a basic overview of both the different types of charts as well as the XML that is used to generate them.

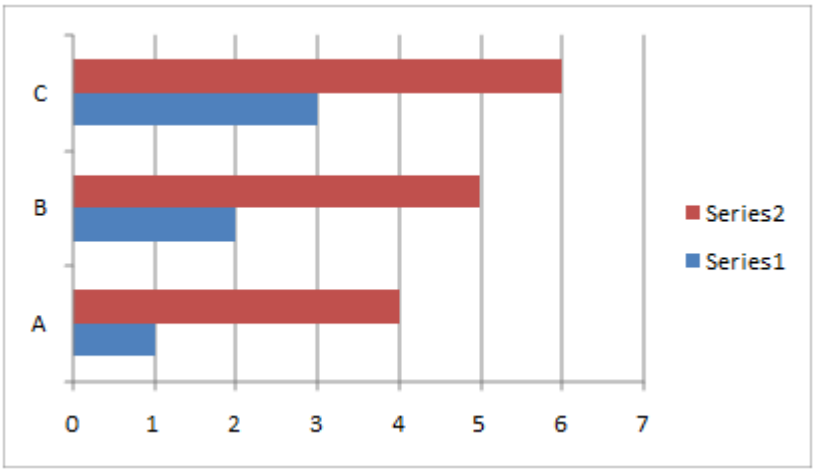
Applications may allow many different runtime behaviors for charts, such as rules for displaying them. This clause and its corresponding reference material define only the XML that is needed to store and generate the charts, and do not dictate any runtime behaviors.

5.13.1.1 Basic Chart Types

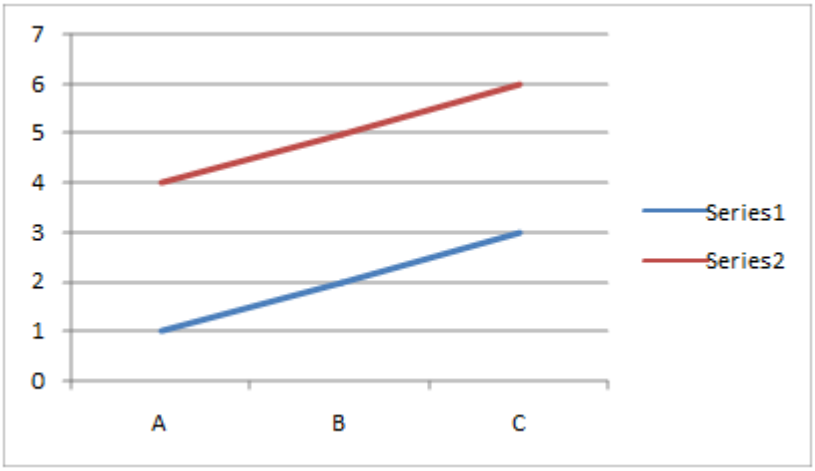
There are 10 basic chart types. Below are examples of each basic type:



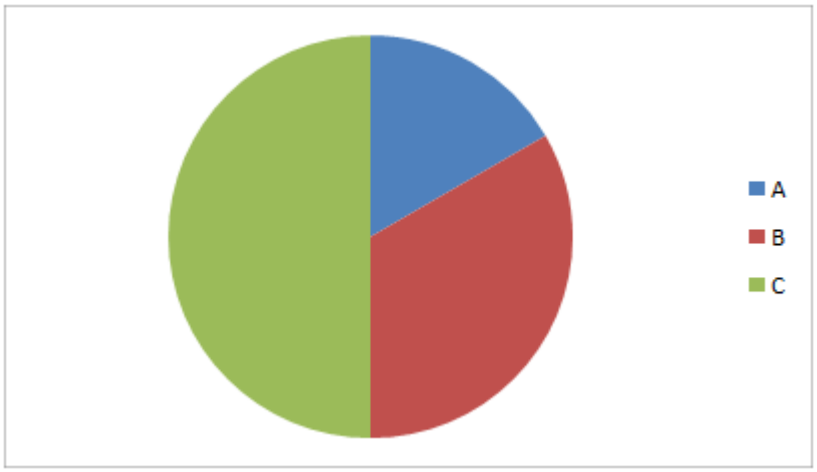
Column Chart (shown above)



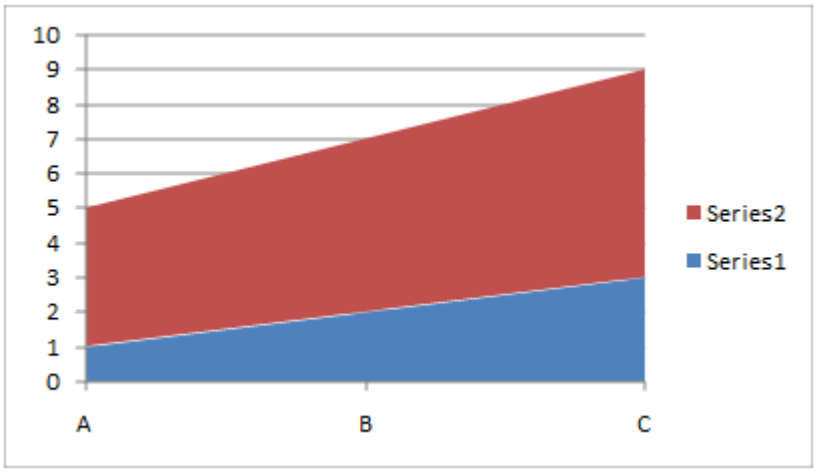
Bar Chart (shown above)



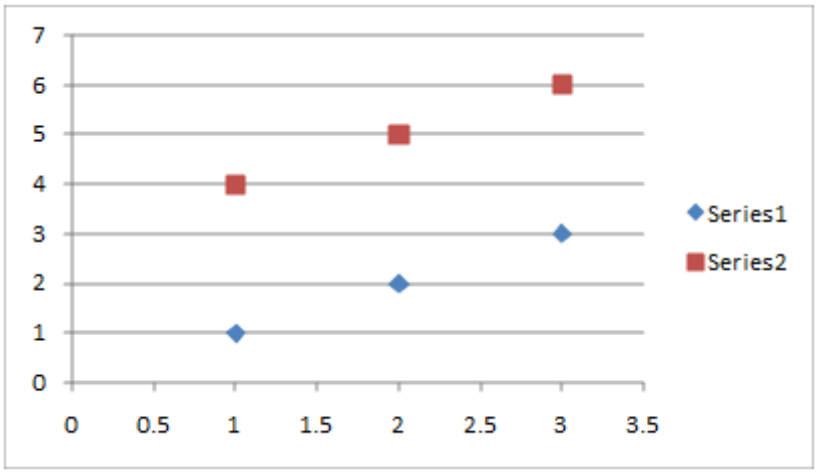
Line Chart (shown above)



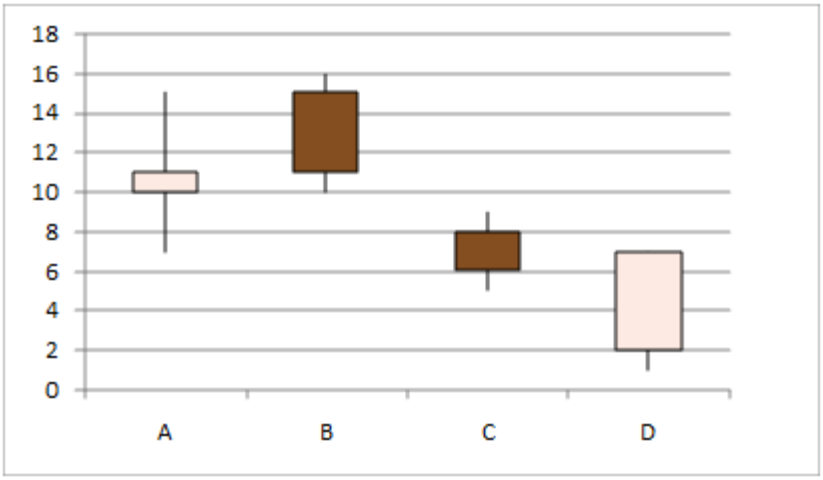
Pie Chart (shown above)



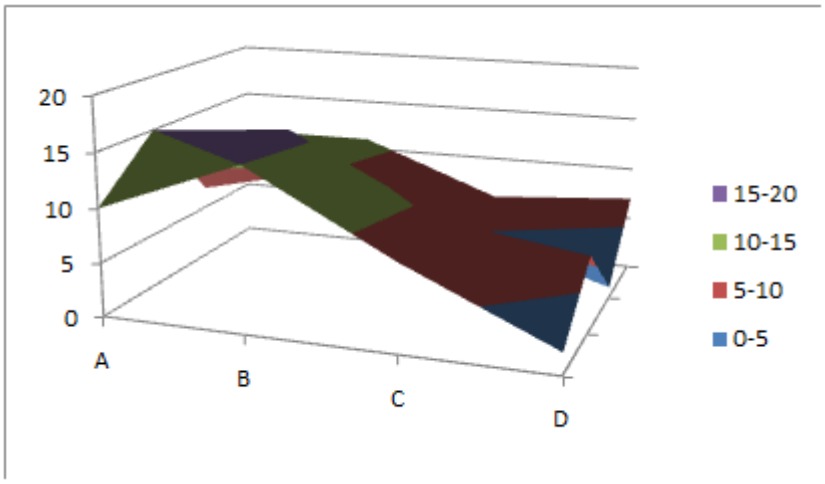
Area Chart (shown above)



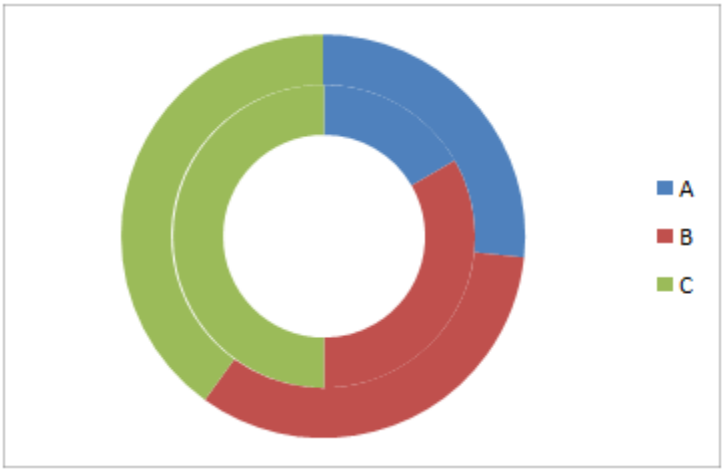
Scatter Chart (shown above)



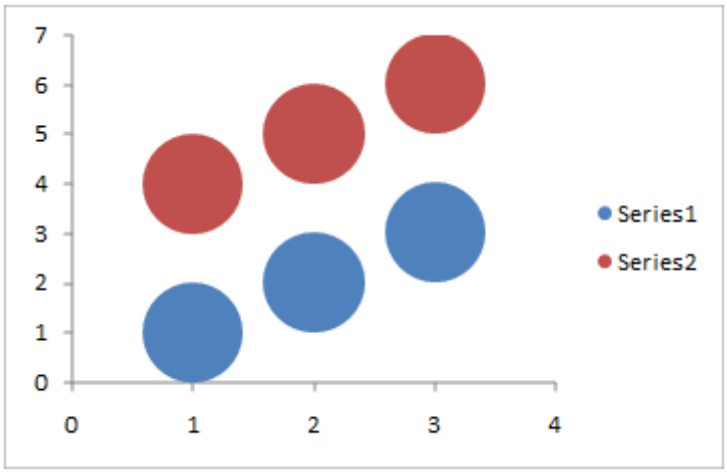
Stock Chart (shown above)



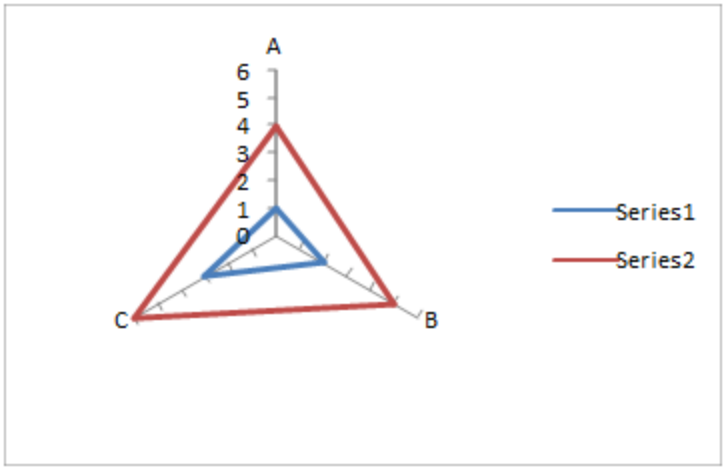
Surface Chart (shown above)



Doughnut Chart (shown above)



Bubble Chart (shown above)



Radar Chart (shown above)

5.13.1.2 Basic Chart Components

Here are some diagrams that label the different individual components of a chart. Some chart components, such as drop lines, are only shown on certain types of charts.

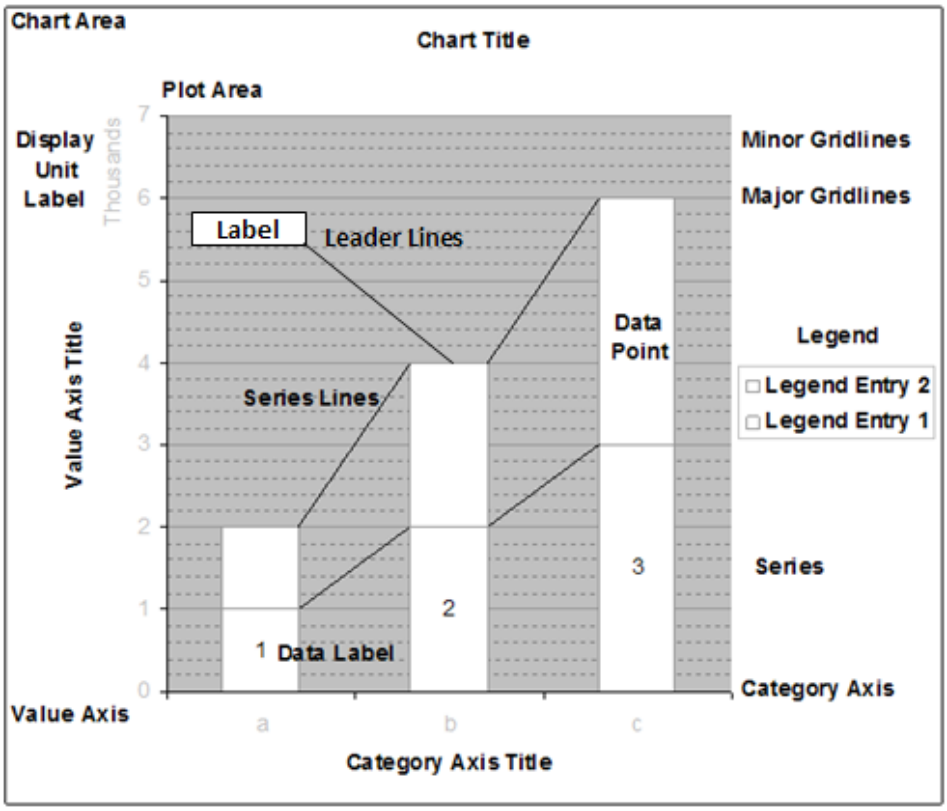
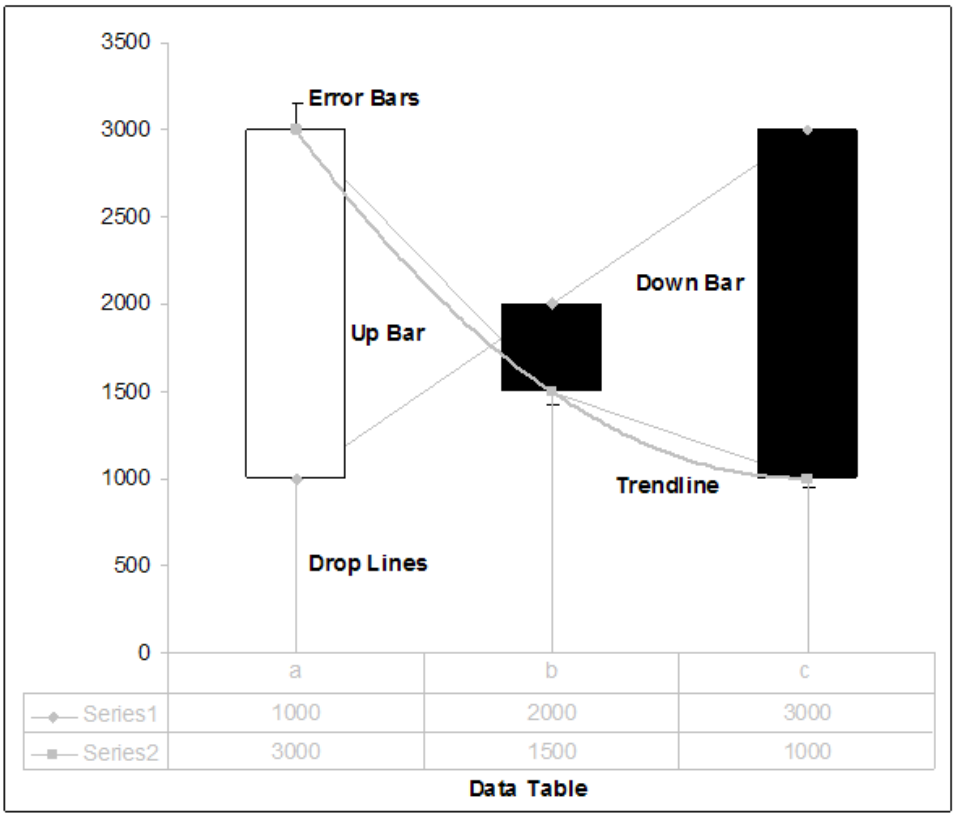
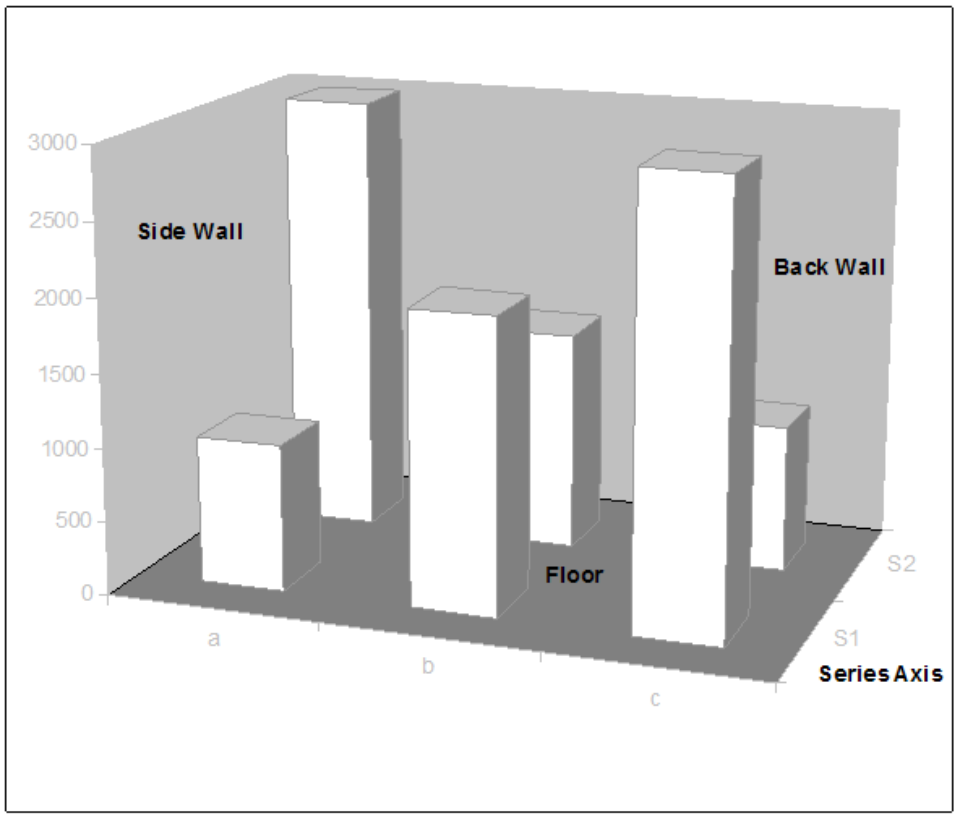


Chart Area
Chart Title
Plot Area
Display Unit Label
Thousands
Minor Gridlines
Major Gridlines
Label
Leader Lines
Data Point
Legend
Series Lines
Legend Entry 2
Legend Entry 1
Value Axis Title
3
Series
2
1
Data Label
Category Axis
Value Axis
a
b
c
Category Axis Title

1

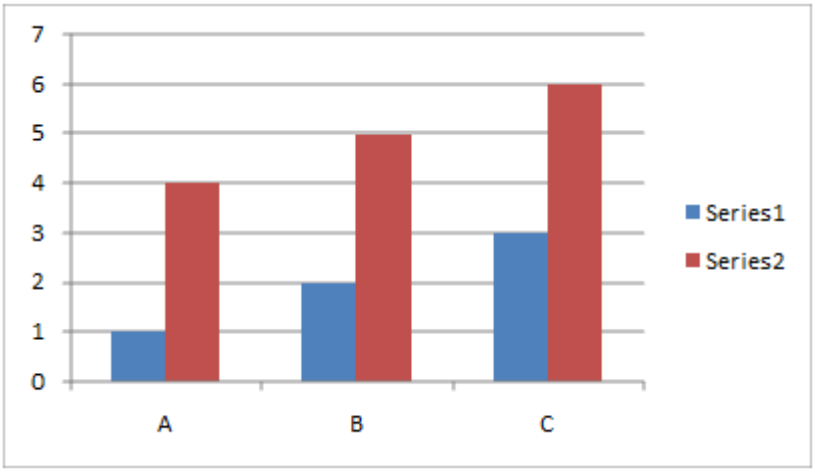

Error Bars
Down Bar
Up Bar
Trendline
Drop Lines
a
b
c
Series1
1000
2000
3000
Series2
3000
1500
1000
Data Table

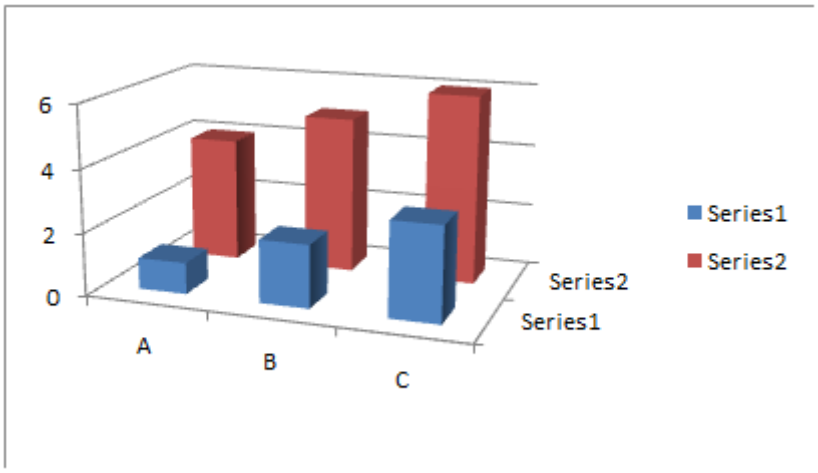
2



5.13.1.3 3D Charts

Most chart types also have three-dimensional representations. 3D charts have extra properties to describe depth, floor, or walls, as well as some other rendering effects. Below is a 2D column chart shown with its 3D counterpart.

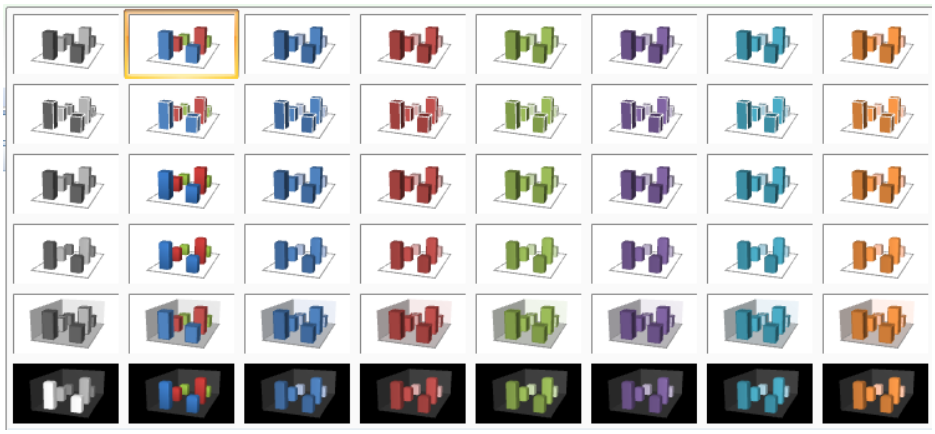




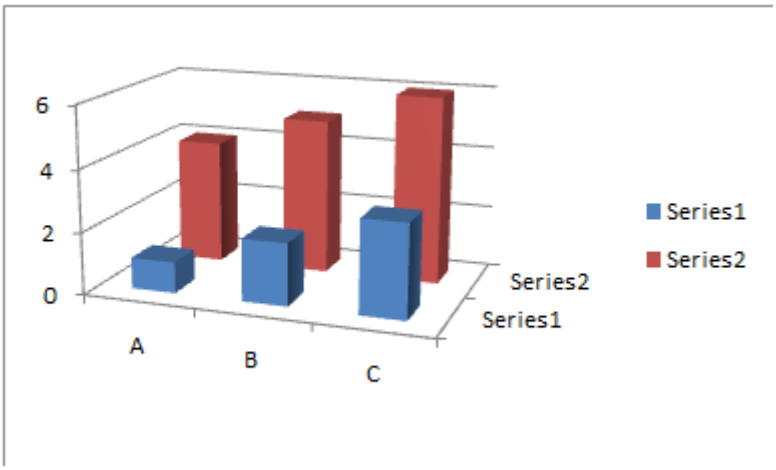
5.13.1.4 Chart Styles

Charts may have different styles applied to them. This is essentially just a coordinated set of coloring and formatting that is applied across an entire chart and all its elements. Styles allow a quick and easy way to coordinate the look and feel of a chart with the rest of the document. Below is a rendering of a column chart and pie chart showing the same data, with the same style applied. There is also a shot of many different types of styles. It is shown for both charts to illustrate how the idea of a style can apply, and be consistent, across different chart types.

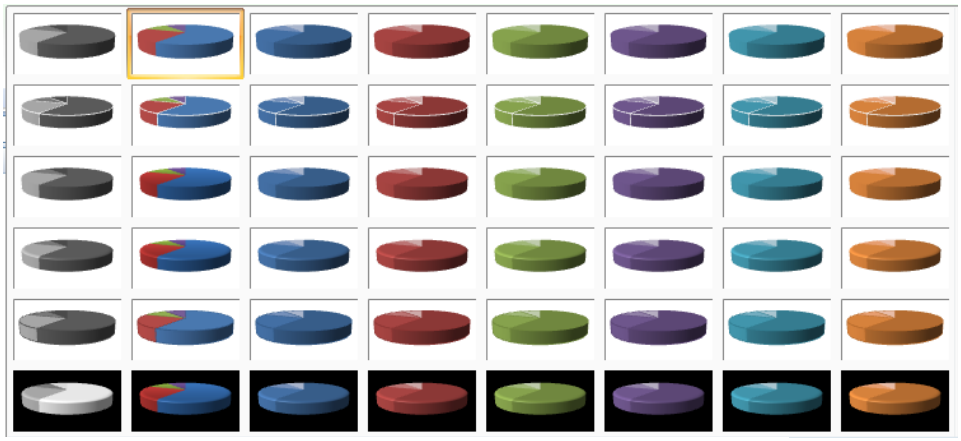
1

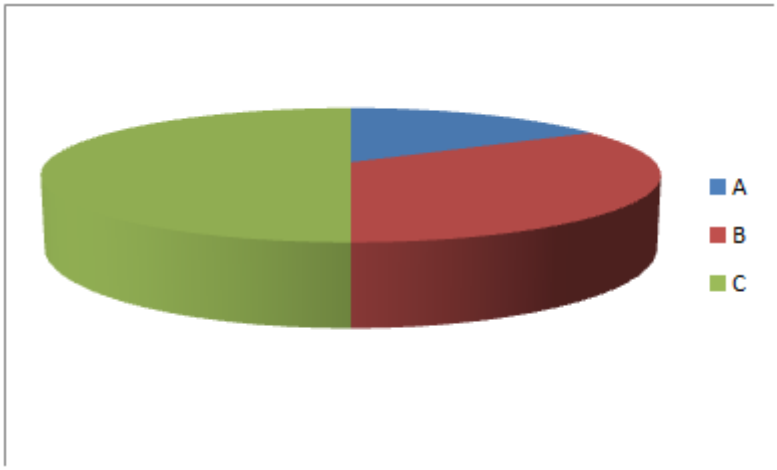


2



3





5.13.2 XML Overview

5.13.2.1 Relationships

A Drawing XML part contains a chart element, which expresses a relationship ID. This ID is referenced by the drawing.rels part, which points to the corresponding chart XML part. Chart XML contains the core definition of the chart.

5.13.2.2 Chart

Different chart types can have many different components defined in the XML, and not all are shown here. For many charts though, at a very high level, the chart XML is composed of the following pieces:

```
<chartSpace>
  <chart>
    <view3D>
    <plotArea>
      <layOut>
      <barChart>
        <cat>
        <val>
      <catAx>
      <valAx>
    </plotArea>
    <legend>
  </chart>
  <printSettings>
</chartSpace>
```

chartSpace is the root node, which contains an element defining the chart, and an element defining the print settings for the chart.

chart is the root element for the chart. If the chart is a 3D chart, then a view3D element is contained, which specifies the 3D view. It then has a plot area, which defines a layout and contains an element that corresponds to, and defines, the type of chart.

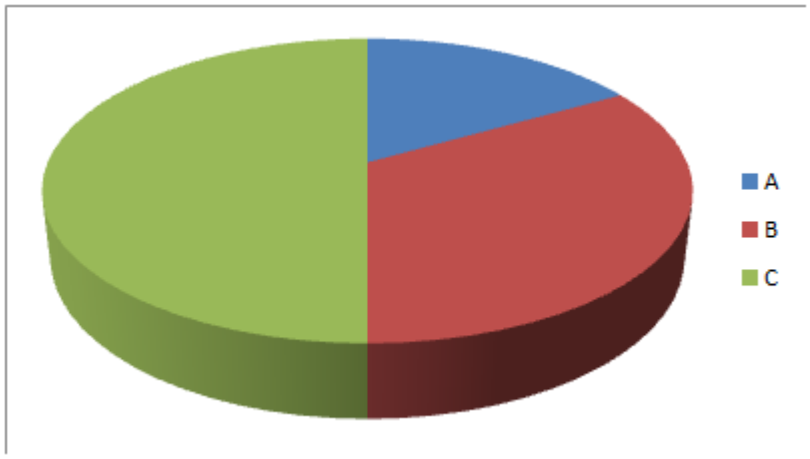
The element that defines the type of chart, barChart in this case, also specify caches for both category axis data (which is really just strings for the categories), as well as the for the numbers, or values, shown on the chart. The cat element defines the string cache for the category axis, and the val element defines the number caches.

Depending on the type of the chart, the plot area may optionally contain elements that define the axes—such as the value axis and category axis—or shape properties. In the example above, the category axis, catAx, and value axis, valAx are defined. These define things like positioning, orientation, label position, and tick marks for the axis. The actual strings and data that the axis corresponds to are defined by the cat and val elements.

Lastly, the chart element contains a legend element which defines the chart legend.

5.13.3 Example

The XML in this clause defines the following 3D chart:



For this example, the xml for the chart element will be shown in detail. The xml for the chart element follows:

```
<c:chart>
  <c:view3D>
    <c:rotX val="30"/>
    <c:perspective val="30"/>
  </c:view3D>
  <c:plotArea>
    <c:layout>
      <c:lastLayoutOuter>
        <c:x val="4.5"/>
        <c:y val="4.5"/>
        <c:w val="324.75"/>
        <c:h val="206.25"/>
      </c:lastLayoutOuter>
      <c:lastLayout>
        <c:x val="10.5"/>
        <c:y val="10.5"/>
        <c:w val="312.75"/>
        <c:h val="194.25"/>
      </c:lastLayout>
    </c:layout>
```

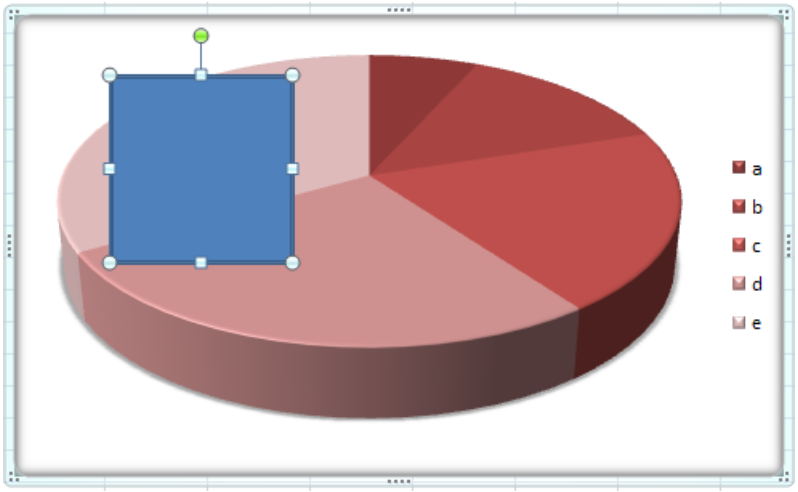
```
<c:pie3DChart>
  <c:varyColors val="1"/>
  <c:ser>
    <c:idx val="0"/>
    <c:order val="0"/>
    <c:cat>
      <c:strRef>
        <c:f>Sheet1!$A$1:$C$1</c:f>
        <c:strCache>
          <c:pt idx="0">
            <c:v>A</c:v>
          </c:pt>
          <c:pt idx="1">
            <c:v>B</c:v>
          </c:pt>
          <c:pt idx="2">
            <c:v>C</c:v>
          </c:pt>
        </c:strCache>
      </c:strRef>
    </c:cat>
    <c:val>
      <c:numRef>
        <c:f>Sheet1!$A$2:$C$2</c:f>
        <c:numCache>
          <c:pt idx="0">
            <c:v>1</c:v>
          </c:pt>
          <c:pt idx="1">
            <c:v>2</c:v>
          </c:pt>
          <c:pt idx="2">
            <c:v>3</c:v>
          </c:pt>
        </c:numCache>
      </c:numRef>
    </c:val>
  </c:ser>
</c:pie3DChart>
```

```
        <c:spPr>
          <a:noFill/>
          <a:ln w="25400">
            <a:noFill/>
          </a:ln>
        </c:spPr>
      </c:plotArea>
      <c:legend>
        <c:legendPos val="r"/>
        <c:layout>
          <c:lastLayout>
            <c:x val="333.75"/>
            <c:y val="81.75"/>
            <c:w val="19.5"/>
            <c:h val="51.75"/>
          </c:lastLayout>
        </c:layout>
      </c:legend>
      <c:plotVisOnly val="1"/>
    </c:chart>
```

5.14 Chart Drawing

5.14.1 Introduction

This subclause provides a high-level overview of the content described in the dml-chartDrawing.xsd schema. The elements in this schema specify how drawing elements are to be described within a chart. For example, suppose we want to specify a rectangle drawing shape within a chart to look like the following:



5.14.2 Overview

The elements that specify the drawing objects within a chart are all located within its respective drawing XML file. This file is located under the "drawings" folder inside the spreadsheet file.

5.14.3 Chart Drawings

Within the drawing*.xml file there is a single drawing file that contains the userShapes element. This element is the parent element for all the drawing elements within a single chart. Its child specifies the anchoring properties of the drawing elements. It is within this element that the main specifications for the drawing elements are located. For example in the above screenshot with a simple shape located in the chart, the XML for this would look like:

```
<c:userShapes>
  <cdr:relSizeAnchor>
    <cdr:from>
      <cdr:x>0.125</cdr:x>
      <cdr:y>0.13194</cdr:y>
    </cdr:from>
    <cdr:to>
      <cdr:x>0.36042</cdr:x>
      <cdr:y>0.53472</cdr:y>
    </cdr:to>
      …
    </cdr:sp>
  </cdr:relSizeAnchor>
</c:userShapes>
```

In the ChartDrawingML code above, there is a single drawing specified almost exactly as it would within the regular DrawingML framework. However, the ChartDrawingML wrapper that is used allows for the specifying of chart specific properties in addition to the normal drawing properties. The most interesting of these are the two anchoring types that define the placement behavior of a drawing within a chart.

5.14.3.1 Anchoring Types

To define a drawing within a chart an anchoring type must be chosen. There are two different anchoring types allowed for use within a chart: Absolute Anchoring and Relative Anchoring. These two types are described in the following two subclauses.

5.14.3.1.1 Absolute Size Anchoring

Absolute Anchoring describes the placement of the drawing within the chart based upon absolute chart coordinates. The absSizeAnchor element specifies anchoring behavior, using percentage-based position coordinates for the anchor location and extent coordinates (in EMUs) for drawing objects, as shown in the example below.<cdr:absSizeAnchor>

```
 <cdr:from>
  <cdr:x>0.125</cdr:x>
```

```
  <cdr:y>0.13194</cdr:y>
 </cdr:from>

    <cdr:ext cx="1238250" cy="1123950"/>
    …
    </cdr:sp>
  </cdr:absSizeAnchor>
```

In this example, there is a single shape specified using absolute anchoring as its anchoring method.

5.14.3.1.2 Relative Size Anchoring

Relative Anchoring describes the placement of the drawing within the chart based upon relative chart coordinates. For instance, if the chart increases in size then the shape will grow as well. This positioning information includes from and to elements which specify a percentage-based coordinate within the chart bounding box. The relSizeAnchor element is what specifies this anchoring behavior and a sample usage is shown below.

```
  <cdr:relSizeAnchor>
    <cdr:from>
      <cdr:x>0.125</cdr:x>
      <cdr:y>0.13194</cdr:y>
    </cdr:from>
    <cdr:to>
      <cdr:x>0.36042</cdr:x>
      <cdr:y>0.53472</cdr:y>
    </cdr:to>
      …
    </cdr:sp>
  </cdr:relSizeAnchor>
```

In this example, there is a single shape specified using relative anchoring as its anchoring method.

5.15 Diagrams

5.15.1 Introduction

This clause provides a high-level overview of the content described in the following schemas: dml-diagramTypes.xsd, dml-diagramDataModel.xsd, dml-diagramStyleDefinition.xsd, dml-diagramLayoutVariables.xsd, dml-diagramElementPropertySet.xsd, dml-diagramColorTransform.xsd, and dml-diagramDefinition.xsd.

The DrawingML diagram file format is broken down into the following subjects:

- Data Model
- Colors
- Quick Styles

- Layout

The best way to understand the above subjects will be to cover them in the ordering above. The seven schemas can be grouped into the subjects as seen in table 1 below.

Data	Colors	Quick Styles	Layout
dml-diagramDataModel.xsd	dml-diagramColorTransform.xsd	dml-diagramStyleDefinition.xsd	dml-diagramTypes.xsd
dml-diagramElementPropertySet.xsd			dml-diagramDefinition.xsd
			dml-diagramLayoutVariables.xsd
			dml-diagramElementPropertySet.xsd

Table 1: DrawingML schemas grouped by subject.

5.15.2 Element Property Set

The schema dml-diagramElementPropertySet.xsd defines a complex type, CT_ElemPropSet, which is a catch-all for holding element properties and customizations, and is used throughout certain complex types in DrawingML. This type contains many properties, and these are explained in subsequent subclauses. The definition of CT_ElemPropSet is as follows:

```
<xsd:complexType name="CT_ElemPropSet">
  <xsd:sequence>
    <xsd:element name="presLayoutVars"
      type="CT_LayoutVariablePropertySet" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="style" type="a:CT_ShapeStyle"
      minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="presAssocID" type="ST_ModelId" use="optional" />
  <xsd:attribute name="presName" type="xsd:string" use="optional" />
  <xsd:attribute name="presStyleLbl" type="xsd:string" use="optional" />
  <xsd:attribute name="presStyleIdx" type="xsd:int" use="optional" />
  <xsd:attribute name="presStyleCnt" type="xsd:int" use="optional" />
```

```
      <xsd:attribute name="loTypeId" type="xsd:string" use="optional" />
      <xsd:attribute name="loCatId" type="xsd:string" use="optional" />
      <xsd:attribute name="qsTypeId" type="xsd:string" use="optional" />
      <xsd:attribute name="qsCatId" type="xsd:string" use="optional" />
      <xsd:attribute name="csTypeId" type="xsd:string" use="optional" />
      <xsd:attribute name="csCatId" type="xsd:string" use="optional" />
      <xsd:attribute name="coherent3DOff" type="xsd:boolean" use="optional" />
      <xsd:attribute name="phldrT" type="xsd:string" use="optional" />
      <xsd:attribute name="phldr" type="xsd:boolean" use="optional" />
      <xsd:attribute name="custAng" type="xsd:int" use="optional" />
      <xsd:attribute name="custFlipVert" type="xsd:boolean" use="optional" />
      <xsd:attribute name="custFlipHor" type="xsd:boolean" use="optional" />
      <xsd:attribute name="custSzX" type="xsd:int" use="optional" />
      <xsd:attribute name="custSzY" type="xsd:int" use="optional" />
      <xsd:attribute name="custScaleX" type="xsd:int" use="optional" />
      <xsd:attribute name="custScaleY" type="xsd:int" use="optional" />
      <xsd:attribute name="custT" type="xsd:boolean" use="optional" />
      <xsd:attribute name="custLinFactX" type="xsd:int" use="optional" />
      <xsd:attribute name="custLinFactY" type="xsd:int" use="optional" />
      <xsd:attribute name="custLinFactNeighborX" type="xsd:int" use="optional" />
      <xsd:attribute name="custLinFactNeighborY" type="xsd:int" use="optional" />
      <xsd:attribute name="custRadScaleRad" type="xsd:int" use="optional" />
      <xsd:attribute name="custRadScaleInc" type="xsd:int" use="optional" />
    </xsd:complexType>
```

5.15.2.1 Presentation Element Properties

The following attributes deal with presentation elements:

- presLayoutVars – The layout variable property set.
- style – The link to the permutation of the style matrix.
- presAssocID – The semantic element associated with this presentation element. This ID is used together with the presName to create a unique key for presentation element indexing.
- presName – The layout node name of this presentation element. This name is used together with presAssocID to create a unique key for presentation element indexing.
- presStyleLbl – The layout node style label of this presentation element..
- presStyleIdx – The layout node style index of this presentation element..
- presStyleCnt – The layout node style count of this presentation element.

5.15.2.2 Document Element Properties

The following attributes deal with the document element:

- loTypeID – The ID of the current diagram type.
- loCatId – The ID of the current diagram category.

- qsTypeID – The ID of the current style type.
- qaCatID – The ID of the current style category.
- csTypeID – The ID of the current color transform.
- csCatID – The ID of the current color transform category.
- coherent3Doff – Enables or disables coherent 3D behavior for styles that have such behavior defined.

5.15.2.3 Semantic Element Properties

The following attributes relate to the semantic element properties:

- phldrT – The text used for display in the element if the placeholder flag is set to `true`. If this field is not set, then the default placeholder text will be used.
- phldr – Indicates that the element is a placeholder or sample item.

5.15.2.4 Customization Properties

The following are customization properties or tweaks:

- custAng – The amount rotation is customized by, in 60,000th of a degree.
- custFlipVert – Vertical flip.
- custFlipHor – Horizontal flip.
- custSzX – Fixed width override for a shape, in emus.
- custSzY – Fixed height override for a shape, in emus.
- custScaleX – Amount that the width is scaled by, in 1,000th of a percent.
- custScaleY – Amount that the height is scaled by, in 1,000th of a percent.
- custT – If text has been customized then layout will no longer change it.
- custLinFactX – A percentage of the shape width that is used for offsetting the shape, in 1,000th of a percent.
- custLinFactY – A percentage of the shape height that is used for offsetting the shape, in 1,000th of a percent.
- custLinFactNeighborX – A percentage of the neighbor's height used for offsetting the shape, in 1,000th of a percent.
- custLinFactNeighborY – A percentage of the neighbor's height used for offsetting the shape, in 1,000th of a percent.
- custRadScaleRad – Defines how much the radius has been scaled by, in 1,000th of a percent.
- custRadScaleInc – Defines how much the include angle has been scaled by, in 1,000th of a percent.

5.15.3 Data Model

The schema dml-diagramDataModel.xsd defines the data model in a diagram. The purpose of the data model is twofold. The first use of the data model is to hold the information contained in a diagram. For example, in figure 1 below, the purpose of the data model would be to hold the information, "one", "two" and "three" for the diagram.

Figure 11: Example diagram with data.

The second use of the data model is to define an initial state of the diagram. This initial state consists of what can be thought of as placeholder data, which an application uses to display a diagram initially before any data has been entered. Figure 2 shows an example of what a diagram might look like in an initial state containing three empty nodes. In this example, the placeholder data consists of three nodes and two connections, which will be explained shortly.

Figure 12: An empty diagram in its initial state.

5.15.3.1 Structural Elements

5.15.3.1.1 Element Type

There is a single simple type, ST_PtType, used to define a type of element; this is defined later. Element types hold the data associated with a diagram and are defined in relation to one-another through relationship types. Seven different types of elements are available to the user:

- doc – A document element. The document element is the root element within a diagram and can be thought of as the canvas which the diagram is drawn on.
- node – A model element. This is the basic element type which is used and can be used to hold text for example.
- asst – This is used in hierarchy diagrams and represents an assistant element.
- pres – A presentation element. This element defines the visual aspects associated with a node, or rather the presentation aspects of an element.
- parTrans – A parent transition element. This element holds the data for a parent-child relationship between two elements of type node.

- sibTrans – A sibling transition element. This element holds the data for the relationship defined between two elements of type node whom are peers of one another.
- unknown – An element type that is used to maintain backward compatibility.

5.15.3.1.2 Relationship Type

There are defined relationships or connections between two model elements. Four types of relationships are defined in the simple type ST_CxnType:

- parOf – Parent-child relationship.
- presOf – Presentation relationship.
- presParOf – Presentation parent of relationship.
- unknownRelationship – An unknown relationship type.

5.15.3.1.3 Element

An element is a single item, such as a node or transition in the data model. Within the realm of DrawingML, the complex type CT_Pt holds information describing an element within a diagram. Within this description lies both the data held within the element, and any formatting on that element. A CT_Pt is defined as follows:

```
<xsd:complexType name="CT_Pt">
  <xsd:sequence>
    <xsd:element name="prSet" type="CT_ElemPropSet" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="spPr" type="a:CT_ShapeProperties"
      minOccurs="0" maxOccurs="1 />
    <xsd:element name="style" type="a:CT_ShapeStyle"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="t" type="a:CT_TextBody" minOccurs="0"
      maxOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="modelId" type="ST_ModelId" use="required" />
  <xsd:attribute name="type" type="ST_PtType" use="optional"
    default="node" />
  <xsd:attribute name="cxnId" type="ST_ModelId" use="optional"
    default="0" />
</xsd:complexType>
```

The attribute modelId holds a unique id for a particular element. This unique id can be referenced elsewhere, for example, from within a connection list. This attribute is required for every point defined in the data model.

The last two attributes of the CT_Pt are optional. The first defines the type of point with the default being a node. The second defines a connection id. This connection id is only used if the point type is of type parTrans, or sibTrans. The connection id refers to a relationship that is defined elsewhere in the data model.

5.15.3.1.4 Relationship

A relationship is a connection between any two model elements. An example of where a relationship would be used can be seen in figures 1 and 2. In each of those examples, the arrows between the nodes have relationships defined. A relationship is defined as follows:

```
<xsd:complexType name="CT_Cxn>
  <xsd:attribute name="modelId" type="ST_ModelId" use="required" />
  <xsd:attribute name="type" type="ST_CxnType" use="optional"
    default="parOf" />
  <xsd:attribute name="srcId" type="ST_ModelId" use="required" />
  <xsd:attribute name="destId" type="ST_ModelId" use="required" />
  <xsd:attribute name="srcOrd" type="xsd:unsignedInt" use="required" />
  <xsd:attribute name="destOrd" type="xsd:unsignedInt" use="required" />
  <xsd:attribute name="parTransId" type="ST_ModelId" use="optional"
    default="0" />
  <xsd:attribute name="sibTransId" type="ST_ModelId" use="optional"
    default="0" />
  <xsd:attribute name="presId" type="xsd:string" use="optional"
    default="" />
</xsd:complexType>
```

The relationship, as with the element, has a unique id associated with it referred to as the modelID. The srcId and destId attributes refer to ids of the source element and destination element, respectively, that this relationship is defined between.

The srcOrd and destOrd refer to the ordinality of siblings for a given connection. For example, if a node had three siblings, A, B, and C, then the srcOrd would define if they were to show up as A, B, C, or perhaps B, C, and then A.

The presId attribute contains the presentation that is associated with this particular relationship.

5.15.3.1.5 Element List

The complex type CT_PtList is simply a sequence of elements. Its definition is as follows:

```
<xsd:complexType name="CT_PtList">
  <xsd:sequence>
    <xsd:element name="pt" type="CT_Pt" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

5.15.3.1.6 Relationship List

This complex type, CT_CxnList, is simply a sequence of connections. Its definition is as follows:

```
<xsd:complexType name="CT_CxnList" oxsd:cname="Relationships">
  <xsd:sequence>
    <xsd:element name="cxn" type="CT_Cxn" minOccurs="0"
      maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>
```

5.15.3.1.7 Data Model

The complex type CT_DataModel defines the data model and contains a sequence of elements. It is defined as follows:

```
<xsd:complexType name="CT_DataModel">
  <xsd:sequence oxsd:emitArgs="flattenSequence">
    <xsd:element name="ptLst" type="CT_PtList" />
      <xsd:element name="cxnLst" type="CT_CxnList" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="bg" type="a:CT_BackgroundFormatting"
        minOccurs="0" />
      <xsd:element name="whole" type="a:CT_WholeE2oFormatting"
        minOccurs="0" />
    </xsd:sequence>
</xsd:complexType>
```

The data model contains a list of elements, a list of connections, and formatting properties for the background object and the diagram container. This complex type is responsible for holding all data-bound information of the diagram being created.

5.15.4 Color Transforms

Color transforms define how colors are applied to diagrams. Color transforms define how color is used in the diagram as a whole, and they mandate things such as which theme color or colors will be used, if there is a tint or shade applied to a certain color or part of the diagram, or if color is even used at all. Some examples of what color transforms can do to a simple diagram can be seen in figure 3.

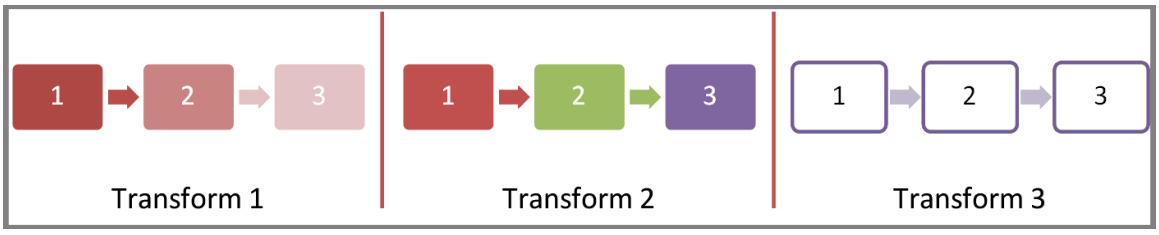


Figure 13: Different examples of a color transform applied to a diagram

5.15.4.1 Structural Elements

The structural elements which come together to create a color transform, or rather, the complex type CT_ColorTransform, are as follows:

- CT_CTName
- CT_CTDescription
- CT_CTCategory
- CT_CTCategories
- ST_ClrAppMethod
- ST_HueDir
- CT_Colors
- CT_CTStyleLabel
- CT_CTVersion
- CT_ColorTransformHeader
- CT_ColorTransformHeaderLst

The complex types CT_CTName (name), CT_CTDescription (description), CT_CTCategory (category), and CT_CTCategories (list of categories) work together to name, describe and categorize the particular color transform. These types are mirrored elsewhere throughout DrawingML in the different subjects in order to perform the same tasks of naming, describing, and categorizing.

The name consists simply of two strings, one of a name for the color transform, which is required, and an optional language tag. The language allows someone to specify a language for a given title. It is possible to specify multiple titles that are language dependant. The description also has the optional language attribute as in the name, along with a second required string attribute which holds the actual description. The usage of this is exactly the same as within CT_CTName. CT_CTName and CT_CTDescription are defined in the following way:

```
<xsd:complexType name="CT_CTName">
  <xsd:attribute name="lang type="xsd:string" use="optional" />
  <xsd:attribute name="val" type="xsd:string" use="required" />
</xsd:complexType>
<xsd:complexType name="CT_CTDescription">
  <xsd:attribute name="lang" type="xsd:string" use="optional" />
  <xsd:attribute name="val" type="xsd:string" use="required" />
</xsd:complexType>
```

The category and categories complex types , CT_CTCategory and CT_CTCategories, respectively, define how the color transform is categorized within the user interface of the application. The category contains a name, or type, along with a priority that defines the ordering of the color transform. The lower the priority, the earlier in the category it will display. If there is a tie, the unique id associated with the color transform will decide the order alphabetically. CT_CTCategories is simply a list of CT_CTCategory. The two complex types are defined as follows:

```
<xsd:complexType name="CT_CTCategory">
  <xsd:attribute name="type" type="xsd:anyURI" use="required" />
  <xsd:attribute name="pri" type="xsd:unsignedInt" use="required"/>
</xsd:complexType>
<xsd:complexType name="T_CTCategories">
  <xsd:sequence minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="cat" type="CT_CTCategory" minOccurs="0"
      maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>
```

5.15.4.1.1 Color Application Method

The simple type ST_ClrAppMethod lists the different options for color to be applied to a diagram. There are three options available to the user: span, cycle, and repeat. Given a list of colors, which go from color A to color B, the differences in these three options can be defined. These options are shown below in Figure 4.

The span option will, from the start of the diagram to the end of the diagram, interpolate between the colors A through B for every node along the way.

The cycle option will interpolate from A to B then back to A from the start of the diagram to the end of the diagram.

The repeat option applies colors A through B one at a time for each point in the diagram, then repeats A through B as needed.

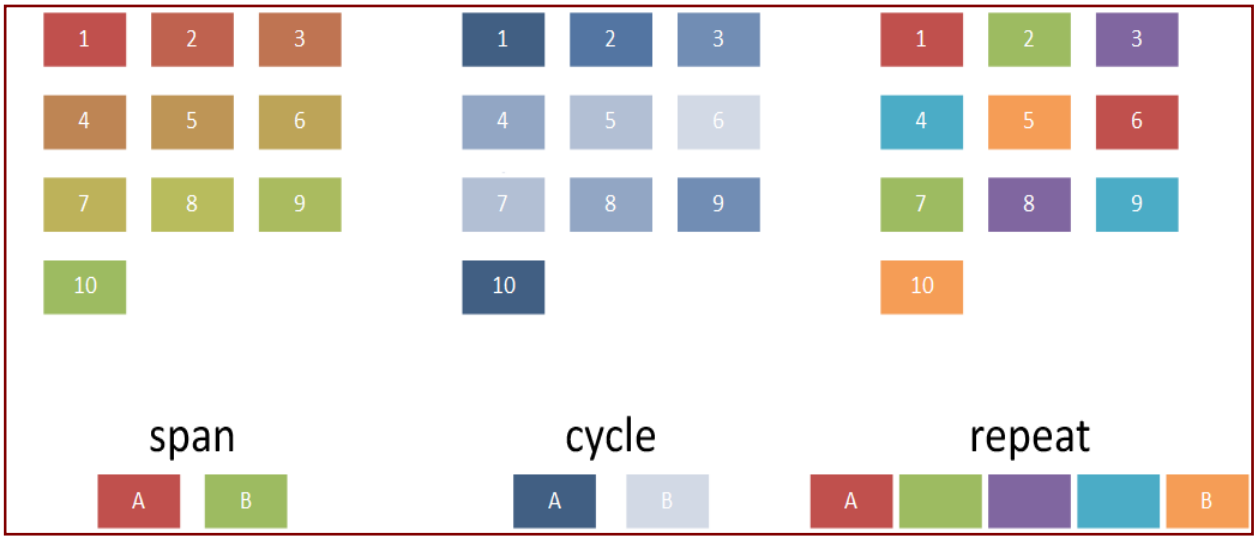


Figure 14: Examples of the three different ways a color transform is applied to a diagram.

5.15.4.1.2 Hue Direction

The simple type ST_HueDir defines the direction of a hue color shift around a color wheel. A user can either define the shift to occur in the clockwise (cw) direction, or in the counterclockwise (ccw) direction. For

example, in Figure 4, the span colors are red and green. The behavior shown in figure 4 is a shift in the cw direction. If the hue shift had been defined in the ccw direction, the colors interpolated between colors A and B would have been in the hues purple and blue. Another example of a hue direction shift in the clockwise can be seen in Figure 5 below along with the color shift from red to yellow and then from yellow to blue along the primary colors. Counterclockwise shifts would occur in the direction of yellow to red and blue to yellow in the examples below.

Figure 15: Example hue shifts in the clockwise direction around a color wheel applied to two diagrams. The three primary colors, red, yellow, and blue are used to represent the three major sections of a color wheel which ranges between red to yellow to blue to red.

5.15.4.1.3 Colors

The complex type CT_Colors holds the actual color values that are to be applied to a given diagram and how those colors are to be applied. It contains the color application method and hue shift direction, and is defined as follows:

```
<xsd:complexType name="CT_Colors>
  <xsd:sequence>
    <xsd:group ref="a:EG_ColorChoice" minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
  <xsd:attribute name="meth" type="ST_ClrAppMethod" use="optional"
    default="span" />
  <xsd:attribute name="hueDir" type="ST_HueDir" use="optional"
    default="cw" />
</xsd:complexType>
```

The sequence of colors is defined via the sequence of EG_ColorChoices.

5.15.4.1.4 Style Label

The complex type CT_CTStyleLabel packages together colors for the different pieces of a diagram. There are six different aspects to a diagram that can be colored independently of one another. Each of the six parts is of type CT_Colors. They are:

- Fill Colors – The colors that actually fill the shapes in the diagram
- Line Colors – The colors of the lines on the shapes in the diagram.
- Effect Colors – The colors of the effects applied to the shapes within the diagram (eg. Glow).
- Text Line Colors – The colors of the lines on the text within the diagram.
- Text Fill Colors – The color of the text within the diagram.
- Text Effect Colors – The colors of the effects applied to the text within the diagram.

The final piece of a style label is simply its name, which is a string. CT_CTStyleLabel is defined as follows:

```
<xsd:complexType name="CT_CTStyleLabel">
  <xsd:sequence>
    <xsd:element name="fillClrLst" type="CT_Colors"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="linClrLst" type="CT_Colors"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="effectClrLst" type="CT_Colors"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="txLinClrLst" type="CT_Colors"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="txFillClrLst" type="CT_Colors"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="txEffectClrLst" type="CT_Colors"
      minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required" />
</xsd:complexType>
```

5.15.4.1.5 Version

The simple type ST_CTVersion defines the minimum version of an application that the color transform will work with. The version corresponds to build numbers in the major.minor.build.revision format and is defined as follows:

```
[0-9]?[0-9])?(\.[0-9]?[0-9])?(\.[0-9]{4})?(\.[0-9]{4}
```

5.15.4.1.6 Color Transform

The complex type CT_ColorTransform brings together all of the pieces into one cohesive structure. This is the actual definition of a color transform, which can be applied to any diagram; it is defined as follows:

```
<xsd:complexType name="CT_ColorTransform">
  <xsd:sequence>
    <xsd:element name="title" type="CT_CTName" minOccurs="0"
      maxOccurs="unbounded" />
    <xsd:element name="desc" type="CT_CTDescription"
      minOccurs="0" maxOccurs="unbounded" />
    <xsd:element name="catLst" type="CT_CTCategories"
      minOccurs="0" />
    <xsd:element name="styleLbl type="CT_CTStyleLabel"
      minOccurs="0" maxOccurs="unbounded" odoc />
  </xsd:sequence>
  <xsd:attribute name="uniqueId" type="xsd:anyURI" use="optional"/>
  <xsd:attribute name="minVer" type="ST_CTVersion" use="optional"
    default="12.0" />
</xsd:complexType>
```

A color transform contains a title, description, category list, and style label in a sequence along with a unique id and a minimum version.

5.15.4.2 Color Transform Header

Two complex types, CT_ColorTransformHeader and CT_ColortransformHeaderLst, help alleviate potential performance concerns with initially loading in a large number of color transforms. The header information contains the minimum information required to load a color transform into the application. Because of this, color transforms themselves can be loaded only when needed, and other initialization work can progress quickly without loading unneeded information.

CT_ColorTransformHeader contains information about the title of the color transform, the description, how it is categorized, the unique id, minimum version, and resId. It is defined in the following way:

```
<xsd:complexType name="CT_ColorTransformHeader">
  <xsd:sequence>
    <xsd:element name="title" type="CT_CTName" minOccurs="1"
      maxOccurs="unbounded" />
    <xsd:element name="desc" type="CT_CTDescription"
      minOccurs="1" maxOccurs="unbounded" />
    <xsd:element name="catLst" type="T_CTCategories"
      minOccurs="0" o:cname="categories" />
  </xsd:sequence>
  <xsd:attribute name="uniqueId" type="xsd:anyURI" use="required"/>
  <xsd:attribute name="minVer" type="ST_CTVersion" use="optional"
    default="12.0" />
  <xsd:attribute name="resId" type="xsd:int" use="optional"
  default="0" />
</xsd:complexType>
```

The complex type CT_ColorTransformHeaderLst simply contains a list of color transform headers. It is defined as follows:

```
<xsd:complexType name="CT_ColorTransformHeaderLst">
  <xsd:sequence>
    <xsd:element name="colorsDefHdr" type="CT_ColorTransformHeader"
      minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>
```

5.15.5 Style Definition

A style definition is similar to a color transform. A style definition defines items such as the font used, the thickness of a contour line, 3-D properties on the diagram, among other things. Style definitions work in combination with color transforms to give an overall look and feel for the diagram. Some examples of style definitions in use are in figure 5.

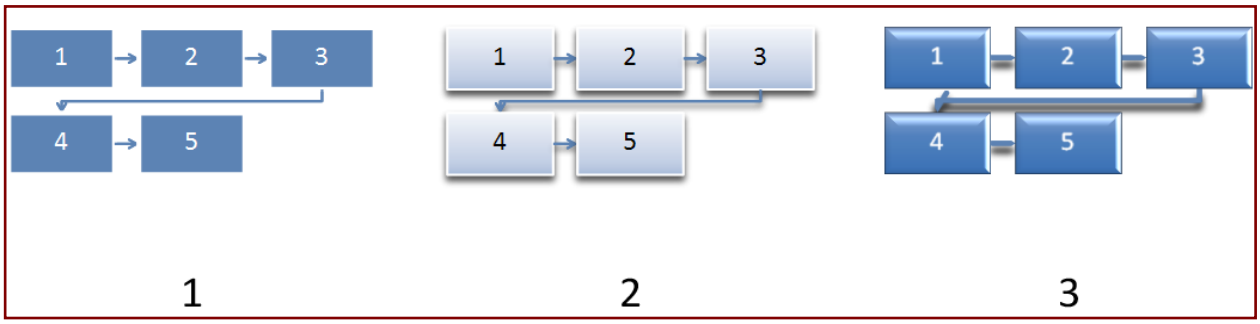


Figure 16: Examples of using three different style definitions on a diagram.

5.15.5.1 Structural Elements

The structural elements which come together to create a style definition are as follows:

- CT_SDName
- CT_SDDescription
- CT_SDCategory
- CT_SDCategories
- CT_TextProps
- CT_StyleLabel
- ST_SDVersion
- CT_StyleDefinition
- CT_StyleDefinitionHeader
- CT_StyleDefinitionHeaderLst

CT_SDName, CT_SDDescription, CT_SDCategory, CT_SDCategories, and ST_SDVersion are all defined exactly as they are within color transforms. These types were recreated within the style definition area to allow slight differentiations to be made, although, at this time they are defined exactly the same.

The text properties, style label, and then the style definition combine together to create a style definition, which is applied to a diagram.

5.15.5.1.1 Text Properties

The complex type CT_TextProps holds any 3-D associated properties of the text that is to be held in the diagram. A CT_TextProps is defined as follows:

```
<xsd:complexType name="CT_TextProps">
  <xsd:sequence>
    <xsd:group ref="a:EG_Text3D" minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
</xsd:complexType>
```

All that is contained within the text properties is an EG_Text3D complex type. The usage of the text properties complex type allows 3-D text properties to be defined for a diagram style.

5.15.5.1.2 Style Label

The style label contains information pertaining to the styleable elements within a diagram. These elements include the shape properties and text properties along with any references to the style matrix or document theme. The shape properties are defined in two different ways: the scene3d element that pertains to the scene on a whole (and includes lighting effects, rotations, and the like), and any 3-D and material settings are held in the sp3d element. CT_StyleLabel is defined as follows:

```
<xsd:complexType name="CT_StyleLabel">
  <xsd:sequence>
    <xsd:element name="scene3d" type="a:CT_Scene3D"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="sp3d" type="a:CT_Shape3D" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="txPr" type="CT_TextProps" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="style" type="a:CT_ShapeStyle"
      minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required" />
</xsd:complexType>
```

As with many other complex types, the style label has an attribute reserved for a name. This is simply a string that names the particular style label. This style label can then be referenced from within a diagram definition, as we shall see below.

Note that the scene3d element contained within a style label acts on the level of an individual shape, rather than the diagram as a whole. A second scene3d element is defined within the style definition; that acts on the diagram level and allows for scene coherent 3-D to be applied to the diagram. A style definition contains a style label.

5.15.5.1.3 Style Definition

The style definition complex type, CT_StyleDefinition, is the root element used to define a style definition. As with the root element of a color transform, within the style definition there exists a title, description, category, unique id, and minimum version number. These elements serve the same purpose as they do within the color transform.

The interesting aspects of a style definition complex type are that it holds a style label, another style index (which is contained within the style label as well), and another scene3d (which is also contained within the style label as has been previously mentioned). The scene3d element applies to the diagram on a whole and allows for scene coherent 3-D to be applied to the diagram. The duplication of the style index is two-fold. If a style index is not defined within the style label, then the default style index, or rather, the index defined in this complex typem is used. Since a diagram definition can reference a style label, and not a style definition, the style index is also required within the style label. A CT_StyleDefniition is defined as follows:

```
<xsd:complexType name="CT_StyleDefinition">
  <xsd:sequence>
    <xsd:element name="title" type="CT_SDName" minOccurs="0"
      maxOccurs="unbounded" />
    <xsd:element name="desc" type="CT_SDDescription"
  minOccurs="0" maxOccurs="unbounded" />
    <xsd:element name="catLst" type="CT_SDCategories"
      minOccurs="0" />
    <xsd:element name="scene3d" type="a:CT_Scene3D"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="style" type="a:CT_ShapeStyle"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="styleLbl" type="CT_StyleLabel"
      minOccurs="1" maxOccurs="unbounded" />
  </xsd:sequence>
  <xsd:attribute name="uniqueId" type="xsd:anyURI" use="optional"/>
  <xsd:attribute name="minVer" type="ST_SDVersion" use="optional" `
    default="12.0" />
</xsd:complexType>
```

5.15.5.1.4 Style Definition Header

The complex types CT_StyleDefinitionHeader and CT_StyleDefinitionHeaderLst perform the same function as the two complex header types in color transforms. They are used to pre-load required information so the actual loading of the style definition can happen only when needed.

5.15.6 Layout

The single largest aspect of DrawingML is Layout. Ultimately, layout is responsible for defining all aspects of the diagram outside of color and style. It defines how the diagram looks, how it behaves, and how the data is to be mapped. Figure 6 shows different examples of how layout works to create a diagram holding the data '1', '2', and '3'.

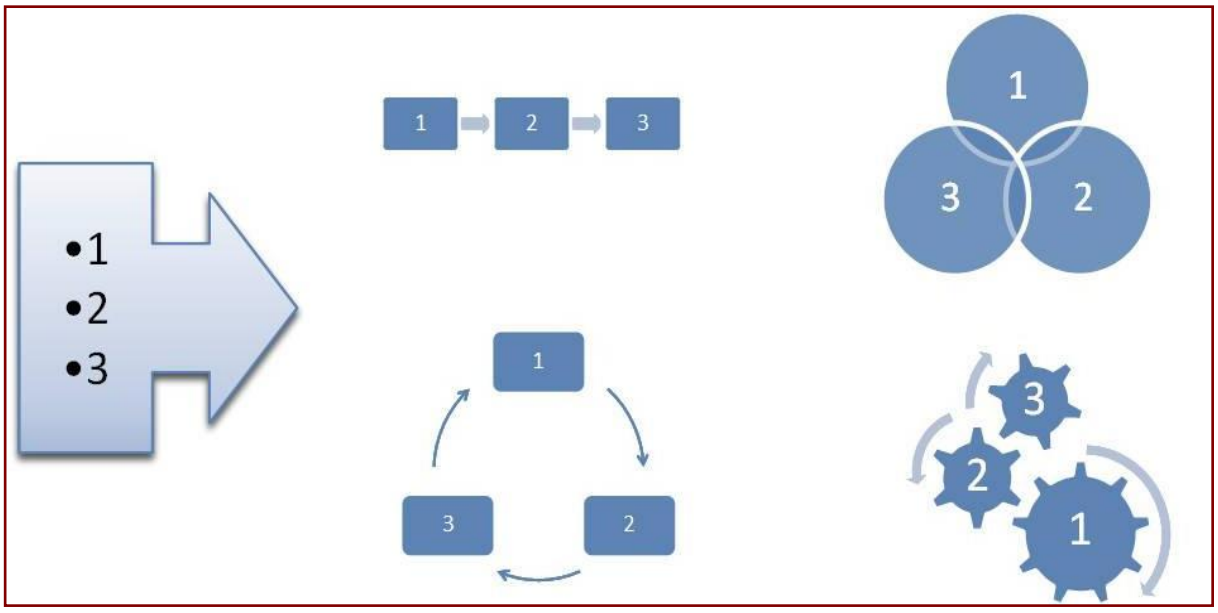


Figure 17: Different layouts mapping the data 1, 2, and 3 to four different diagrams

There are two aspects to layout: defining the numerous complex types utilized by diagram definitions, and the diagram definitions themselves. Diagram definitions are a fundamental part of layout, and utilize everything in order to define new diagrams.

5.15.6.1 Basic Layout Types

There are a very large number of simple types defined in this subject. These simple types are all associated with properties of a diagram that can be modified to create a desired behavior. These simple types, along with an explanation of what each does, follow.

5.15.6.1.1 Algorithm Type

ST_AlgorithmType is responsible for which algorithm will be used to layout the diagram. The algorithm layout chosen determines if the diagram behaves as if it were a simple list, a circular cycle, or some other type of diagram. The algorithms available are:

- unknown – Unknown algorithm type. This type is used for extensibility reasons. Use of this algorithm type can used by future versions/implementations which define new algorithm types other than the above mentioned algorithm types. By using this type, the current implementations know that they will not be able to correctly render the diagram if they have only implemented the above mentioned algorithms for layout.

- composite – The composite algorithm specifies the size and position for all child layout nodes. You can use it to create graphics with a predetermined layout or in combination with other algorithms to create more complex shapes.
- conn – the connector algorithm lays out and routes connecting lines, arrows, and shapes between layout nodes
- cycle – the cycle algorithm lays out child layout nodes around a circle or portion of a circle using equal angle spacing
- hierChild – the hierarchy child algorithm works with the hierRoot algorithm to create hierarchical tree layouts. This algorithm aligns and positions its child layout nodes in a linear path under the hierRoot layout node.
- hierRoot – the hierarchy root algorithm works with the hierChild algorithm to create hierarchical tree layouts. The hierRoot algorithm aligns and positions the hierRoot layout node in relation to the hierChild layout nodes.
- pyra – the pyramid algorithm lays out child layout nodes along a vertical path and works with the trapezoid shape to create a pyramid
- lin – the linear algorithm lays out child layout nodes along a linear path
- sp – the space algorithm is used to specify a minimum space between other layout nodes or as an indication to do nothing with the layout node's size and position
- tx – manges layout of text within a shape
- snake – the snake algorithm lays out child layout nodes along a linear path in two dimensions, allowing the linear flow to continue across multiple rows or columns

5.15.6.1.2 Axis Type

The simple type ST_AxisType defines how layout maps data to the diagram for a given point in the diagram. The different ways this can be mapped is as follows:

- self – the layout maps to the current data point
- ch – the layout node can map to the children of the current data point, but not to descendants lower in the hierarchy
- des – the layout node can map to a descendant of the current data point
- desOrSelf – the layout node can map the current data point, or to a descendant of the current data point
- par – the layout node maps to the parent data point
- ancst – the layout node can map to the ancestors of the current data point (parents, grandparents, great grandparents, etc)
- ancstOrSelf – the layout node can map an ancestor data point or the current data point
- followSib – the layout node can map to a following sibling peer to the current data point
- precedSib – the layout node can map to a preceding sibling peer to the current data point.
- root – the layout node can map to the root
- none – the layout node doesn't map to any data point

5.15.6.1.3 Boolean Operators

Boolean type operators, defined by ST_BoolOperator, are for layout.

- none – no operator defined
- equ – 'equal to' operator, returns true if the two compared values are equal, false otherwise
- gte – 'greater than or equal to' operator
- lte – 'less than or equal to' operator

5.15.6.1.4 Child Order Type

The simple type ST_ChildOrderType specifices the child order type for a layout node.

- b – bottom
- t – top

5.15.6.1.5 Constraint Types

The simple type ST_ConstraintTypes defines the constraints that can be used as limits or modifications to the diagram or the nodes held within the diagram. These constraints manage the behavior of many properties that can be defined within the diagram. The different constraints that can be applied to a diagram are as follows:

- unknown – An unknown constraint. This can be used by implementing applications to define a constraint type outside of the scope of what is defined by this simple type.
- alignOff – specifies the alignment offset
- begMarg – specifies the beginning margin
- bendDist – specifies the distance at which a connector will bend
- begPad – specifies the distance between the edge of the transition node and the connector shape
- b – bottom alignment
- bMarg – the bottom margin
- bOff – specifies the amount of offset relative to the bottom of the node
- ctrX – specifies the center of the node in the X-direction
- ctrXOff – specifies the amount of offset of the center of the node in the X-direction
- ctrY – specifies the center of the node in the Y-direction
- ctrYOff – specifies the amount of offset of the center of the node in the Y-direction
- connDist – specifies the distance between connectors. Intended for use with boolean and reference constraints. Overrides values specified in the layout definition part
- diam – specifies the diameter, and is used within the cycle algorithm type
- endMarg – specifies the end margins
- endPad – specifies the distance between the edge of the transition node and the connector shape
- h – specifies the height
- hArH – specifies the arrowhead height
- hOff – specifies the height offset

- l – specifies the left
- lMarg – specifies the amount of left margin
- lOff – specifies the amount of left offset
- r – specifies right
- rMarg – specifies the amount of right margin
- rOff – specifies the amount of right offset
- primFontSz – specifies the primary font size
- pyraAcctRatio - specifies the fraction of the width of the diagram that is reserved for the fly-outs at their shortest distance
- secFontSiz – specifies the secondary font size
- stemThick – specifies the thickness of the shaft on an arrow
- t – specifies the top
- tMarg – specifies the amount of top margin
- tOff – specifies the amount of top offset
- userA through userZ – User defined information. This set of enumerations allow a user to define specific values which can be referenced at a later time within a diagram definition. For example, if the value 5.1 was important to the layout and definition of a diagram, the user could define userA to be equal to 5.1 and then use the userA constraint within the diagram definition. This would allow a single place to update the value across the entire diagram definition.
- w – specifies the width
- wArH – specifies the width of an arrowhead
- wOff – specifies the amount of width offset

5.15.6.1.6 Constraint Relationships

The simple type ST_ConstraintRelationship defines the application of retrieval data the constraint is applied to. The following relationships are available:

- self – the constraint is applicable to the current point
- ch – the constraint is applicable to a child of the current point
- des – the constraint is applicable to a descendant of the current point

5.15.6.1.7 Element Type

The simple type ST_ElementType defines the type of element, or point which get created and how they are created from the data at hand. The different ways to pull from the data to create points are as follows:

- all – use all of the data points, nodes and transitions
- doc – use the document level, or root data point
- node – use only data nodes input by the user
- norm – in place for extensibility and behaves exactly opposite of the asst element type
- nonNorm – in place for extensibility and behaves exactly opposite of the nonAsst element type
- asst – use assistant data nodes within hierarchy algorithm

- nonAsst – use non-assistant nodes within the hierarchy algorithm
- parTrans – Use only parent transitions between nodes. Parent transitions are similar to sibling transitions, except that they represent parent/child relationships. Parent transitions are most commonly used in hierarchy diagrams, such as organization charts, to draw lines between parent and child nodes.
- pres – specifies that the node is related to the presentation level
- sibTrans – Use only sibling transitions between data nodes. These transitions represent sibling relationships between nodes, and are frequently mapped to arrows between shapes in the drawing. A sibTrans value is sometimes used to create white space between nodes.

5.15.6.1.8 Parameter ID

The simple type ST_ParameterId defines parameters that can be used to modify the behavior or algorithms. The modifications are as follows:

- horzAlign – specifies the horizontal alignment
- vertAlign – specifies the vertical alignment
- chDir – specifies the child direction
- chAlign – specifies the alignment of the children
- secChAlign – specifies a secondary child alignment
- linDir – specifies whether children are arranged from left to right, right to left, top to bottom, or bottom to top
- secLinDir – specifies a secondary linear direction in which children are
- arranged from left to right, right to left, top to bottom, or bottom to top
- stElem – specifies the point type of the layout node to use as the first shape in the cycle
- bendPt – specifies where the bend point is to be located on connectors
- connRout – specifies whether the connector is drawn as a single straight line, orthogonal lines with a single bend, or a curve that uses the diam constraint
- begSty – specifies whether the beginning of the connector has an arrowhead
- endSty – specifies whether the end of the connector has an arrowhead
- dim – specifies the connector dimension, 2-D, 3-D, or custom
- rotPath - if rotPath=`alongPath`, the algorithm rotates all children perpendicular to the line from the cycle's center to the child node; otherwise they are not rotated. The `alongPath` value does not take rotation into account when determining if shapes overlap
- ctrShpMap - `None` specifies to place nodes around a circle. First node (`fNode`) specifies to place the first node in the center and the remaining nodes around the circle
- nodeHorzAlign – specifies the horizontal node alignment
- nodeVertAlign – specifies the vertical node alignment
- fallback – specifies if the fallback occurs in a single dimension (e.g. vertically) or if it occurs in two dimensions
- txDir – specifies where the text of the first node starts
- pyraAcctPos – specifies the placement of the fly-out grandchildren

- pyraAcctTxMar – specifies the placement of one edge of the child text
- txBlDir – Specifies the text block direction, vertical or horizontal
- txAnchorHorz – Specifies the horizontal text anchor position.
- txAnchorVert – Specifies the vertical text anchor position.
- txAnchorHorzCh – Specifies the horizontal text anchor position for child text.
- txAnchorVertCh – Specifies the vertical text anchor position for child text.
- parTxLTRAlign – Specifies the paragraph alignment of parent text when the shape has only parent text. This parameter applies when the text direction is left to right.
- partTxRTLAlign – Specifies the paragraph alignment of parent text when the shape has only parent text. This parameter applies when the text direction is right to left.
- shpTxLTRAlignCh – Specifies the paragraph alignment of all text within the shape when the shape contains both parent and child text. This parameter applies when the text direction is left to right.
- shpTxRTLAlignCh – Specifies the paragraph alignment of all text within the shape when the shape contains both parent and child text. This parameter applies when the text direction is right to left.
- autoTxRot – specifies how text is oriented relative to the shape
- grDir – Specifies the direction of growth for the snake algorithm.
- flowDir – specifies weather nodes are arranged in rows or columns for the snake algorithm.
- contDir – Specifies the direction of subsequent rows or columns in the snake algorithm.
- bkpt – specifies the point at which the diagram starts to snake
- off – specifies whether each row and column is centered or offset from the previous row or column
- hierAlign – specifies the alignment of the hierarchy
- bkPtFixedVal – specifies where the sname should break if bkpt is set to fixed
- stBulletLvl – Specifies the level at which to start using bullets for incoming text.
- stAng – Specifies the angle at which the first shape is placed. Angles are in degrees, measured clockwise from a line pointing straight upward from the center of the cycle.
- spanAng – Specifies the angle the cycle spans. Final shapealign text is placed at stAng+spanAng, unless spanAng=360. In that case, the algorithm places the text so that shapes do not overlap
- ar – Specifies the aspect ratio (width to height) of the composite node to use when determining child constraints. A value of 0 means leave the width and height constraints as is. The algorithm may temporarily shrink one dimension to achieve that ratio
- lnSpPar – specifies the line spacing of the parent
- lnSpAfParP – specifies the line spacing after the parent paragraph
- LnSpCh specifies the line spacing of a child
- lnSpAfChP – specifies the line spacing after the child paragraph
- rtShortDist – Specifies the routing to use the shortest distance for connectors.
- alignTx – specifies if to hold text or not
- pyraLvlNode – If pyramid has a composite child node, specifies the name of the node that is a child of the composite that makes up the pyramid itself. If the node specifies a trapezoid shape, it modifies the adjustment handles to construct a pyramid.

- pyraAcctBkgdNode – If pyramid has a composite child node, specifies the child node that should hold the child text.
- pyraAcctTxNode – If pyramid has a composite child node, specifies the name of the node that is a child of the composite that makes up the child flyout shape.
- srcNode – Specifies the name of the layout node from which to start the connection.
- dstNode – Specifies the name of the layout node from which to end the connection from.
- begPts – Specifies the point type for the beginning of a connector.
- endPts – Specifies the point type for the end of a connector.
- unknown – An unkown parameter id. This can be used by implementing applications to define a parameter id outside of the scope of what is defined by this simple type.

5.15.6.1.9 Function Type

The simple type ST_FunctionType defines different types of conditional expressions that can be utilized. The different types of expressions are:

- cnt – Specifies the count of items.
- pos - Retrieves the position of the node in the specified set of nodes.
- revPos - Reverse position function.
- posEven - Returns 1 if the specified node is at an even numbered position in the data model.
- posOdd - Returns 1 if the specified node is in an odd position in the data model.
- var - Used to reference a variable.
- depth - Specifies the depth of items.
- maxDepth - Defines the maximum depth of items.

5.15.6.1.10 Function Operator

The simple type ST_FunctionOperator defines the different condition expression operators that can be used. The different operators are as follows:

- equ – equal
- neq – not equal
- gt – greater than
- lt – less than
- gte – greater than or equal to
- lte – less than or equal to

5.15.6.1.11 Horizontal Alignment

The simple type, ST_HorizontalAlignment, specifies the different options available for alignment horizontally. The options are:

- l – left
- ctr – center
- r – right

- none – none

5.15.6.1.12 Vertical Alignment

The simple type, ST_VerticalAlignment, specifies the different options available for alignment vertically. The options are:

- t – top
- mid – middle
- b – bottom
- none – none
- Child Direction

The simple type ST_ChildDirection is used to specify the direction the children are laid out. The different options are:

- horz – horizontally
- vert – vertically

5.15.6.1.13 Child Alignment

The simple type ST_ChildAlignment defines the alignment parameter types for children. The different types are:

- t – top
- b – bottom
- l – left
- r – right

5.15.6.1.14 Secondary Child Alignment

The simple type ST_SecondaryChildAlignment defines secondary alignment parameter types for children. The simple type ST_ChildAlignment is mirrored here with the addition of the none type.

5.15.6.1.15 Linear Direction

The simple type ST_LinearDirection defines the linear direction parameter types. The types are as follows:

- fromL – from left
- fromR – from right
- fromT – from top
- fromB – from bottom

5.15.6.1.16 Secondary Linear Direction

The simple type ST_SecondaryLinearDirection defines a secondary linear direction parameter. This simple type mirrors exactly the simple type ST_LinearDirection with the addition of the none type.

5.15.6.1.17 Starting Element

The simple type ST_StartingElement specifies the first node point type for a cycle diagram. The different starting elements are:

- node – node
- trans – transition

5.15.6.1.18 Rotation Path

The simple type ST_RotationPath specifies the way in which the algorithm rotates children. The different rotation types are:

- none – no rotation is performed
- alongPath – the children are rotated perpendicular to the line from the cycle's center to the child node

5.15.6.1.19 Center Shape Mapping

The simple type ST_CenterShapeMapping specifies how the first node of a cycle diagram is laid out within the diagram. The different places to put the first node are:

- none – the node is laid out around the circle
- fNode – the node is placed in the center of the circle and the remaining nodes along the outside of the circle

5.15.6.1.20 Bend Point

The simple type ST_BendPoint specifies where the bend point is to be located along elbow connectors. The different options are:

- beg – beginning
- def – default
- end – end

5.15.6.1.21 Connector Routing

The simple type ST_ConnectorRouting defines how the routing of a connector happens within the diagram. The different routing options are:

- stra – straight
- bend – an elbow connection
- curve – a curved connection
- longCurve – a curved connection with a larger radius than simple curve

5.15.6.1.22 Arrowhead Style

The simple type ST_ArrowheadStyle defines the style of the arrowhead used on a connector. The different options are:

- auto – automatic
- arr – an arrowhead is used
- noArr – no arrowhead is used

5.15.6.1.23 Connector Dimension

The simple type ST_ConnectorDimension defines the dimension of a connector used in a diagram. The different dimension types are:

- 1D – a single dimension connector, for example, a line
- 2D – a two dimensional connector, for example, an arrow
- cust – custom

5.15.6.1.24 Connector Point

The simple type ST_ConnectorPoint defines the point at which the connector starts and ends. The different beginning and ending types are:

- auto – automatic
- bCtr – bottom center
- ctr – center
- midL – middle left
- midR – middle right
- tCtr – top center
- bL – bottom left
- bR – bottom right
- tL – top left
- tR – top right
- radial – radial

5.15.6.1.25 Node Horizontal Alignment

The simple type ST_NodeHorizontalAlignment defines the alignment of a node in the horizontal direction. The different alignments are:

- l – left
- ctr – center
- r – right

5.15.6.1.26 Node Vertical Alignment

The simple type ST_NodeVerticalAlignment defines the alignment of a node in the vertical direction. The different alignments are:

- t – top
- mid – mid

- b – bottom

5.15.6.1.27 Fallback Dimension

The simple type ST_FallbackDimension defines how many dimensions fallback will resize the diagram in. The different options for fallback dimension are:

- 1D – fallback occurs in a single dimension (X or Y)
- 2D – fallback occurs in two dimensions (X and Y)

5.15.6.1.28 Text Direction

The simple type ST_TextDirection specifies where the text on the first node starts. The different text directions are:

- fromT – from top
- fromB – from bottom

5.15.6.1.29 Pyramid Accent Position

The simple type ST_PyramidAccentPosition defines where the position of the fly-out grandchildren. The possible positions are:

- bef – before
- after – after

5.15.6.1.30 Pyramid Text Margin

The simple type ST_PyramidAccentTextMargin specifies the alignment of the text in the fly-out grandchildren. The different alignments are:

- step – the text is against the edge of the pyramid
- stack – the text aligns

5.15.6.1.31 Text Block Direction

The simple type ST_TextBlockDirection defines the text block direction. The different direction the text can have are:

- vert – vertical
- horz – horizontal

5.15.6.1.32 Text Anchor Horizontal

The simple type ST_AnchorHorizontal is responsible for anchoring text horizontally. The available options are:

- none – no anchor set
- ctr – the text is anchored the center

5.15.6.1.33 Text Anchor Vertical

The simple type ST_AnchorVertical is responsible for anchoring the text vertically. The available options are:

- t – top anchor
- mid – middle anchor
- b – bottom anchor

5.15.6.1.34 Text Alignment

The simple type ST_TextAlignment defines the text alignment. The available options are:

- l – left
- ctr – ctr
- r – right

5.15.6.1.35 Auto Text Rotation

The simple type ST_AutoTextRotation defines the behavior of the text as the containing shape is rotated. The following options are available:

- none – no rotation, the text rotates with the shape
- upr – upright
- grav – gravity

5.15.6.1.36 Grow Direction

The simple type ST_GrowDirection defines the growing behavior of the snake algorithm. The following options are available:

- tL – grow from the top left
- tR – grow from the top right
- bL – grow from the bottom left
- gR – grow from the bottom right

5.15.6.1.37 Flow Direction

The simple type ST_FlowDirection Specifies whether nodes are arranged in rows or columns for the snake algorithm. The following options are available:

- row – Row
- col – column

5.15.6.1.38 Continue Direction

The simple type ST_ContinueDirection specifies the direction of the subsequent row or column in the snake algorithm. The following options are available:

- revDir – reverse direction

- sameDir – same direction

5.15.6.1.39 Breakpoint

The simple type ST_Breakpoint defines the behavior of a snake diagram's breaking behavior. The available options are:

- endCnv – end of canvas
- bal – balanced
- fixed – fixed

5.15.6.1.40 Offset

The simple type ST_Offset defines the behavior of whether each row or column in the snake algorithm is offset from the previous row or column. The available options are:

- ctr – the offset is center based
- off – there is an offset defined

5.15.6.1.41 Hierarchy Alignment

The simple type ST_HierarchyAlignment specifies the relationship between parent and children in a hierarchy diagram. The following options exist:

- tL – top left
- tR – top right
- tCtrCh – top center children
- tCtrDes – top center descendants
- bL – bottom left
- bR – bottom right
- bCtrCh – bottom center children
- bCtrDes – bottom center descendants
- lT – left top
- lB – left bottom
- lCtrCh – left center children
- lCtrDes – left center descendants
- rT – right top
- rB – right bottom
- rCtrCh – right center children
- rCtrDes – right center descendants

5.15.6.2 Variable Type

The simple type ST_VariableType defines the type of the conditional expression. These variables turn user interface options on and off. The available variable types are:

- unknown – Unknown variable type. This can be used by implementing applications to define a variable type outside of the scope of what is defined by this simple type.
- orgChart – organizational chart command
- chMax – used for the insert shape dropdown commands
- chPref – used for the insert shape button
- bulEnabled – used for the insert bullet command
- dir – diagram direction, RTL or LTR
- hierBranch – stores the different layouts for org chart
- animOne – exposes options for animation
- animLvl – exposes options for animation

5.15.6.2.1 Output Shape Type

The simple type ST_OutputShapeType defines special shape types which are unique to diagrams. The unique types are:

- none – none
- conn – connector

5.15.6.3 Diagram Definitions

Diagram definitions define the look of a diagram. They utilize almost all the aspects of the file format discussed thus far in order to create layout properties which get translated into visual diagrams.

There are a few more simple types that need to be defined before talking about the larger aspects of what it takes to create a diagram definition. Many of these simple types are provided as wrappers or lists of the above mentioned simple types.

5.15.6.3.1 Lists

There are a group of simple types which act as lists of the simple types already mentioned. They are all defined in the following general way:

```
<xsd:simpleType name= NAME OF TYPE >
  <xsd:list itemType= NAME OF SIMPLE TYPE />
</xsd:simpleType>
```

The list of these list types are the following simple types:

- ST_AxisTypes – list of ST_AxisType
- ST_ElementTypes – list of ST_ElementType
- ST_Ints – list of xsd:int
- ST_UnsignedInts – list of xsd:unsignedInt
- ST_Booleans – list of xsd:boolean

5.15.6.3.2 Function Value

The simple type ST_FunctionValue is a value for a condition expression. It is defined as:

```
<xsd:simpleType name="ST_FunctionValue" final="restriction">
  <xsd:union memberTypes="xsd:int xsd:boolean ST_Direction
    ST_HierBranchStyle ST_AnimOneStr ST_AnimLvlStr" />
</xsd:simpleType>
```

5.15.6.3.3 Direction

The simple type ST_Direction defines the direction the diagram is to be laid out. The directions available are:

- norm – normal
- rev – reversed

5.15.6.3.4 Hierarchy Branch Style

The simple type ST_HierBranchStyle changes the behavior of the branch style in hierarchy, or org chart, diagrams. This value can be modified by a user directly from the user interface. The different types of branch styles are:

- l – left
- r – right
- hang – hanging
- std – standard
- init – initial

5.15.6.3.5 One by One Animation

The simple type ST_AnimOneStr allows for differentiation in the way a one-by-one animation is displayed in the user interface. The following options are available:

- none – nothing is displayed
- one – the term one-by-one is used
- branch – the term branch one-by-one is used to distinguish a hierarchy diagram

5.15.6.3.6 Level Animation

The simple type ST_AnimLvlStr acts very much like the type ST_AminOneStr, as it allows for two different descriptions of a single animation depending upon the desired behavior for a particular diagram. The following allows for differentiation of radial diagrams. The different options are:

- none – nothing
- lvl – normal depth first traversal
- ctr – allows a radial diagram to be shown with the center node first

5.15.6.3.7 Org Chart Flag

The complex type CT_OrgChart defines that the diagram will be an organizational chart. Organizational charts contain special behavior in that assistants can now be utilized correctly. The complex type is defined as follows:

```
<xsd:complexType name="CT_OrgChart">
  <xsd:attribute name="val" type="xsd:boolean" default="false"
    use="optional" />
</xsd:complexType>
```

5.15.6.3.8 Node Count

The simple type ST_NodeCount holds a value that is used by the complex types CT_ChildMax and CT_ChildPref.

5.15.6.3.9 Child Max

This complex type defines when the user interface for inserting a child shape is to become disabled for a given node, or rather the max number of children that the user interface will be enabled for. This complex type is defined as:

```
<xsd:complexType name="CT_ChildMax">
  <xsd:attribute name="val" type="ST_NodeCount" default="-1"
    use="optional" />
</xsd:complexType>
```

5.15.6.3.10 Child Preference

This complex type defines how many children are inserted with a single action through the user interface to add a child. This is useful in hierarchy diagrams in which one would like to specify that every shape should have three children. A single click of the add shape button would add three children. The complex type is defined as follows:

```
<xsd:complexType name="CT_ChildPref">
  <xsd:attribute name="val" type="ST_NodeCount" default="-1"
    use="optional" />
</xsd:complexType>
```

5.15.6.3.11 Bullets Enabled

This complex type defines if the user interface for inserting a bullet into a shape is enabled or disabled for a given node. The complex type is defined as follows:

```
<xsd:complexType name="CT_BulletEnabled">
  <xsd:attribute name="val" type="xsd:boolean" default="false"
    use="optional" />
</xsd:complexType>
```

5.15.6.3.12 Direction

This complex type defines the direction of the diagram, be it normal or reversed. The complex type is defined as:

```
<xsd:complexType name="CT_Direction">
  <xsd:attribute name="val" type="ST_Direction" default="norm"
    use="optional" />
</xsd:complexType>
```

5.15.6.3.13 Hierarchy Branch Style

This complex type defines the hierarchy branch style for a diagram. The complex type is defined as:

```
<xsd:complexType name="CT_HierBranchStyle">
  <xsd:attribute name="val" type="ST_HierBranchStyle" default="std"
    use="optional" />
</xsd:complexType>
```

5.15.6.3.14 Animate as One

This complex type defines the animate as one value for a diagram. The complex type is defined as:

```
<xsd:complexType name="CT_AnimOne" >
  <xsd:attribute name="val" type="ST_AnimOneStr" default="one"
    use="optional" />
</xsd:complexType>
```

5.15.6.3.15 Animate by Level

This complex type defines the animate by level value for a diagram. The complex type is defined as:

```
<xsd:complexType name="CT_AnimLvl">
  <xsd:attribute name="val" type="ST_AnimLvlStr" default="none"
    use="optional" />
</xsd:complexType>
```

5.15.6.3.16 Layout Property Set

The complex type CT_LayoutPropertySet holds all of the layout properties for a given diagram. The layout property set is a single structure which contains most of what has been talked about thus far in a diagram definition. The layout property set is defined as:

```
<xsd:complexType name="CT_LayoutVariablePropertySet">
  <xsd:sequence>
    <xsd:element name="orgChart" type="CT_OrgChart"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="chMax" type="CT_ChildMax" minOccurs="0"
      maxOccurs="1" />
```

```
      <xsd:element name="chPref" type="CT_ChildPref"
        minOccurs="0" maxOccurs="1" />
      <xsd:element name="bulletEnabled" type="CT_BulletEnabled"
        minOccurs="0" maxOccurs="1" />
      <xsd:element name="dir" type="CT_Direction" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="hierBranch" type="CT_HierBranchStyle"
        minOccurs="0" maxOccurs="1" />
      <xsd:element name="animOne" type="CT_AnimOne" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="animLvltype="CT_AnimLvl" minOccurs="0"
        maxOccurs="1" />
    </xsd:sequence>
  </xsd:complexType>
```

Because all of the contents of this complex type have already been discussed, no further detail on this complex type needs to be given.

5.15.6.3.17 Iterators

The attribute group AG_IteratorAttributes defines the attributes used by the iterators forEach, presOf, and if. The attribute group is defined as follows:

```
  <xsd:attributeGroup name="AG_IteratorAttributes">
    <xsd:attribute name="axis" type="ST_AxisTypes" use="optional"
      default="none" />
    <xsd:attribute name="ptType" type="ST_ElementTypes"
      use="optional" default="all" />
    <xsd:attribute name="hideLastTrans" type="ST_Booleans"
      use="optional" default="true" />
    <xsd:attribute name="st" type="ST_Ints" use="optional" default="1" />
    <xsd:attribute name="cnt" type="ST_UnsignedInts" use="optional"
      default="0" />
    <xsd:attribute name="step" type="ST_Ints" use="optional" default="1" />
  </xsd:attributeGroup>
```

5.15.6.3.18 Constraints

The attribute group AG_ConstraintAttributes defines the attributes used to specify a constraint. The attribute group is defined as:

```
  <xsd:attributeGroup name="AG_ConstraintAttributes">
    <xsd:attribute name="type" type="ST_ConstraintType" use="required" />
    <xsd:attribute name="for" type="ST_ConstraintRelationship"
      use="optional" default="self" />
    <xsd:attribute name="forName" type="xsd:IDREF" use="optional" />
```

```
    <xsd:attribute name="ptType" type="ST_ElementType" use="optional"
      default="all" />
  </xsd:attributeGroup>
```

5.15.6.3.19 Constraint References

The attribute group AG_ConstraintRefAttributes defines the attributes used to specify a constraint reference. The attribute group is defined as:

```
  <xsd:attributeGroup name="AG_ConstraintRefAttributes">
    <xsd:attribute name="refType" type="ST_ConstraintType"
      use="optional" default="unknown" />
    <xsd:attribute name="refFor" type="ST_ConstraintRelationship"
      use="optional" default="self" />
    <xsd:attribute name="refForName" type="xsd:IDREF"
      use="optional" />
    <xsd:attribute name="refPtType" type="ST_ElementType"
      use="optional" default="all" />
  </xsd:attributeGroup>
```

5.15.6.3.20 Constraint

The complex type CT_Constraint define a constraint within the layout framework. A constraint acts as a limit or sets a value to a given parameter in a diagram definition, for example, it can be used to specify that all nodes of a give point type are the same size. A constraint is defined as:

```
  <xsd:complexType name="CT_Constraint">
    <xsd:attributeGroup ref="AG_ConstraintAttributes" />
    <xsd:attributeGroup ref="AG_ConstraintRefAttributes" />
    <xsd:attribute name="op" type="ST_BoolOperator" use="optional"
      default="none" />
    <xsd:attribute name="val" type="xsd:double" use="optional"
      default="0" />
    <xsd:attribute name="fact" type="xsd:double" use="optional"
      default="1" />
  </xsd:complexType>
```

5.15.6.3.21 Constraint List

The complex type CT_Constraints is a sequence of CT_Constraint complex types. It is defined as:

```
  <xsd:complexType name="CT_Constraints">
    <xsd:sequence>
      <xsd:element name="constr" type="CT_Constraint" minOccurs="0"
        maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
```

5.15.6.3.22 Rule

The complex type CT_NumericRule defines a layout framework constraint rule. Rules are run after the diagram is created in order to specify what happens when the diagram doesn't fully fit within the bounds. This allows for specific behavior to be defined rather than using default rules for fitting the diagram. A rule is defined in the following way:

```
<xsd:complexType name="CT_NumericRule" >
  <xsd:attributeGroup ref="AG_ConstraintAttributes" />
  <xsd:attribute name="val" type="xsd:double" use="optional"
    default="NaN" />
  <xsd:attribute name="fact" type="xsd:double" use="optional"
    default="NaN" />
  <xsd:attribute name="max" type="xsd:double" use="optional"
    default="NaN" />
</xsd:complexType>
```

5.15.6.3.23 Rule List

The complex type CT_Rules is simply a list of CT_NumericRule complex types. It is defined in the following manner:

```
<xsd:complexType name="CT_Rules">
  <xsd:sequence>
    <xsd:element name="rule" type="CT_NumericRule"
      minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>
```

5.15.6.3.24 Presentation Of

The complex type CT_PresentationOf defines the mapping between data and the diagram. The complex type is defined in the following manner:

```
<xsd:complexType name="CT_PresentationOf">
  <xsd:attributeGroup ref="AG_IteratorAttributes" />
</xsd:complexType>
```

5.15.6.3.25 Layout Shape

The simple type ST_LayoutShapeType is a simple type that contains all of the shapes available which can be used within a diagram. The simple type is defined as a union of ST_OutputShapeType and an externally defined ST_ShapeType.

5.15.6.3.26 Index1

The simple type ST_Index1 defines a 1-based index that is used to index values elsewhere. The simple type is defined as:

```
<xsd:simpleType name="ST_Index1">
  <xsd:restriction base="xsd:unsignedInt">
    <xsd:minInclusive value="1" />
  </xsd:restriction>
</xsd:simpleType>
```

5.15.6.3.27 Adjust Handle

The complex type CT_Adj specifies a shape adjust handle modification. The shapes within a diagram can be modified based on their adjust handles, for example, the radius of the corner rounding in a rounded rectangle can be adjusted using this complex type. The complex type is defined in the following manner:

```
<xsd:complexType name="CT_Adj">
  <xsd:attribute name="idx" type="ST_Index1" use="required" />
  <xsd:attribute name="val" type="xsd:double" use="required" />
</xsd:complexType>
```

5.15.6.3.28 Adjust Handle List

The complex type CT_AdjLst holds all of the adjust handles for a given shape. The number of adjust handles accessible varies shape by shape, but there are usually less than four for a given shape. The complex type is defined in the following way:

```
<xsd:complexType name="CT_AdjLst" o:cname="CAdjList">
  <xsd:sequence>
    <xsd:element name="adj" type="CT_Adj" minOccurs="0"
      maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>
```

5.15.6.3.29 Shape

The complex type CT_Shape specifies a shape for a layout node. The shape complex type holds all of the information associated with the particular layout node and all of the adjustments or modifications that can be made to the shape. The rot attribute specifies a rotation on the shape. The blip attribute specifies an image that is used as a background fill for the shape and the blipPhldr attribute specifies whether or not the shape shows up with an image placeholder. The zOrderOff attribute specifies an offset to be used for the z-ordering of this shape, while the lkTxEntry attribute prevents text editing within the shape. A shape is defined in the following manner:

```
<xsd:complexType name="CT_Shape">
  <xsd:sequence>
    <xsd:element name="adjLst" type="CT_AdjLst" minOccurs="0"
      maxOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="rot" type="xsd:double" use="optional" default="0" />
```

```
    <xsd:attribute name="type" type="ST_LayoutShapeType"
      use="optional" default="none" />
    <xsd:attribute ref="r:blip" use="optional" />
    <xsd:attribute name="zOrderOff" type="xsd:int" use="optional"
      default="0" />
    <xsd:attribute name="hideGeom" type="xsd:boolean" use="optional"
      default="false" />
    <xsd:attribute name="lkTxEntry" type="xsd:boolean" use="optional"
      default="false" />
    <xsd:attribute name="blipPhldr" type="xsd:boolean" use="optional"
      default="false" />
  </xsd:complexType>
```

5.15.6.3.30 Parameter

The complex type CT_Parameter holds the information regarding an algorithm parameter. The complex type is defined as:

```
  <xsd:complexType name="CT_Parameter">
    <xsd:attribute name="type" type="ST_ParameterId" use="required" />
    <xsd:attribute name="val" type="xsd:string" use="required" />
  </xsd:complexType>
```

5.15.6.3.31 Algorithm

The complex type CT_Algorithm defines the algorithm which the diagram will use to layout the nodes which contain the data. Also defined here are the optional list of parameters which are associated with this algorithm and modify its behavior. An algorithm is defined in the following manner:

```
  <xsd:complexType name="CT_Algorithm" >
    <xsd:sequence>
      <xsd:element name="param" type="CT_Parameter" minOccurs="0"
        maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="type" type="ST_AlgorithmType" use="required" />
    <xsd:attribute name="rev" type="xsd:unsignedInt" use="optional"
      default="0" />
  </xsd:complexType>
```

5.15.6.3.32 Layout Node

The complex type CT_LayoutNode is the main building block of a diagram. A layout node contains enough information to lay out itself and its children. The name attribute is simply a unique string given to the layout node. The styleLbl attribute references the style label that is used to style the layout node. This style label has already been defined in this document. A layout node is defined in the following manner:

```
<xsd:complexType name="CT_LayoutNode">
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="alg" type="CT_Algorithm" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="shape" type="CT_Shape" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="presOf" type="CT_PresentationOf"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="constrLst" type="CT_Constraints"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="ruleLst" type="CT_Rules" minOccurs="0"
    maxOccurs="1" />
    <xsd:element name="varLst"
      type="CT_LayoutVariablePropertySet" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="forEach" type="CT_ForEach" />
    <xsd:element name="layoutNode" type="CT_LayoutNode" />
    <xsd:element name="choose" type="CT_Choose" />
  </xsd:choice>
  <xsd:attribute name="name" type="xsd:ID" use="optional" />
  <xsd:attribute name="styleLbl" type="xsd:string" use="optional" />
  <xsd:attribute name="chOrder" type="ST_ChildOrderType"
    use="optional" default="b" />
  <xsd:attribute name="moveWith" type="xsd:IDREF" use="optional" />
</xsd:complexType>
```

5.15.6.3.33 For Each

The complex type CT_ForEach defines a for each iterator. The iteration behaves as if it were a for each loop. The complex type is defined as:

```
<xsd:complexType name="CT_ForEach">
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="alg" type="CT_Algorithm" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="shape" type="CT_Shape" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="presOf" type="CT_PresentationOf"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="constrLst" type="CT_Constraints"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="ruleLst" type="CT_Rules" minOccurs="0"
      maxOccurs="1" />
```

```
      <xsd:element name="forEach" type="CT_ForEach" />
      <xsd:element name="layoutNode" type="CT_LayoutNode" />
      <xsd:element name="choose" type="CT_Choose" />
    </xsd:choice>
    <xsd:attribute name="name" type="xsd:ID" use="optional" />
    <xsd:attribute name="ref" type="xsd:IDREF" use="optional" />
    <xsd:attributeGroup ref="AG_IteratorAttributes" />
  </xsd:complexType>
```

5.15.6.3.34 When

The complex type CT_When defines an if conditional expression. The complex type is usually used in conjunction with the else counterpart which is defined next. The CT_When complex type is defined in the following manner:

```
  <xsd:complexType name="CT_When">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="alg" type="CT_Algorithm" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="shape" type="CT_Shape" minOccurs="0"
        maxOccurs="1" />
      <xsd:element name="presOf" type="CT_PresentationOf"
        minOccurs="0" maxOccurs="1" />
      <xsd:element name="constrLst" type="CT_Constraints"
        minOccurs="0" maxOccurs="1" />
      <xsd:element name="ruleLst" type="CT_Rules" minOccurs="0"
        maxOccurs="1" o:cname="Rules" />
      <xsd:element name="forEach" type="CT_ForEach" />
      <xsd:element name="layoutNode" type="CT_LayoutNode" />
      <xsd:element name="choose" type="CT_Choose" />
    </xsd:choice>
    <xsd:attribute name="name" type="xsd:ID" use="optional" />
    <xsd:attributeGroup ref="AG_IteratorAttributes" />
    <xsd:attribute name="func" type="ST_FunctionType"
      use="required" />
    <xsd:attribute name="arg" type="ST_FunctionArgument"
      use="optional" />
    <xsd:attribute name="op" type="ST_FunctionOperator"
      use="required" />
    <xsd:attribute name="val" type="ST_FunctionValue"
        use="required" />
  </xsd:complexType>
```

5.15.6.3.35 Otherwise

The complex type CT_Otherwise is the else counterpart to the already defined if conditional expression. The complex type is defined as:

```
<xsd:complexType name="CT_Otherwise" o:cname="DDOtherwise">
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="alg" type="CT_Algorithm" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="shape" type="CT_Shape" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="presOf" type="CT_PresentationOf"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="constrLst" type="CT_Constraints"
      minOccurs="0" maxOccurs="1" />
    <xsd:element name="ruleLst" type="CT_Rules" minOccurs="0"
      maxOccurs="1" />
    <xsd:element name="forEach" type="CT_ForEach" />
    <xsd:element name="layoutNode" type="CT_LayoutNode" />
    <xsd:element name="choose" type="CT_Choose" />
  </xsd:choice>
  <xsd:attribute name="name" type="xsd:ID" use="optional" />
</xsd:complexType>
```

5.15.6.3.36 Choose Statement

The complex type CT_Choose packages together the if and else conditions into an actual if/else statement. The complex type is defined in the following manner:

```
<xsd:complexType name="CT_Choose" o:cname="DDChoose">
  <xsd:sequence>
    <xsd:element name="if" type="CT_When" maxOccurs="unbounded" />
    <xsd:element name="else" type="CT_Otherwise" minOccurs="0" />
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:ID" use="optional" />
</xsd:complexType>
```

5.15.6.3.37 Sample Data

The complex type CT_SampleData defines how the data model is to be populated in an initial manner. The complex type holds a temporary data model when there is no data model present in order to display a diagram on an initial insert. The complex type is defined by:

```
<xsd:complexType name="CT_SampleData">
  <xsd:sequence>
    <xsd:element name="dataModel" type="CT_DataModel" minOccurs="0" />
  </xsd:sequence>
```

```
    <xsd:attribute name="useDef" type="xsd:boolean" use="optional"
      default="false" />
  </xsd:complexType>
```

5.15.6.3.38 Common Structures

CT_Category, CT_Categories, CT_Name, CT_Description, and ST_Version are defined just as their counterparts in the subclauses above, and they perform the same tasks.

5.15.6.3.39 Diagram Definition

The complex type CT_DiagramDefinition is the root element for a diagram definition. It is defined in the following manner:

```
  <xsd:complexType name="CT_DiagramDefinition">
    <xsd:sequence>
      <xsd:element name="title" type="CT_Name" minOccurs="0"
        maxOccurs="unbounded" />
      <xsd:element name="desc" type="CT_Description"
        minOccurs="0" maxOccurs="unbounded" />
      <xsd:element name="catLst" type="CT_Categories" minOccurs="0 />
      <xsd:element name="sampData" type="CT_SampleData" minOccurs="0" />
      <xsd:element name="styleData" type="CT_SampleData" minOccurs="0" />
      <xsd:element name="clrData" type="CT_SampleData" minOccurs="0" />
      <xsd:element name="layoutNode" type="CT_LayoutNode" />
    </xsd:sequence>
    <xsd:attribute name="uniqueId" type="xsd:anyURI" use="optional" />
    <xsd:attribute name="minVer" type="ST_Version" use="optional"
      default="12.0" />
    <xsd:attribute name="defStyle" type="xsd:anyURI" use="optional" />
  </xsd:complexType>
```

6. Introduction to VML

This clause is informative.

This clause contains a detailed introduction to the components of Vector Markup Language (VML).

6.1 Introduction

This section provides an overview of the most common parts of VML. The VML format is a legacy format originally introduced with Office 2000 and is included and fully defined in this specification for backwards compatibility reasons. The DrawingML format is a newer and richer format created with the goal of eventually replacing any uses of VML in the Office Open XML formats. VML should be considered a deprecated format included in Office Open XML for legacy reasons only and new applications that need a file format for drawings are strongly encouraged to use preferentially DrawingML .

VML is an XML-based exchange, editing, and delivery format for high-quality vector graphics. VML facilitates the exchange and subsequent editing of vector graphics between a wide variety of productivity and design applications. VML is based on XML 1.0, which is an open, simple, text-based language for describing structured data. VML also supports other World Wide Web Consortium standards, such as Cascading Style Sheets 2.0 (CSS), which specifies style information and 2-D positioning.

As the VML format is a format provided for backward compatibility, many VML elements are defined in the same `urn:schemas-microsoft-com:vml` namespace that is currently used by millions of documents already using VML. In the documentation this is typically shortened to a v: prefix in the VML tag by defining `xmlns:v="urn:schemas-microsoft-com:vml".` The namespaces used for VML are legacy namespaces. Once again, VML should be considered a deprecated format included in Office Open XML for legacy reasons only and new applications that need a file format for drawings are strongly encouraged to use preferentially DrawingML .

Additional elements and attributes are defined in namespaces that reflect how they are used (all VML namespaces defined in this Standard maintain the legacy namespace structure for backward compatibility):

- urn:schemas-microsoft-com:office:office (office document)
- urn:schemas-microsoft-com:office:word (word-processing document)
- urn:schemas-microsoft-com:office:excel (spreadsheet document)
- urn:schemas-microsoft-com:office:powerpoint (presentation document)

6.2 Shape Element

The Shape element is the basic building block of VML. A shape may exist on its own or within a Group element. Shape defines many attributes and sub-elements that control the look and behavior of the shape. A

shape must define at least a Path and size (Width, Height). VML also uses properties of the CSS2 style attribute to specify positioning and sizing.

Note that this subclause also applies to the set of pre-defined shape primitives provided by the VML elements Arc, Curve, Image, Line, Oval, Polyline, Rect, and RoundRect.

The following attributes are used to define a minimal shape:

Attribute	Description
FillColor	Brush color that fills the closed path of a shape.
Position	Type of positioning used to place an element.
Top	Position of the shape relative to the element above it in the flow of the page.
Left	Position of the shape relative to the element left of it in the document flow.
Width	Width of the shape.
Height	Height of the shape.
Path	Line that makes up the edges of a shape.

The following example creates a minimal shape:

```
<v:shape fillcolor="green"
  style="position:relative;top:1;left:1;width:50;
  height:50" path="m 1,1 l 1,50, 50,50, 50,1 x e">
</v:shape>
```

Although there is no official categorization of the Shape element's attributes or sub-elements, it is useful to think of them in groups. The following sections broadly describe the characteristics of the Shape element. A few fundamental attributes and elements are introduced here. For complete details, see the VML reference in Part 4.

6.2.1 Geometry

The following attributes affect the basic structure or outline of the shape.

Attribute	Description
Adj	Adjustment value used to define values for a formula.
Height*	Height of the shape.
Path	Line that makes up the edges of a shape.
Width*	Width of the shape.

* indicates a CSS2 style property

Element	Description

Element	Description
Callout	Defines a callout for a shape.
Extrusion	Defines an extrusion for a shape.
Path	Defines a path for a shape.
Skew	Defines a skew for a shape.
Stroke	Defines a stroke for a shape.
TextBox	Defines a textbox for a shape.
TextPath	Defines a text path for a shape.

6.2.1.1 Height and Width Attributes

Height and Width may be specified using any of the following units. If no unit is specified, pixels is assumed.

Relative	
em	Height of the element's font
ex	Height of the letter "x"
px	Pixels
%	Percentage

Absolute	
in	Inches
cm	Centimeters
mm	Millimeters
pt	Points
pc	Picas

For example:

```
style="position:relative;top:1;left:1;width:50;height:50"
style="position:relative;top:1;left:1;width:10%;height:10%"
```

6.2.1.2 Path Attribute

The Path attribute contains specially formatted text that describes a set of points and line connections between them that define the shape's outline. The path defined must be closed. A path is begun by specifying `m` and a coordinate. This indicates a moveto the given coordinate. Line segments are drawn using `l` (lineto) and specifying subsequent coordinates. A line is closed with `x` after the closing coordinate. The path is ended with `e`.

For example:

```
path="m 1,1 l 1,50, 50,50, 50,1 x e"
```

This starts at (1,1), draws a line to (1,50), (50,50) and (50,1), where the line is closed and the path ended.

The coordinates specified correspond to relative coordinate space (the size of units in relative space can be set by the CoordSize attribute). The shape's actual size is determined by the Height and Width attributes.

For example:

```
<v:shape style="position:relative;top:1;left:1;width:5000;
  height:5000" fillcolor="teal"
  path="m 1,1 l 1,10 10,10 10,1 1,1 x e" />
```

```
<v:shape  style="position:relative;top:1;left:1;width:2500;
  height:2500" fillcolor="teal"
  path="m 1,1 l 1,10 10,10 10,1 1,1 x e" />
```

More than one closed line path may be specified in the Path attribute and each closed region is filled.

```
<v:shape  style="position:relative;top:1;left:1;width:5000;
  height:5000" fillcolor="teal"
  path="m 1,1 l 1,10 10,10 10,1 1,1 x m 20,20 l 20,40 40,40
  40,20 20,20 x e" />
```

The optional Path element, which allows for the creation of more complex paths and regions, overrides the Path attribute if it is specified.

6.2.2 Placement

These attributes affect the layout and placement of shapes. Placement may be defined relative to other shapes or non-VML content that also exists in the container holding the shape.

Attribute	Description
AllowOverlap	Determines if a shape can overlap other shapes.
CoordOrigin	Specifies the coordinate unit origin of the rectangle that bounds a shape.
CoordSize	Specifies the horizontal and vertical units of the rectangle that bounds a shape.
Flip*	Switches the orientation of a shape.
Left*	Determines the position of the shape relative to the element left of it in the document flow.
Margin-Bottom*	Specifies the bottom edge of the shape's containing rectangle relative to the shape anchor.
Margin-Left*	Specifies the left edge of the shape's containing rectangle relative to the shape anchor.

Attribute	Description
Margin-Right*	Specifies the right edge of the shape's containing rectangle relative to the shape anchor.
Margin-Top*	Specifies the top edge of the shape's containing rectangle relative to the shape anchor.
MSO-Position-Horizontal*	Specifies the horizontal positioning data for objects in WordprocessingML.
MSO-Position-Horizontal-Relative*	Specifies relative horizontal position data for objects in WordprocessingML.
MSO-Position-Vertical*	Specifies the vertical position data for objects in WordprocessingML.
MSO-Position-Vertical-Relative*	Specifies relative vertical position data for objects in WordprocessingML.
MSO-Wrap-Distance-Bottom*	Defines the distance from the bottom side of the shape to the text that wraps around it.
MSO-Wrap-Distance-Left*	Defines the distance from the left side of the shape to the text that wraps around it.
MSO-Wrap-Distance-Right*	Defines the distance from the right side of the shape to the text that wraps around it.
MSO-Wrap-Distance-Top*	Defines the distance from the shape top to the text that wraps around it.
MSO-Wrap-Edited*	Determines whether the wrap coordinates were customized by the user.
MSO-Wrap-Mode*	Defines the wrapping mode for text.
Position*	Defines the type of positioning used to place an element.
RelativePosition	Defines a relative position for an object.
Rotation*	Defines the angle that a shape is rotated.
Top*	Defines the position of the shape relative to the element above it in the flow of the page.
Z-Index*	Determines the display order of overlapping shapes.

* indicates a CSS2 style property

6.2.2.1 CoordOrigin and CoordSize Attributes

These attributes define the relative coordinate space of a shape. This space is scaled up or down to match the specified Width and Height of the shape. Coordinates in the Path attribute (or element) are relative to the space defined by CoordOrigin and CoordSize, so the Path definition never needs to change simply to scale the shape.

CoordSize defines the “width” and “height” of the local coordinate space. CoordOrigin defines the top-left coordinate of this space.

For example:

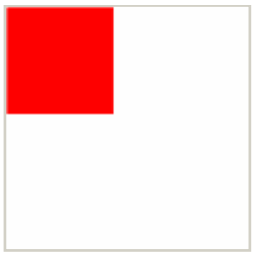
```
coordorigin="0,0"
coordsize="200,200"
```
Extents of local space are (0,0) to (200,200)

```
coordorigin="-100,-100"
coordsize="200,200"
```
Extents of local space are (-100,-100) to (100,100)

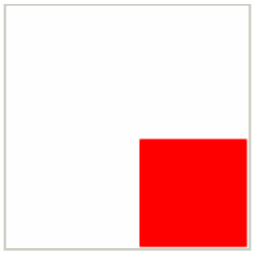
This local space definition affects the position of the shape. Changing the CoordOrigin translates the shape within the local space. Changing the CoordSize affects the size of the shape by changing the size of the local space relative to the shape’s Width and Height.

For example:

```
coordorigin="0,0"
coordsize="500,500"
style="position:absolute;top:0;left:0;width:100pt;
  height:100pt"
```



```
coordorigin="-250,-250"
coordsize="500,500"
style="position:absolute;top:0;left:0;width:100pt;
  height:100pt"
```



```
coordorigin="0,0"
coordsize="250,250"
style="position:absolute;top:0;left:0;width:100pt;
height:100pt"
```

6.2.2.2 Position Attribute

Position can be specified as “static”, “relative” or “absolute”. Static positioning keeps the shape inline with the current flow of the surrounding content – the Top and Left attributes are ignored. Relative uses the Top and

Left attributes to position the shape relative to its position in the current flow. Absolute uses the Top and Left attributes to position the shape with respect to its container.

6.2.3 Formatting

These attributes and elements affect the fill and line properties of the shape.

Attribute	Description
BorderBottomColor	Bottom border color of an inline shape.
BorderLeftColor	Left border color of an inline shape.
BorderRightColor	Right border color of an inline shape.
BorderTopColor	Top border color of an inline shape.
BWMode	Determines how a shape will render for black-and-white output devices.
BWNormal	Defines the black-and-white mode for normal black-and-white output devices.
BWPure	Defines the black-and-white mode for pure black-and-white output devices.
Chromakey	Defines a color that will be transparent and show anything behind the shape.
FillColor	Defines the brush color that fills the closed path of a shape.
Filled	Determines whether the closed path will be filled.
ForceDash	Determines whether a dashed outline is used to draw a shape when a shape has no line or fill.
HR	Specifies that a shape is a horizontal rule.
HRAlign	Defines the alignment of a horizontal rule.
HRHeight	Defines the thickness of a horizontal rule.
HRNoShade	Determines whether a horizontal rule will be displayed with 3-D shading.
HRPct	Defines the length of a horizontal rule as a percentage of page width.
HRStd	Determines whether a shape is a standard horizontal rule.
HRWidth	Defines the length of a horizontal rule.
StrokeColor	Defines the brush color that strokes the path of a shape.
Stroked	Defines whether the path will be stroked.
StrokeWeight	Defines the brush thickness that strokes the path of a shape.

Element	Description
Fill	Defines a fill for a shape.
Imagedata	Defines image data for a shape.
Shadow	Defines a shadow for a shape.

6.2.4 Other

These are miscellaneous attributes and elements.

Attribute	Description
Alt	Defines alternative text to be displayed instead of a graphic.
AllowInCell	Determines whether a shape can be placed in a table.
Bullet	Determines whether a shape is a graphical bullet.
Button	Determines whether a shape will be processed as a button.
Class	Refers to a definition of a CSS style.
ConnectorType	Indicates the type of connector used for joining shapes.
DoubleClickNotify	Sends an event message when a shape is double-clicked.
HRef	Defines a URL for a shape. When the shape is clicked, the browser will load the URL.
ID	Provides a unique identifier for an element.
InsetMode	Specifies whether the host calculates the internal text margin instead of using the inset attribute of the textbox element.
OLE	Specifies whether the shape is an embedded OLE object.
OLEIcon	Determines whether an OLE object will be displayed as an icon.
OnEd	Determines whether the extra handles of a shape are hidden.
OnMouseOver	Triggers a mouse event for a shape.
PreferRelative	Determines whether the original size of an object is saved after reformatting.
Print	Determines whether the shape will be printed.
ReGroupID	Defines a previous group for a shape.
RuleInitiator	Determines whether a rules engine will be used.
RuleProxy	Determines whether a proxy for the rules engine will be used.
Spt	Defines a number used to identify types of shapes.
TableLimits	List of minimum height values for each row in a table.
TableProperties	Determines table properties.
Target	Defines a frame or window that a URL will be displayed in.
Title	Defines the text displayed when the mouse pointer moves over the shape.
Type	Defines a reference to the ID of a ShapeType element.
UserDrawn	Determines whether the user has added the shape to a master slide.
UserHidden	Determines whether a script anchor is hidden.
Visibility	Determines whether a shape is displayed.
WrapCoords	Defines the bounding polygon that surrounds a shape.

1

Element	Description
Formulas	Defines formulas for a shape.

Element	Description
Handles	Defines handles for a shape.
Locks	Defines a lock for a shape.

6.3 Group Element

The Group element is used to collect multiple objects so they can be positioned and transformed as a single unit. Objects that reference their parent container's coordinate space become relative to the group's local space when inserted into a group. Using groups supports creation of complex shapes, composed of many sub-shapes, that can be treated as a single entity.

Group supports a subset of the Shape element's attributes.

Attribute			
AllowInCell	Class	HRPct	Style
AllowOverlap	CoordOrig	HRStd	TableLimits
Alt	CoordSize	HRWidth	TableProperties
BorderBottomColor	DoubleClickNotify	ID	Target
BorderLeftColor	HR	OnEd	Title
BorderRightColor	HRAlign	OnMouseOver	UserDrawn
BorderTopColor	HRef	Print	UserHidden
Bullet	HRHeight	ReGroupID	WrapCoords
Button	HRNoShade	RelativePosition	

The following elements are valid inside a Group:

Element			
Arc	Image	Polyline	Shape
Curve	Line	Rect	ShapeType
Group	Oval	RoundRect	

6.4 ShapeType Element

The ShapeType element defines a definition, or template, for a shape. Such a template is "instantiated" by creating a Shape element that references the ShapeType. The shape can override any value specified by its ShapeType, or define attributes and elements the ShapeType does not provide. A ShapeType may not reference another ShapeType.

The attributes and elements a ShapeType uses are identical to those of the Shape element, with these exceptions.

ShapeType may not use the Type element.

CSS positioning attributes are ignored and not passed to individual Shape instances.

Visibility is always hidden.

A VML authoring agent may make the ShapeType visible, in which case the CSS positioning attributes are meaningful.

The ShapeType element is used to define a shape once and reference it multiple times throughout a document. One of the most useful attributes or elements a ShapeType defines is a complex Path. Since Path coordinates are defined in a relative coordinate space that scales with a shape's height and width, this is very flexible for defining a shape outline that can be custom scaled and formatted for a given use.

6.5 VML Usage in the Office Open XML Format

6.5.1 OfficeArt Shapes

WordprocessingML takes advantage of the template-based shape definition VML provides. This example shows how the two shapes in the screenshot below are created.

The star is first defined using a ShapeType.

```
<v:shapetype id="_x0000_t12" coordsize="21600,21600" o:spt="12"
  path="m10800,l8280,8259,,8259r6720,5146l4200,21600r6600,
  -5019l17400,21600,14880,13405,21600,8259r-8280,xe">
  <v:stroke joinstyle="miter" />
  <v:path gradientshapeok="t" o:connecttype="custom"
…
  o:connectlocs="10800,0;0,8259;4200,21600;17400,21600;21600,8259"
    textboxrect="6720,8259,14880,15628" />
</v:shapetype>
```

The first star is created by referencing the ShapeType via the Type attribute. It sets its own positioning and scaling.

```
<v:shape id="_x0000_s1026" type="#_x0000_t12"
  style="position:absolute;margin-left:33pt;margin-top:25.5pt;
  width:47.25pt;height:47.25pt;z-index:251656704" />
```

The second star is created by referencing the ShapeType and providing its own positioning, scaling and formatting.

```
<v:shape id="_x0000_s1027" type="#_x0000_t12"
  style="position:absolute;margin-left:145.5pt;margin-top:25.5pt;
  width:47.25pt;height:47.25pt;z-index:251657728"
  fillcolor="#4f81bd [3204]" strokecolor="#f2f2f2 [3041]"
  strokeweight="3pt">
  <v:shadow on="t" type="perspective" color="#27405e [1604]"
    opacity=".5" offset="1pt" offset2="-1pt" />
</v:shape>
```

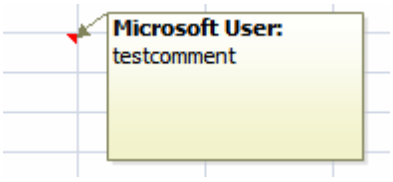
The example contains only the two star shapes. What follows is the entire document:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<w:document "…">
<w:body>
  <w:p>
    <w:r w:rsidR="00496863">
      <w:rPr>
        <w:noProof />
      </w:rPr>
      <w:pict>
      <v:shapetype id="_x0000_t12" coordsize="21600,21600"
        o:spt="12"
        path="m10800,l8280,8259,,8259r6720,5146l4200,21600r6600,
        -5019l17400,21600,14880,13405,21600,8259r-8280,xe">
        <v:stroke joinstyle="miter" />
        <v:path gradientshapeok="t" o:connecttype="custom"
          o:connectlocs="10800,0;0,8259;4200,21600;
          17400,21600;21600,8259"
          textboxrect="6720,8259,14880,15628" />
      </v:shapetype>
      <v:shape id="_x0000_s1026" type="#_x0000_t12"
        style="position:absolute;margin-left:33pt;
        margin-top:25.5pt;
        width:47.25pt;height:47.25pt;z-index:251656704" />
      </w:pict>
    </w:r>
    <w:r w:rsidR="00496863">
      <w:rPr>
        <w:noProof />
      </w:rPr>
      <w:pict>
```

```
        <v:shape id="_x0000_s1027" type="#_x0000_t12"
          style="position:absolute;margin-left:145.5pt;
          margin-top:25.5pt;width:47.25pt;height:47.25pt;
          z-index:251657728" fillcolor="#4f81bd [3204]"
          strokecolor="#f2f2f2 [3041]" strokeweight="3pt">
          <v:shadow on="t" type="perspective"
            color="#27405e [1604]"
            opacity=".5" offset="1pt" offset2="-1pt" />
        </v:shape>
      </w:pict>
    </w:r>
  </w:p>
  <w:sectPr w:rsidR="00694E3A" w:rsidRPr="00694E3A" w:rsidSect="00694E3A">
    <w:pgSz w:w="12240" w:h="15840" />
    <w:pgMar w:top="1440" w:right="1440" w:bottom="1440"
      w:left="1440"
      w:header="720" w:footer="720" w:gutter="0" />
    <w:cols w:space="720" />
    <w:docGrid w:linePitch="360" />
  </w:sectPr>
</w:body>
</w:document>
```

6.5.2 SpreadsheetML Comments

The visible box shown for comments attached to cells is persisted using VML. The comment contents are stored separately as part of SpreadsheetML.



The package item xl/worksheets/sheet1.xml contains the following reference:

```
<legacyDrawing r:id="rId1" />
```

This is a relationship defined in xl/worksheets/_rels/sheet1.xml.rels:

```
<Relationship Id="rId1" Type="…/legacyDrawing"
  Target="../drawings/legacyDrawing1.vml" />
```

The package item xl/drawings/legacyDrawing1.vml defines the yellow gradient rectangle. Again, note that the basic rectangle is defined using a ShapeType. This is reused if multiple comments exist.

```
<xml "…">
  <o:shapelayout v:ext="edit">
    <o:idmap v:ext="edit" data="1" />
  </o:shapelayout>
  <v:shapetype id="_x0000_t202" coordsize="21600,21600" o:spt="202"
    path="m,l,21600r21600,l21600,xe">
    <v:stroke joinstyle="miter" />
    <v:path gradientshapeok="t" o:connecttype="rect" />
  </v:shapetype>
  <v:shape id="_x0000_s1027" type="#_x0000_t202"
    style="position:absolute;
    margin-left:107.25pt;margin-top:52.5pt;width:96pt;height:55.5pt;
    z-index:1" fillcolor="#f2f3cb" strokecolor="#81835a"
    o:insetmode="auto">
    <v:fill color2="#fefefb" type="gradient">
      <o:fill v:ext="view" type="gradientUnscaled" />
    </v:fill>
    <v:shadow on="t" color="silver" opacity=".5" obscured="t" />
    <v:path o:connecttype="none" />
    <v:textbox style="mso-direction-alt:auto">
      <div style="text-align:left" />
    </v:textbox>
    <x:ClientData ObjectType="Note">
      <x:MoveWithCells />
      <x:SizeWithCells />
        <x:Anchor>2, 15, 3, 10, 4, 15, 7, 4</x:Anchor>
      <x:AutoFill>False</x:AutoFill>
      <x:Row>4</x:Row>
      <x:Column>1</x:Column>
      <x:Visible />
    </x:ClientData>
  </v:shape>
</xml>
```

6.5.3 WordprocessingML Text Box

WordprocessingML stores all textbox geometry using VML. This example shows how a simple text box is stored.

All the VML is embedded directly in the word/document.xml file as it is intermingled with other XML. VML is used to define the graphic content. Within the VML textbox tag, additional information about the text box text is added. The following is the section of the document.xml that defines the text box.

```
<w:r w:rsidR="00735D93">
  <w:rPr>
    <w:noProof />
  </w:rPr>
  <w:pict>
    <v:roundrect id="_x0000_s1027" style="position:absolute;
      margin-left:193.2pt;margin-top:-18pt;width:385.75pt;
      height:36.5pt;z-index:251660288;mso-width-percent:900;
      mso-position-horizontal-relative:page;
      mso-position-vertical-relative:margin;mso-width-percent:900;
      mso-width-relative:margin" arcsize="2543f" o:allowincell="f"
      stroked="f">
      <v:shadow on="t" type="perspective" color="#4f81bd [3204]"
        origin="-.5,-.5" offset="-3pt,-3pt" offset2="6pt,6pt"
        matrix=".75,,,.75" />
      <v:textbox style="mso-next-textbox:#_x0000_s1027;
        mso-fit-shape-to-text:t" inset=",,36pt,18pt">
        <w:txbxContent>
          <w:p>
            <w:pPr>
              <w:rPr>
                <w:i />
                <w:color w:val="7F7F7F" w:themeColor="background1"
                  w:themeShade="7F" />
              </w:rPr>
            </w:pPr>
            <w:r w:rsidR="00CA19B3">
              <w:rPr>
                <w:i />
                <w:color w:val="7F7F7F" w:themeColor="background1"
                  w:themeShade="7F" />
              </w:rPr>
              <w:t>Text box</w:t>
            </w:r>
          </w:p>
        </w:txbxContent>
      </v:textbox>
      <w10:wrap type="square" anchorx="page" anchory="margin" />
    </v:roundrect>
```

```
    </w:pict>
  </w:r>
```

This general format is used for any type of textbox, such as those added automatically when a cover page is added to a document.

End of informative text.

7. Introduction to Shared MLs

This clause is informative.

7.1 Math

In this subclause, every mathematical expression is called an *equation*, even if the expression is merely a string of variables or a single object such as a fraction. In XML, an equation is called *OMath*. A Math Paragraph is a group of one or more equations separated by soft carriage returns; that is, they are separate equations that comprise a single paragraph. A Math Paragraph carries its own justification that can be separate from the justification of the paragraph that contains it. Different equations within a Math Paragraph cannot have different types of justification.

Equations can be Display (the only text on the line) or Inline (on a line with text outside of the equation). The Display vs. Inline state is not specified by this standard; instead, a text processor determines how to format Display and Inline equations. Display and Inline equations innately carry different formatting characteristics; Inline equations consume less vertical space so as not to disrupt line spacing with adjacent lines. This means, for example, reducing the size of fractions and n-ary objects that grow.

The following subclauses introduce each of the objects (also called *functions*) that comprise the majority of the Equations schema. As a text processor and not a calculation engine, when converting equations into XML representation, more attention is given to the layout and appearance than to the mathematical meaning of the expressions. That is, $\overrightarrow{abc}\overrightarrow{abc}$ and $\overbrace{abc}\overbrace{abc}$ are represented with the same object, although they carry different mathematical meanings, because both consist of text paired with a stretching character. Similarly, $\frac{n}{k}$ and $\binom{n}{k}$ are represented as the same object. Though mathematically they have different meaning, their layout is similar.

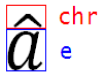
Although the functionality described in this clause is purely about the appearance of equations, other markup defined in this Office Open XML Standard provides independent functionality enabling calculation of mathematical expressions. Formulas in SpreadsheetML (§3.15.1) and Fields in WordprocessingML (§2.17.1) are two examples.

7.1.1 Accent Object

Consider the following letters having diacritical marks:

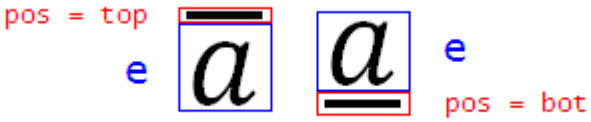
$\dot{a}\ \ddot{a}\ \tilde{a}\ \hat{a}\ \vec{a}$

The accent object is used to represent any baseline text having a combining diacritical mark placed above the base. The accent has only one child, the base element. The accent mark itself is stored as a property. In the examples above, the only difference in the XML representations is the character.



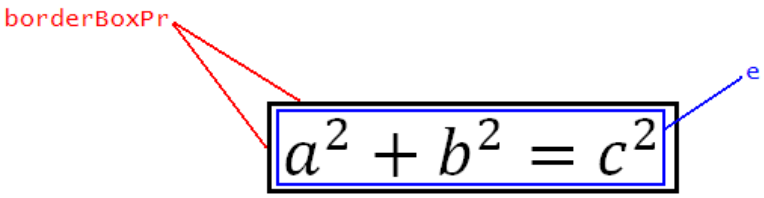
7.1.2 Bar Object

The bar object consists of baseline text with a bar drawn above or below the base. The bar has only one child, the base element. The location of the bar is stored as a property. For example:

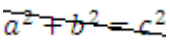


7.1.3 Border Box Object

The Border Box object consists of math text—often a formula the author wishes to call out or give special attention—surrounded by a border. Any combination of the edges of the border can be hidden. For example:



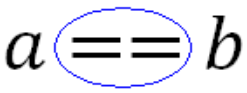
The Border Box can also be used to "cross out" text with a horizontal, vertical, or diagonal (from top-left to bottom-right or from top-right to bottom-left) strikethrough, as follows:



7.1.4 Box Object

The Box object is used to group components of an equation, to apply a single property to everything in the box. The Box serves a number of distinct purposes, including grouping characters to form a single operator (an operator emulator), and thereby inheriting the alignment and manual break properties of operators; grouping a differential such as dx; preventing line breaks from occurring within; and allowing text inside to be reduced in script level.

An example of a Box serving as an operator emulator is:

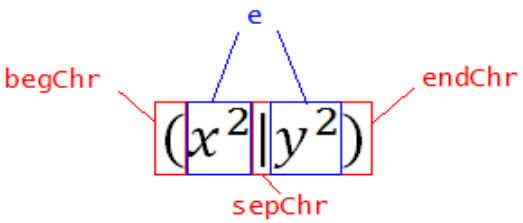


7.1.5 Delimiters

Delimiters consist of opening and closing delimiting characters (such as parentheses, braces, brackets, and vertical bars), and an element contained inside. If two or more elements are contained within delimiters, separating characters are used.

Delimiters can grow to the height of the object they contain. For example, parentheses could grow quite tall to enclose this multi-row matrix: $\begin{pmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & 1 \end{pmatrix}$. Or, at the user's discretion, they can maintain their height regardless of the content inside, as in $\left[\frac{a}{b+\frac{c}{d}}\right]$.

Delimiters have a single type of child, the base argument, which can be used multiple times in the object to signify that a separator character is to be used. For example:



If the separator character is not specified in XML, the vertical bar is used.

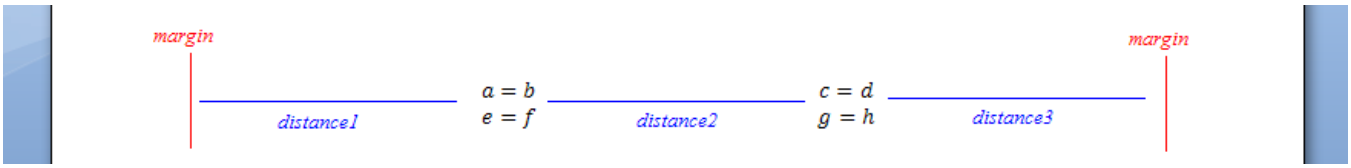
7.1.6 Equation Array Object

The Equation Array object consists of one or more equations grouped as an object. Within the equation array, multiple components can be aligned to each other. Examples of equation arrays are:

$$\begin{aligned} x - y + z &= 10 \\ 3x + y + 2z &= 34 \\ -5x + 2y - z &= -14 \end{aligned}$$ and $$f(x) = \begin{cases} -x, & x < 0 \\ x, & x \geq 0 \end{cases}$$

Equation arrays can have "maximum distribution" such that they occupy the entire width of the column that contains them, as in:

Or, they can have "object distribution" such that there is even spacing between the margin and text (distance1 = distance2 = distance 3):



7.1.7 Fraction Object

The Fraction object consists of a numerator and denominator separated by a fraction bar. The Fraction object is used to classify the different styles of fractions. It is also used to classify the stack object, which places one element above another, with no fraction bar. The four types of fractions are shown below:

Stacked Fraction: $\frac{a}{b}$

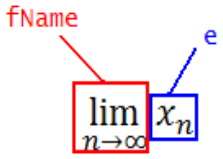
Skewed Fraction: a/b

Linear Fraction: a/b

Stack (noBar) Fraction: $\begin{matrix} n \\ k \end{matrix}$

7.1.8 Function Apply Object

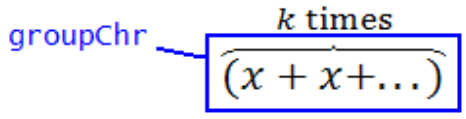
The Function Apply object consists of a function name (or object) applied to a base. The function name, by default, does not use math italics. The Function Apply object consists of a function name (a string or object) and a base element acted upon, as in:



The user can modify the text in a function name, or can add strings to be recognized automatically by the text processor as function names.

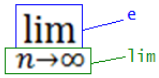
7.1.9 Group Character Object

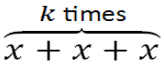
The Group Character object consists of a character drawn above or below text, often with the purpose of visually grouping items. In the following example, the text above the overbrace is not part of the group character object; it is included only to demonstrate a real-world example of the object in use:



7.1.10 Upper and Lower Limits

Upper Limits and Lower Limits are treated as separate (but similar) objects in the XML representation. Both consist of text on the baseline and reduced-size text immediately above or below it. Examples include:



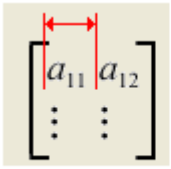
and 

, where in the second example the upper limit is k times and the base is $\overbrace{x+x+\cdots}$.

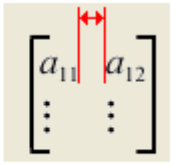
7.1.11 Matrix Object

The Matrix object consists of one or more elements laid out in one or more rows and one or more columns (delimiters not included). Examples include $\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$ and $\begin{bmatrix} 1 & \\ & 1 \end{bmatrix}$.

The entire matrix can be aligned, with respect to the surrounding text, at the center, with the top row, or with the bottom row. This property is defined as baseJc. Spacing between columns can be defined using cGp, cGpRule, and cSp. Column Gap refers to the space between the end of one column and the start of the next; column spacing refers to the space between two corresponding edges of adjacent columns.

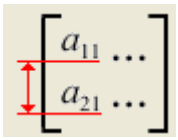


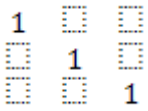
Column Spacing



Column Gap

Row spacing can also be defined using rSp and rSpRule. Row spacing is defined as the distance between baselines on adjacent matrix rows:



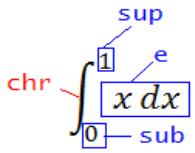
Finally, a matrix can have hidden placeholders (hidePlc). The identity matrix above has hidden placeholders, while the following matrix has placeholders showing: 

7.1.12 N-ary Object

The N-ary object consists of an n-ary object, a base (or operand), and optional upper and lower limits, as in:

$$\int_0^1 x\,dx \qquad \sum_k \binom{n}{k} \qquad \prod_{k=1}^{n} A_k \qquad \bigcup_{n=1}^{m} (X_n \cap Y_n)$$

The components of an n-ary object are as follows:



Other properties of the n-ary object are:

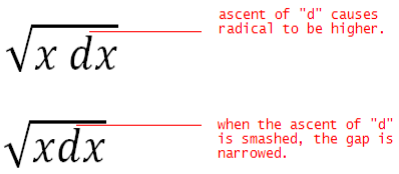
- grow: specifies whether the n-ary object grows to the height of its operand, or stays a fixed height
- limLoc: specifies the placement of n-ary limits: either to the right of the n-ary operator (the subSup position) or centered above and below (the undOvr position).
- supHide: specifies that the upper limit is hidden and no placeholder shows
- subHide: specifies that the upper limit is hidden and no placeholder shows

7.1.13 Phantom Object

The Phantom object allows extra spacing, horizontal, vertical, or both, to be added or suppressed during layout for enhanced appearance.

In the following example, the two radicals are unbalanced: $\sqrt{\frac{a}{b}} = \sqrt{x}$. For enhanced typography, the radical bars and bottom points should line up. To accomplish this, the user should adjust the height of the second radical, to make it the height of the fraction. However, no extra padding should be added to the width. The user can accomplish this by inserting a phantom of the fraction under the second radical, as in: $\sqrt{\frac{a}{b}} = \sqrt{\frac{}{x}}$. In this case, the radicals line up, and the phantom fraction acts as ghost text that adds vertical space but no width (zeroWid). The phantom can also be used to add horizontal space, alone or in conjunction with vertical space.

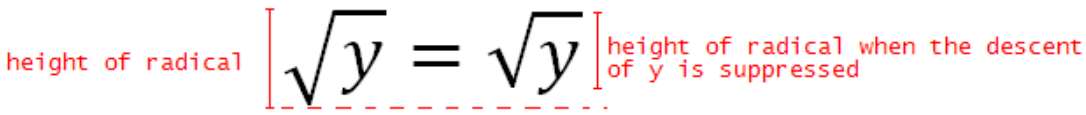
Phantoms are not always invisible. The "smash" is a type of phantom in which the content remains visible. However, part or all of the smash can be ignored during layout of text around it. For example, examine the following two radicals:



A discerning typographer might desire less vertical spacing between the tip of the "d" and the radical bar in the first example. By placing the differential in a smash and assigning it zero height (zeroAsc), the spacing is reduced.

Note that in this same example, when the differential term is placed inside a phantom, the spacing between the first and second characters changes. Again, the discerning typographer wishes that despite the presence of the phantom, differential spacing is retained. By assigning the phantom transparency for spacing (transp), proper spacing is preserved.

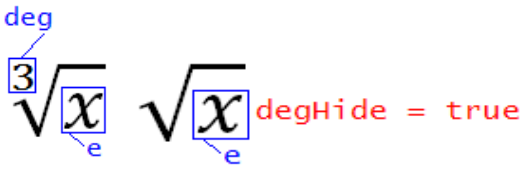
Finally, zeroDesc phantom allows the descent of the phantom base to be ignored during layout. The following example illustrates the usage of zeroDesc:



Each of the phantom properties can be applied whether the phantom is visible or hidden (the show property).

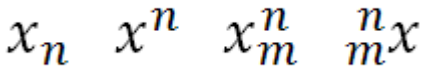
7.1.14 Radical Object

The Radical object consists of a radical, a base e, and an optional degree. When the degree is not shown and a placeholder character is not to appear, the property degHide is used.

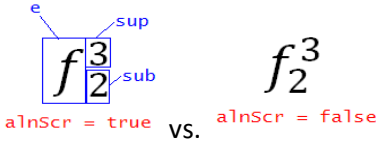


7.1.15 Scripts (Superscript, Subscript, SubSuperscript, PreSubSuperscript)

There are four distinct but related objects that consist of a base and a smaller “script” term either raised or lowered, on the left or right of the base. These are the Subscript, Superscript, SubSuperscript, and PreSubSuperscript:



The SubSuperscript has the option of aligning scripts (alnScr), as in:

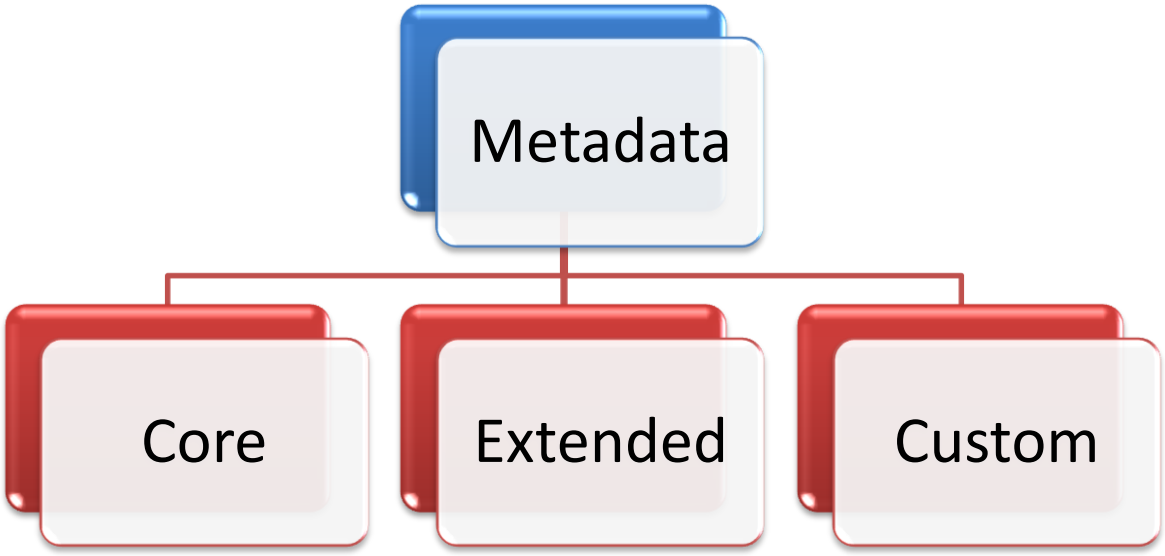


vs.

.

7.2 Metadata

Office Open XML document metadata consists of 43 well-defined properties and user-defined custom properties. Metadata properties are divided into three categories: Core, Extended, and Custom.



Each metadata category is represented by a document part with a corresponding relationship type, content type, and schema. Each metadata property is associated with exactly one metadata part.

The following table lists all well-defined metadata properties:

Property	Category
category	Core
contentStatus	Core
contentType	Core
Created	Core
Creator	Core
description	Core
identifier	Core
keywords	Core
language	Core
lastModifiedBy	Core
lastPrinted	Core
modified	Core

Property	Category
revision	Core
subject	Core
title	Core
version	Core
Application	Extended
AppVersion	Extended
Characters	Extended
CharactersWithSpaces	Extended
Company	Extended
DigSig	Extended
DocSecurity	Extended
HeadingPairs	Extended
HiddenSlides	Extended
HLinks	Extended
HyperlinkBase	Extended
HyperlinksChanged	Extended
Lines	Extended
LinksUpToDate	Extended
Manager	Extended
MMClips	Extended
Notes	Extended
Pages	Extended
Paragraphs	Extended
PresentationFormat	Extended
ScaleCrop	Extended
SharedDoc	Extended
Slides	Extended
Template	Extended
TitlesOfParts	Extended
TotalTime	Extended
Words	Extended

7.2.1 Metadata Properties

Metadata properties are represented as XML elements with associated name and type. There are two types of properties: simple and complex. Simple properties are singular XML elements whose type and value is defined

by the type and value of that XML element. Complex properties contain nested variant type XML elements that define the type and value of complex data such as arrays and vectors. Metadata properties are non-repeatable and must be defined within their associated metadata part. All metadata properties may be empty or omitted. If all properties of a metadata part are omitted, that part may be excluded from the document.

Simple property and custom complex property

```
<dc:creator>John Smith<dc:creator>
<property fmtid="{D5CDD505-2E9C-101B-9397-08002B2CF9AE}" pid="2" name="Editor">
  <vt:lpwstr>John Smith</vt:lpwstr>
</property>
```

7.2.2 Core Properties

Core properties are a predefined set of metadata properties common to all packages, and are discussed in detail in §11 of Part 2 of this Standard: "Open Packaging Conventions".

7.2.3 Extended Properties

Extended properties are a predefined set of metadata properties that are specifically applicable to Office Open XML documents. Extended properties consist of 24 simple properties and 3 complex properties stored in the part targeted by the relationship of type:
http://schemas.openxmlformats.org/officeDocument/2006/relationships/extended-properties.

7.2.4 Custom Properties

Custom properties allow users to extend pre-defined metadata properties with user-defined properties. Custom properties are stored in the part targeted by the relationship of type:
http://schemas.openxmlformats.org/officeDocument/2006/relationships/custom-properties. Each property is represented as a property XML element and uniquely identified through the name, fmtid, and pid attributes. All custom properties are considered complex properties. The type and value of custom properties are specified by its child variant type XML elements.

7.2.5 Variant Types

Office Open XML defines 35 XML elements representing commonly-used variant types to enable the representation and round-tripping of complex data. Variant type XML elements are used as child elements of complex metadata properties to define the type and value.

7.3 Custom XML Data

Within an Office Open XML document, it is sometimes desirable or necessary to store custom XML data (that is, data in a format not defined by this Office Open XML specification) within the contents of the package. To accommodate this need, Office Open XML allows the storage of any arbitrary XML within a package as the target of the http://schemas.openxmlformats.org/officeDocument/2006/relationships/customXml relationship (valid source parts are listed within Part 1 of the Standard).

The following examples illustrate potential uses of this mechanism:

- A document which collects and displays information from a backend data source might want to store the original form of that backend data, so it can be manipulated and uploaded to the original data source at a later date.
- A document author may wish to store additional metadata in an XML format not defined by this Office Open XML specification's existing metadata schemas.
- A document management system may wish to store data tracking the workflow status, retention policies, and so on for this document along with the document.

Once present within a package, this custom XML data shall be maintained as a distinct part separate from the contents of the document, spreadsheet, or presentation. If there are multiple distinct streams of custom XML data, each is maintained as a separate part within the package, so that each can be manipulated independently of all others.

Each Custom XML Data part may also have an implicit relationship to a Custom XML Data Properties part which stores:

- The target namespace of all XML schemas which shall be used to validate the content of the custom XML data part.
- A GUID which shall remain constant over the lifetime of this part (and can therefore be used to uniquely identify it).

7.4 Bibliography

Office Open XML offers functionality to store bibliography entries to permit automatic formatting of citations and bibliographies in the document according to a set of documentation rules defined in an XSLT.

7.4.1 Types of Sources

The Office Open XML formats support a collection of predefined source types for bibliography entries based on the categories most commonly used in various citation and bibliography style guidelines . The set of predefined source types can be extended as needed. The recommended approach for extending this set is to use the `Misc` type, and then leverage the methods described in Part 5 of this standard for extending the format with new attributes or elements. The following types of sources are predefined:

- `Book` (Book)
- `BookSection` (Book Section)
- `JournalArticle` (Journal Article)
- `MagOrNewsArticle` (Magazine or Newspaper Article)
- `ConferenceProceedings` (Conference Proceedings)
- `Report` (Report)
- `SoundRecording` (Sound Recording)
- `Performance` (Performance)
- `Art` (Art)
- `DocumentFromInternetSite` (Document from Internet Site)

- `InternetSite` (Internet Site)
- `Film` (Film)
- `Interview` (Interview)
- `Patent` (Patent)
- `ElectronicSource` (Electronic Source)
- `Case` (Case)
- `Misc` (Miscellaneous)

7.4.2 Child Elements

Each Source element has a number of elements as children, each of which represents a different piece of data for the bibliography entries. For example, a book might have an author, title, publisher, year, and city. Most are self-explanatory, but this document will pay special attention to some of the more complex children.

The child elements are:

- AbbreviatedCaseNumber
- AlbumTitle
- Author
- BookTitle
- Broadcaster
- BroadcastTitle
- CaseNumber
- ChapterNumber
- City
- Comments
- ConferenceName
- Country
- CountryRegion
- Court
- Day
- DayAccessed
- Department
- Distributor
- Edition
- Guid
- Institution
- InternetSiteTitle
- Issue
- JournalName
- LCID
- Medium

- Month
- MonthAccessed
- NumberVolumes
- Pages
- PatentNumber
- PeriodicalTitle
- PlacePublished
- ProductionCompany
- PublicationTitle
- Publisher
- RecordingNumber
- RefOrder
- Reporter
- SourceType
- ShortTitle
- StandardNumber
- StateProvince
- Station
- Tag
- Theater
- ThesisType
- Title
- Type
- URL
- Version
- Volume
- Year
- YearAccessed

An example of the XML defining a source of type `Book` with the title `Office Open XML formats`, and two Authors named `Jones, Brian` and `Davis, Tristan` is:

```
<b:Source>
  <b:Tag>Las07</b:Tag>
  <b:SourceType>Book</b:SourceType>
  <b:Author>
    <b:Author>
      <b:NameList>
        <b:Person>
          <b:Last>Jones</b:Last>
          <b:First>Brian</b:First>
        </b:Person>
```

```
            <b:Person>
              <b:Last>Davis</b:Last>
              <b:First>Tristan</b:First>
            </b:Person>
          </b:NameList>
        </b:Author>
      </b:Author>
      <b:Title>Office Open XML formats</b:Title>
      <b:Year>2007</b:Year>
      <b:City>Trondheim</b:City>
      <b:Publisher>Publisher</b:Publisher>
      <b:Comments>Comments</b:Comments>
      <b:RefOrder>1</b:RefOrder>
      <b:Guid>{DCC25FA1-67CC-4013-B56A-2D42CED7FF0C}</b:Guid>
      <b:LCID>0</b:LCID>
    </b:Source>
```

7.4.3 Author

There are two elements with the same name: Author. The first Author element is a container for the set of contributors attributed to the current source. The second Author element is a child of the first and is used to represent a single contributor. The valid set of contributors is defined as:

- Artist
- Author
- BookAuthor
- Compiler
- Composer
- Conductor
- Counsel
- Director
- Editor
- Interviewee
- Interviewer
- Inventor
- Performer
- ProducerName
- Translator
- Writer

For example, a bibliographic source with an author (Davis, Tristan), editor (Jaeschke, Rex), and translator (Jones, Brian) would be represented by a group of elements representing the three contributors and their specific roles inside an outer Author element, as in:

```
<b:Author>
  <b:Author>
    <b:NameList>
      <b:Person>
        <b:Last>Davis</b:Last>
        <b:First>Tristan</b:First>
      </b:Person>
  </b:Author>
  <b:Editor>
    <b:NameList>
      <b:Person>
        <b:Last>Jaeschke</b:Last>
        <b:First>Rex</b:First>
      </b:Person>
  </b:Editor>
  <b:Translator>
    <b:NameList>
      <b:Person>
        <b:Last>Jones</b:Last>
        <b:First>Brian</b:First>
      </b:Person>
  </b:Translator>
</b:Author>
```

7.4.4 LCID, Guid, Tag, and RefOrder

Four of the child elements for the Source element support important functionality for consuming applications that generate bibliographies using an externally defined stylesheet. The LCID element describes the language to be used when displaying the bibliography entry. This piece of data provides an instruction to the consuming application on the grammar of the citations and bibliography (including international name formats, date formats, and punctuation marks).

The Guid and Tag elements can be leveraged if an application wishes to uniquely identify the bibliography entry described in the Source element. For example, when a Source element is brought into a document and the Tag value for that Source element matches that of another Source element already in the document, the existing Source elements values could be overwritten with the new Source element. Two Source elements with the same Tag element value cannot exist in the same document. GUIDs can then be used in conjunction with Tags to indicate whether a Source element has been edited. When a Source element from one document has been edited, an application may decide to apply the edits to matching Source elements in other documents.

The RefOrder element for a source indicates the position, in numeric sequence, for the first reference to the source within the document text. This information is used in bibliography styles that sort sources by order in the document rather than alphabetical order.

End of informative text.

8. Miscellaneous Topics

This clause is informative.

8.1 Additional Characteristics

Office Open XML provides a way for a producer to provide information to consumers regarding how the data was created and how it should be interpreted. This information is provided by one or more *additional characteristics*.

A producing application has the option of writing out as many or as few additional characteristics as desired.

A consuming application has the option of acting on the additional characteristics or ignoring them

The additional characteristics are stored in a separate XML part, as follows:

```
<additionalCharacteristics>
  <characteristic name='name of characteristic'
    relation='well defined set of relation types'
    val='string' vocabulary='uri'/>
</additionalCharacteristics>
```

For example, consider the case in which `numColumns` is the characteristic name to specify the maximum number of columns supported by the producing SpreadsheetML application, so that the consuming application can understand how to distinguish cell references and variables unambiguously.

The relation attribute specifies the way in which the val attribute should be interpreted. The possible values for relation are: `lt | le | eq | gt | ge`, which mean <, <=, =, >=, >, respectively, and relate to numerical comparison for values and alphabetical comparison for ordering of strings. These relations permit expression of the maximum value, the minimum value, the value, and so on.

The vocabulary attribute is a URI that provides a namespace for the specific characteristic names provided as values of the name attribute. This allows for the creation of a vocabulary of characteristics of interest within a given domain of application without concern for name conflict between vocabularies.

Another example use case would be for a producer to inform the consumer that the computations used to calculate the stored numbers in the SpreadsheetML formulas have a particular numeric precision expressed by the mantissa and exponent. A consumer can optionally check those values to determine whether, for example, the values should be recalculated. The XML to represent these characteristics might look like the following:

```
<additionalCharacteristics>
  <characteristic name='precisionMantissa'
    relation='gt'
    val='-9007199254740992'/>
  <characteristic name='precisionMantissa'
    relation='lt' val='9007199254740992'/>
  <characteristic name='precisionExponent'
    relation='ge' val='-1075'/>
  <characteristic name='precisionExponent'
    relation='le' val='970'/>
</additionalCharacteristics>
```

8.2 Embeddings

Office Open XML provides facilities allowing the embedding of any object within a document. For example, a WordprocessingML document might include data as an embedded SpreadsheetML document rather than a native WordprocessingML table, in order to allow that data to be edited and recalculated by a SpreadsheetML calculation engine, rather than having it stored as a static table of data.

Office Open XML provides for two classes of embedded objects:

- *Embedded Packages* - An embedded Office Open XML document embedded within another Office Open XML document, with both documents stored in the format defined by this Office Open XML specification. For example, a PresentationML document embedded within a SpreadsheetML document results in an embedded package.
- *Embedded Objects* - Any other embedded object data. The data stored in the object shall be identified by a unique string, referred to as its *ProgID*. This string shall be used to determine both the type of data and the application (if any) that shall be used to load and edit the embedded object data.

Office Open XML also allows an image to be optionally associated with the embedded object data, for use when the embedded object application and data itself is not used by the consuming application (e.g. when the object cannot be loaded – the object is from an unknown source; the object is known, but the application has chosen not to load it for performance reasons, and so on).

8.2.1 Embedded Packages

Whenever an Office Open XML document is stored as an embedded object, the embedding shall be referred to as an embedded package. Embedded packages shall be the target of the Embedded Package relationship defined in Part 1: http://schemas.openxmlformats.org/officeDocument/2006/relationships/package. this Office Open XML specification

8.2.2 Embedded Objects

For all other embeddings, the embedded object is stored in an arbitrary format defined by the application whose data is being embedded. These generic embedded objects shall be the target of the Embedded Object relationship: http://schemas.openxmlformats.org/officeDocument/2006/relationships/oleObject. When

parsing the data stored in an embedded object part, an application shall use the associated ProgID (whose location is described in the following subclauses) for the object.

8.2.3 Embeddings in a WordprocessingML Document

When an embedding is stored in a WordprocessingML document, it is stored in one of the following ways:

- In line with text - The object is displayed within the regular text stream (modifying line height and so on to accommodate it).
- Floating – The object is positioned absolutely or relatively within the document and text flow is modified as needed around it.

Each case permits the storage of both the object and the optional VML representation of the image that may be used when the object data is not used by the hosting application as follows:

8.2.3.1 Embeddings In Line With Text

When the embedding is present in line with text, it is stored as follows:

- The WordprocessingML object element specifies the presence of an embedded object in line with text.
- The child Office VML Drawing OLEObject element shall specify the details about the embedding itself, including an explicit relationship to the appropriate Embedded Package or Embedded Object part.
- The child VML shape element shall specify the presence of the image which may be used to represent the object.

For example, if we embed a SpreadsheetML worksheet in a WordprocessingML document, the following run content would be present:

```
<w:r>
  <w:object w:dxaOrig="7247" w:dyaOrig="2920">
    <v:shape id="_x0000_i1026" type="#_x0000_t75"
      style="width:362.25pt;height:146.25pt" o:ole="">
      <v:imagedata r:id="rId6" o:title="" />
    </v:shape>
    <o:OLEObject Type="Embed" ProgID="Excel.Sheet.8"
      ShapeID="_x0000_i1026" DrawAspect="Content" ObjectID="_1218026609"
      r:id="rId7" />
  </w:object>
</w:r>
```

If we examine this markup, it can be seen that:

- We have an inline embedded object, as defined by the object element.
- The OLEObject element specifies that that object is stored as an `Embed`, and that its ProgID is `Excel.Sheet.8` (the ProgID code for Microsoft Excel worksheets); it also specifies that the associated

image (when the object data cannot be used) is stored in the VML shape with a shape ID of `_x0000_i1026`.

- The associated VML shape element with an id attribute value of `_x0000_i1026` shall be used in place of the object whenever it is not loaded - this shape is typically, but is not required to be, stored in the same object element as the OLEObject element. This shape specifies its desired size and provides an explicit relationship to the part that stores the image data.

8.2.3.2 Floating Embeddings

When the embedding is present as a floating object, it is stored as follows:

- The WordprocessingML pict element specifies the presence of a floating image in the document.
- The child Office VML Drawing OLEObject element shall specify the details about the embedding itself, including an explicit relationship to the appropriate Embedded Package or Embedded Object part.
- The child VML shape element shall specify the presence of the image that may be used to represent the object in place of loading the actual object data.

For example, if we embed a SpreadsheetML worksheet in a WordprocessingML document as a floating object, the following run content would be present:

```
<w:r>
  <w:pict>
    <v:shapetype id="_x0000_t75" coordsize="21600,21600" o:spt="75"
      o:preferrelative="t" path="m@4@5l@4@11@9@11@9@5xe" filled="f"
      stroked="f">
      …
    </v:shapetype>
    <v:shape id="_x0000_s1028" type="#_x0000_t75"
      style="position:absolute;margin-left:354.75pt;margin-
      top:642.75pt;width:182.3pt;height:73.6pt;z-index:251660288">
     <v:imagedata r:id="rId4" o:title="" />
    </v:shape>
    <o:OLEObject Type="Embed" ProgID="Excel.Sheet.8"
      ShapeID="_x0000_s1028" DrawAspect="Content" ObjectID="_1218026611"
      r:id="rId5" />
  </w:pict>
</w:r>
```

If we examine this markup, it can be seen that:

- We have a floating image, as defined by the pict element.
- The OLEObject element specifies that that floating image is actually an embedding that is stored as an Embed, and that its ProgID is `Excel.Sheet.8` (the ProgID code for Microsoft Excel worksheets); it also specifies that the associated image (when the object data cannot be used) is stored in the VML shape with a shape ID of `_x0000_s1028`.

- The associated VML shape element with an id attribute value of `_x0000_s1028` shall be used in place of the object whenever it is not loaded - this shape is typically, but is not required to be, stored in the same pict element as the corresponding OLEObject element. This shape specifies its desired size and provides an explicit relationship to the part which stores the image data.

8.2.4 Embeddings in a SpreadsheetML Document

When an embedding is present in a SpreadsheetML document, it shall be stored as follows:

- In the worksheet, the oleObjects element shall store one or more oleObject child elements, one for each embedding within the current worksheet. Each of those oleObject child elements shall also store: an explicit relationship to the associated Embedded Package or Embedded Object part, the ProgID for that embedded object, and (optionally) the last four digits of the shape ID for the associated VML shape. The shape ID itself shall be of the form `_x0000_s####`, where # specifies a single Arabic numeral, in order to be referenced as the alternate image for an embedding in a SpreadsheetML document.
- In the worksheet, the sibling legacyDrawing element shall contain an explicit relationship to the VML Drawing part that (optionally) contains the image data which may be used in place of loading the actual object data.

For example, if we embed a `Contoso Test` object (an example for illustration) in a SpreadsheetML document, the following markup would be stored in the appropriate Sheet part:

```
<s:worksheet>
  …
  <s:legacyDrawing r:id="rId9" />
  <s:oleObjects>
    <s:oleObject progId="Contoso.Test.1" shapeId="1025" r:id="rId5"/>
  </s:oleObjects>
</s:worksheet>
```

If we examine this markup, it can be seen that:

- The oleObject element specifies that we have one embedded object on the worksheet. Its attributes specify that the object is of type `Contoso.Test.1` and that the explicit relationship to the embedded object is `rId5`.
- The sibling legacyDrawing element specifies that the Legacy Drawing part which contains the associated legacy drawing data is contained at the target of the relationship with an ID of `rId9`.
- If we examine the VML Drawing part's contents, we'll see the shape which ends in `1025`, which contains the alternate image for the object:

```
<v:shape id="_x0000_s1025" type="#_x0000_t75" style='position:absolute;
  margin-left:240.75pt;margin-top:105.75pt;width:334.5pt;height:253.5pt;
  z-index:1' filled="t" fillcolor="window [65]" stroked="t"
  strokecolor="windowText [64]" o:insetmode="auto">
  <v:fill color2="window [65]"/>
  <v:imagedata o:relid="rId1" o:title=""/>
  <x:ClientData ObjectType="Pict">
    <x:SizeWithCells/>
    <x:Anchor>5, 1, 7, 1, 11, 63, 23, 19</x:Anchor>
    <x:CF>Pict</x:CF>
  </x:ClientData>
</v:shape>
```

8.2.5 Embeddings in a PresentationML Document

When an embedding is present in a PresentationML document, it shall be stored as follows:

- In the slide, the embedding is stored as a graphic frame using the graphicFrame element in PresentationML.
- The graphicData element for the frame shall have the appropriate URI for its contents: http://schemas.openxmlformats.org/presentationml/2006/ole. Its child element shall be the PresentationML oleObj element, which stores an explicit relationship to the associated Embedded Package or Embedded Object part, the ProgID for that embedded object, and (optionally) the shape ID for the associated VML shape.
- The Slide part shall also have an implicit relationship to a VML Drawing part that (optionally) contains the image data to be used in place of loading the actual object data.

For example, if we embed the `Equation.3` object in a PresentationML document, the following markup would be stored in the shape tree of the appropriate Slide part:

```
<p:graphicFrame>
  …
  <a:graphic>
  C:\Documents and Settings\tristand\Local Settings\Temp\Temporary Directory 4 for
embeddedObject.pptx.zip\ppt\slides\slide1.xml   <a:graphicData
      uri="http://schemas.openxmlformats.org/presentationml/2006/ole">
      <p:oleObj spid="_x0000_s1026" name="Equation" r:id="rId3"
        imgW="320" imgH="272" progId="Equation.3">
        <p:embed />
      </p:oleObj>
    </a:graphicData>
  </a:graphic>
</p:graphicFrame>
```

If we examine this markup, it can be seen that:

- The uri attribute on the graphicData element is http://schemas.openxmlformats.org/presentationml/2006/ole, which dictates that this is an embedded object
- It contains an oleObj element that specifies that the properties of the embedded object. Its attributes specify that the object is of type `Equation.3` and that the explicit relationship to the embedded object is `rId3`.
- The slide may also contain an implicit relationship to a Legacy Drawing part. If we examine the legacy drawing part's contents, the shape with ID `_x0000_s1026` (if present) defines the alternate image:

```
<v:shape id="_x0000_s1026" type="#_x0000_t75" style='position:absolute;
  left:282pt;top:24pt;width:152pt;height:129.25pt'>
  <v:imagedata o:relid="rId1" o:title=""/>
</v:shape>
```

8.3 Future Extensibility

This clause provides a high-level overview of the extensibility model for Office Open XML documents, and a description of packaging conventions in the context of DrawingML and PresentationML. Two main constructs are described: extensibility lists (extLst/ext) and alternate content blocks (AlternateContent).

To illustrate certain points, a number of examples refer to versions of a (fictitious) PresentationML consumer/producer called PML. The 2003 version is called PML 2003; the 2007 version is called PML 2007; and so on.

8.3.1 Terminology

Here are some terms useful when discussing future extensibility.

- *Round tripping* involves the interchange of documents between different consumers/producers, as well as between different versions of the same consumer/producer. The pair of consumers/producers can be on the same or different platforms. Consider the case in which a document is created by the PML 2007. This document is then opened by PML 2003, edited, and saved. The edited document is now opened and used by PML 2007. In this case, the document originally created by PML 2007 has been round-tripped through PML 2003. It is also possible to round-trip a document created by PML 2003 through PML 2007.
- A *Downrev* (or down-level) version of a consumer/producer refers to one that understands an older version of a given schema. An *Uprev* (or up-level) version of a consumer/producer refers to one that understands a newer version of a given schema. The terms Downrev and Uprev are typically used in relative reference to one another. As an example, let's consider again, the two consumer/producer PML 2003 and PML 2007, where PML 2007 was released sometime after PML 2003, and, consequently, PML 2007 understands a newer revision of the DrawingML schema then does PML 2003. PML 2003 is referred to as the Downrev version while PML 2007 is referred to as the Uprev version. It is assumed that the Downrev version has less capability than the Uprev version.

8.3.2 What is Future Extensibility?

The main objective of future extensibility is to design an infrastructure that allows the file format to be extended for representation of data structures in future versions of a given consumer/producer.

On the surface, that isn't hard to do; there could be a special extension bit bucket allocated across every existing schema element, and any future extension could be placed there. However, the problem is more complex than that. The infrastructure must allow document interoperability between current consumers/producers and future consumers/producers, some which have not yet even been designed or built. That is, future extensibility involves building forward compatibility into the document infrastructure while remaining compatible with the current version.

8.3.3 Future Extensibility Requirements

There are three design goals to be considered: visual fidelity, editability, and security.

- *Visual fidelity* involves the desire for users of two consumers/producers to see visually the same thing. This seems like a simple design goal to meet, but, in practice, is not easy to achieve. The difference lies in the capabilities of an Uprev and Downrev consumer/producer. Typically, an Uprev consumer/producer has been extended to have new base capabilities that are not present in the Downrev consumer/producer. As such, the Downrev client does not have the base primitives necessary to express visually the new capability introduced in the Uprev consumer/producer.

  Consider the case in which PML 2007 has the capability to highlight text with a given color, while PML 2003 does not. Given the desire to have PML 2007 documents interoperate with PML 2003, it is necessary for PML 2003 to some way to express visually that text highlight. For example, it might insert a picture of the highlighted text instead of inserting the text itself, since PML 2003 does know how to deal with pictures.
- *Editability* involves the desire for two consumers/producers to be able to edit the same content. Using the highlighted text example from above, despite the fact that PML 2003 and PML 2007 have different capabilities, one would still desire to edit highlighted text a regardless of which version of PML is in use. Again, this becomes difficult when the underlying capabilities of the consumer/producer versions are different.
- *Security* involves the desire to have multiple representations of the same data synchronized. This desire is referred to as security for the reason that out-of-sync representation can have dire consequences. For example, there might be multiple representations for a sensitive piece of information such as a Social Security number. If this piece of information were edited, it would be critical to keep all alternate representations in sync. What if that information were deleted altogether? If only one representation was deleted but others remained, it would be possible for one to have sensitive information in a document when the intent was to have it deleted.

One solution to try to solve the visual fidelity and editability goals is to have two equivalent representations for the same construct. In the highlighted text example above, a picture of the highlighted text (also called a rasterized version of the highlighted text) is an equivalent representation of the highlighted text itself. One

might use the highlighted text representation when the underlying consumer/producer is capable of understanding it; otherwise, the picture version would be used.

Clearly, these design goals compete with each other. While a picture representation of text is capable of capturing full visual fidelity of how extended text looks, obviously that representation doesn't offer the same editability properties of text. One can't manipulate a picture of text nearly as easily as the text itself.

The competing nature of these design characteristics requires that one choose an extensibility construct that offers the best mix of desired characteristics. It will not always be possible to have visual fidelity, editability, and security, at the same time.

8.3.4 Future Extensibility Constructs

The two extensibility constructs used to represent extensions in OOXML schemas are:

8.3.4.1 extLst/ext

The extLst construct is used for straight-up extension of existing schemas of a non-visual nature. The term *straight up* refers to the notion that sometimes extension means refining the semantics of existing constructs. In doing so, an extension sometimes overrides the meaning of previous schemas. extLst and ext were not designed for this scenario. Instead of overriding existing meaning, these two constructs purely augment existing schemas. The nature of the augmentation must be such that it does not overlap any semantics embedded in existing schema constructs.

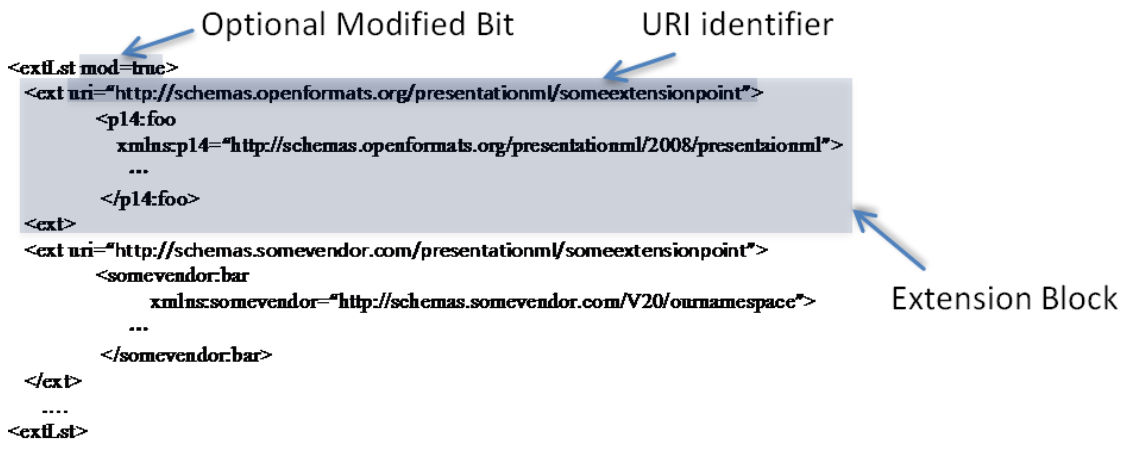
Consider a schema that represents an address, which contain a house number, a street name, a city, a state, and a postal code. An example of a straight-up extension is the addition of a field that describes whether this address is a business or residential location. This is a straight-up extension because the notion of whether an address is business or residential does not conflict with any information that is embedded in the existing schema. Now let's consider the case in which the Postal Service replaces a purely numeric postal code with one that can contain alphanumeric characters. Such a change would not be a straight-up extension because the new representation conflicts with the old representation of the same data, namely the postal code.

Some extensions are visual in nature. An example would be extending a schema to represent text that has been highlighted. By definition, highlighting text is a visual extension. Contrast that to the case of adding a business or residential classification. The latter does not necessarily involve any visual change to the way data is presented.

The extLst and ext constructs are for extensions of a non-visual nature. The main reason their use is limited to this scenario lies in the fact that they do not offer the capability to create alternative representations of the same data.

8.3.4.1.1 extLst/ext Syntax

The extLst and ext construct can be placed only at specific locations within the OOXML schemas. Its syntax is as below:



An extLst is a list of extension blocks that are placed one after the other. Each extension block has an uri attribute, which serves as an identifier to indicate the kind of extension that has been placed here. Upon encountering an extension block, a processing consumer, will determine whether it knows how to process extensions matching that attribute. If the consumer knows how to process such an extension, the markup contained within that extension block is processed. Otherwise, the extension block is preserved so long as the underlying structure being extended by the extLst has not been deleted.

There is no limit to the number of ext extension block constructs. The order of extension blocks can be arbitrary.

An optional modified attribute, mod, is available on extLst. This attribute is set to `true` whenever an edit has occurred at the extended location. Its presence is to aid up-level clients that receive modified documents that have been edited in down-level consumers/producers.

8.3.4.1.2 Round-Trip Behavior of ext Blocks

When extLsts are processed, some consumers/producers will understand some extensions, but not others. The preservation model of ext blocks is that unprocessed extensions are always preserved and retained as long as the underlying schema extended by the structure remains.

8.3.4.1.3 Example

Consider the case in which the notion that each shape can be associated with a given layer, is to be added. The schema for this might look like the following:

```
<p:sld xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/3/main"
       xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
       xmlns:p="http://schemas.openxmlformats.org/presentationml/2006/3/main">
  <p:cSld name="">
    <p:spTree>
      <p:nvGrpSpPr>...
      <p:grpSpPr>...
      <p:sp>
        <p:nvSpPr>
          <p:cNvPr id="3" name="Rectangle 3" descr=""/>
          <p:cNvSpPr/>
          <p:nvPr/>
          <p:extLst mod="false">
            <p:ext uri="http://.../layerExtension">
              <p14:spLayer xmlns:p14="http://schemas.openxmlformats.org/drawingml/2009/3/main" layer="1"/>
            </p:ext>
          </p:extLst>
        </p:nvSpPr>
        <p:spPr>...
        <p:style>...
        <p:txBody>...
      </p:sp>
    </p:spTree>
  </p:cSld>
  <p:clrMapOvr>...
  <p:timing>...
</p:sld>
```

The extLst block is under the non-visual shape properties (i.e., p:nvSpPr). A uri attribute identifies the extension.

Now consider how this markup will be processed by PML 2007 and PML 2009, where PML 2009 is an up-level version of PML 2007.

PML 2007 processes the above markup, and ignores the ext block because it doesn't understand this extension. However, this block will be preserved for any other consumer/producer that may understand it.

PML 2009 processes the above markup, and understands how to deal with layer extensions as indicated by the uri. The spLayer extension is returned, PML 2009 processes the extension and is responsible for writing out any updates to this markup, as required. For example, layer might be changed from 1 to 2.

Note that the extension is a straight-up extension in that layer information is orthogonal to all other non-visual properties, such as the ID, name, and description.

Being non-visual in nature, the information in this extension does not directly affect the appearance of the shape.

8.3.4.2 AlternateContent Blocks

An *alternate content block* allows for an alternative representation of information. In some cases, the desire will be to revise a schema with a newer representation. It will also be common to express visual differences using alternate content blocks. Recall that typically lower-level clients will not have the same capability as their future cousins (i.e., the up level version). As such, any future extension done in the up-level version will need to be expressed in a form that the lower-level version can understand. Hence, the need for alternate representations.

8.3.4.2.1 AlternateContent Syntax

```
<AlternateContent>
  <Choice Requires="namespacefoo">
    <Somemarkup/>
  <Choice Requires="namespacefoo namespacefoobar">
    <Somealternatemarkup/>
  <Fallback>
    <Choiceoflastresort/>
</AlternateContent>
```

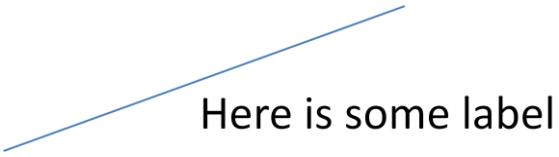
The AlternateContent element and its children, Choice and Fallback, are used to provide alternates for specified content. Each Choice element is examined in turn. The Requires attribute specifies a set of space-delimited namespaces that must be understood in order to select that choice. If there is a match between required namespaces and what the consumer understands, the appropriate Choice is returned. If there are multiple possible matches, only the first match is returned. An optional Fallback element can be used, and is utilized as a default when no match occurs.

8.3.4.2.2 Example

Using PML 2007 and PML 2009, let's assume that PML 2007 understands some current schema version while PML 2009 will understand some future extended version of the current schema.

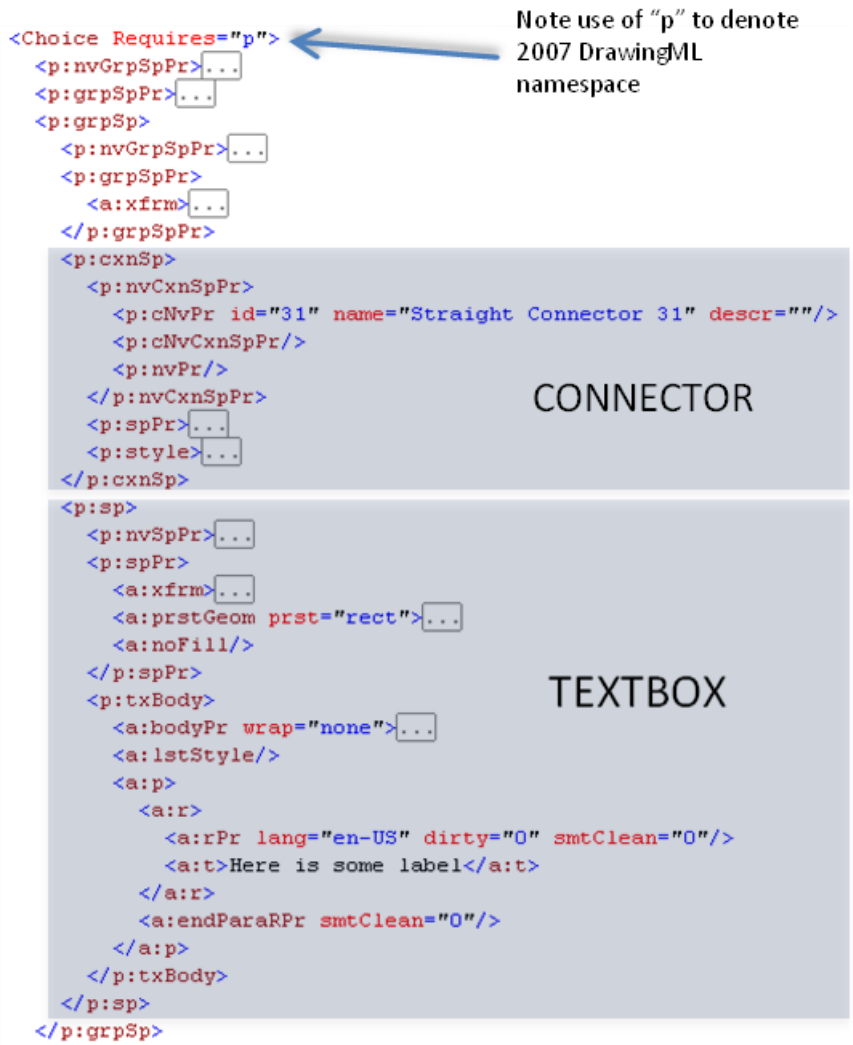
PML 2007 knows how to handle connectors:

Now let's suppose that we desire to add a notion of labels on connectors:



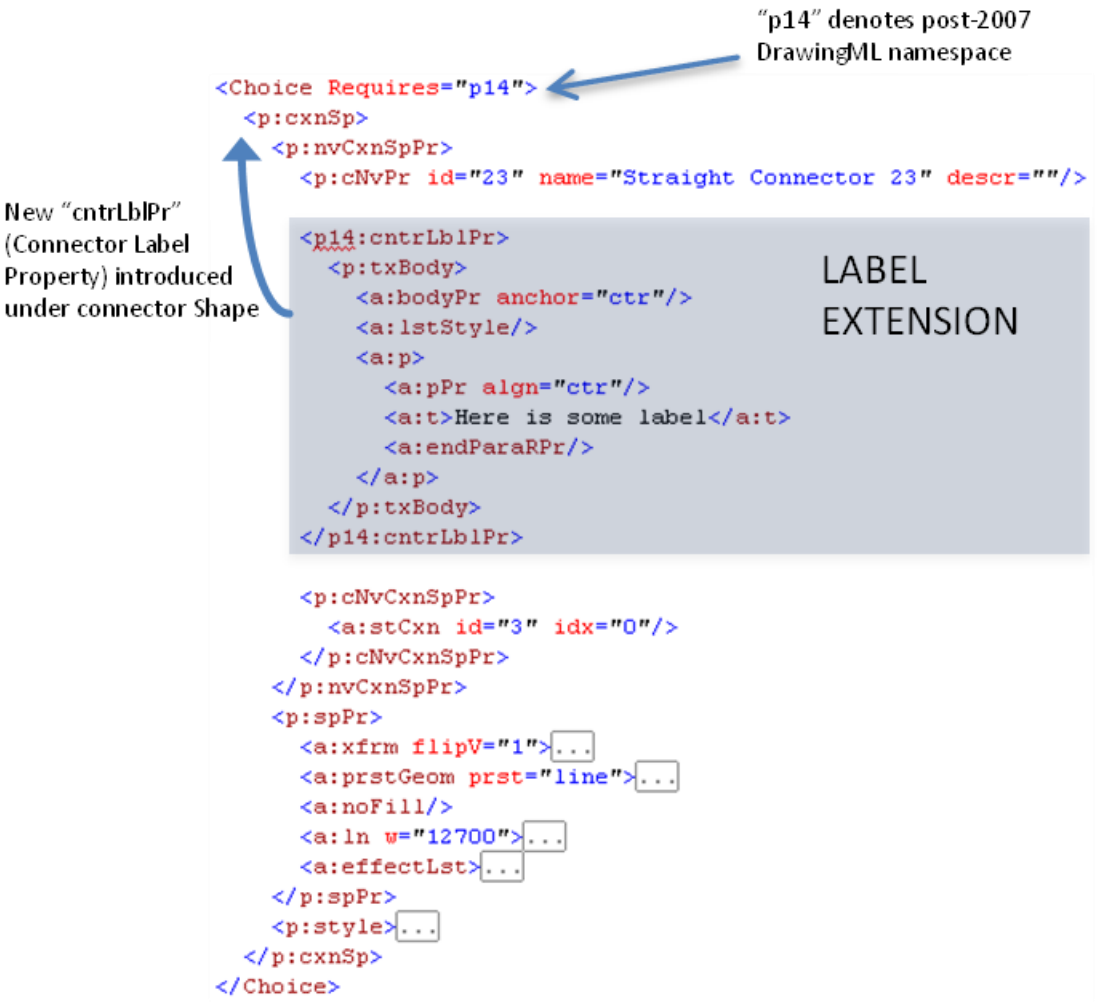
Two alternate representations are required: PML 2009's schema has been extended to natively understand how to represent a label on a connector; however, PML 2007 does not. With PML 2007, we approximate this feature by representing the connector and label independently. The label is represented using a textbox. To keep the two elements together, for convenience sake, they are grouped.

Looking at the corresponding XML, PML 2007's markup looks as follows:




```
<Choice Requires="p">
  <p:nvGrpSpPr>...
  <p:grpSpPr>...
  <p:grpSp>
    <p:nvGrpSpPr>...
    <p:grpSpPr>
      <a:xfrm>...
    </p:grpSpPr>
    <p:cxnSp>
      <p:nvCxnSpPr>
        <p:cNvPr id="31" name="Straight Connector 31" descr=""/>
        <p:cNvCxnSpPr/>
        <p:nvPr/>
      </p:nvCxnSpPr>
      <p:spPr>...
      <p:style>...
    </p:cxnSp>
    <p:sp>
      <p:nvSpPr>...
      <p:spPr>
        <a:xfrm>...
        <a:prstGeom prst="rect">...
        <a:noFill/>
      </p:spPr>
      <p:txBody>
        <a:bodyPr wrap="none">...
        <a:lstStyle/>
        <a:p>
          <a:r>
            <a:rPr lang="en-US" dirty="0" smtClean="0"/>
            <a:t>Here is some label</a:t>
          </a:r>
          <a:endParaRPr smtClean="0"/>
        </a:p>
      </p:txBody>
    </p:sp>
  </p:grpSp>
</Choice>
```

PML 2009 has been extended such that we may represent a label natively:



The final markup putting these choices together is as follows:

```
<p:sld xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/3/main"
       xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
       xmlns:p="http://schemas.openxmlformats.org/presentationml/2006/3/main"
       xmlns:p14="http://schemas.openxmlformats.org/presentationml/2008/3/main">
  <p:cSld name="">
    <p:spTree>
      <p:nvGrpSpPr>...
      <p:grpSpPr>...
      <AlternateContent>
        <Choice Requires="p14">
          <!-- Uplevel Choice goes here -->
        </Choice>
        <Choice Requires="p">
          <!-- Downlevel Choice goes here -->
        </Choice>
      </AlternateContent>
    </p:spTree>
  </p:cSld>
  <p:clrMapOvr>...
  <p:timing>...
</p:sld>
```

AlternateContent Block

8.3.4.2.3 AlternateContent Round-Trip Behavior

AlternateContent maintains multiple representations for the same content. Consider an extreme case. Using the example above, let's suppose one edited the label using PML 2007. As PML 2007 wouldn't understand future representations, there is no possibility that it could keep PML 2009's markup consistent with the edit performed. Considering a simple case, let's suppose one deleted the label in its entirety. PML 2007 would only know how to delete the corresponding textbox, and would not know how to update the corresponding cntrLblPr.

If this textbox contained sensitive information, one might consider this a security leak. The user's belief is that the information in the textbox was deleted, yet it persists in an alternate representation.

To solve this problem, all AlternateContent choices are discarded when an edit is performed at the location the AlternateContent is placed. It is the consuming client's responsibility to replace the discarded AlternateContent with a new representation.

If an edit to the label occurred in PML 2007, the PML 2009 version is discarded.

If an edit occurs in PML 2009, since PML 2009 understands both PML 2007 and PML 2009 schemas, it is possible for PML 2009 to write an updated AlternateContent Block encompassing an update to both choices.

End of informative text.

Office Open XML

Part 4: Markup Language Reference

December 2006

Table of Contents

Foreword

This multi-part Standard deals with Office Open XML Format-related technology, and consists of the following parts:

- Part 1: "Fundamentals"
- Part 2: "Open Packaging Conventions"
- Part 3: "Primer"
- **Part 4: "Markup Language Reference" (this document)**
- Part 5: "Markup Compatibility and Extensibility"

This part, Part 4, includes a number of annexes that refer to data files provided in electronic form only.

Introduction

This Part is one piece of a Standard that describes a family of XML schemas, collectively called *Office Open XML*, which define the XML vocabularies for word-processing, spreadsheet, and presentation documents, as well as the packaging of documents that conform to these schemas.

The goal is to enable the implementation of the Office Open XML formats by the widest set of tools and platforms, fostering interoperability across office productivity applications and line-of-business systems, as well as to support and strengthen document archival and preservation, all in a way that is fully compatible with the large existing investments in Microsoft Office documents.

1. Part Overview

This clause is informative.

For convenience, the following subclauses specify the root elements for each part. Full discussion of the use of each part can be found in Part 1 of this Office Open XML Standard.

1.1 WordprocessingML Part Summary

Part	Root Element	Ref.
Alternative Format Import	Not applicable	n/a
Comments	comments	§2.13.4.6
Document Settings	settings	§2.15.1.78
Endnotes	endnotes	§2.11.8
Font Table	fonts	§2.8.2.11
Footer	ftr	§2.10.3
Footnotes	footnotes	§2.11.15
Glossary Document	glossaryDocument	§2.12.10
Header	hdr	§2.10.4
Mail Merge Recipient Data	recipients	§2.14.29
Main Document	document	§2.2.3
Numbering Definitions	numbering	§2.9.17
Style Definitions	styles	§2.7.3.18
Web Settings	webSettings	§2.15.2.44

1.2 SpreadsheetML Part Summary

Part	Root Element	Ref.
Calculation Chain	calcChain	§3.6.2
Chartsheet	chartsheet	§3.3.1.11
Comments	comments	§3.7.5
Connections	connections	§3.13.2
Custom Property	Not applicable	n/a
Custom XML Mappings	mapInfo	§3.16.3
Dialogsheet	dialogSheet	§3.3.1.32
Drawing	wsDr	§5.6.2.34

Part	Root Element	Ref.
External Workbook References	externalLink	§3.14.8
Metadata	metadata	§3.9.8
Pivot Table	pivotTableDefinition	§3.10.1.73
Pivot Table Cache Definition	pivotCacheDefinition	§3.10.1.67
Pivot Table Cache Records	pivotCacheRecords	§3.10.1.68
Query Table	queryTable	§3.12.2
Shared String Table	sst	§3.4.9
Shared Workbook Revision Headers	headers	§3.11.1.1
Shared Workbook Revision Log	revisions	§3.11.1.16
Shared Workbook User Data	users	§3.11.2.2
Single Cell Table Definitions	singleCells	§3.5.2.2
Styles	styleSheet	§3.8.39
Table Definition	table	§3.5.1.2
Volatile Dependencies	volTypes	§3.15.6
Workbook	workbook	§3.2.27
Worksheet	worksheet	§3.3.1.96

1.3 PresentationML Part Summary

Part	Root Element	Ref.
Comment Authors	cmAuthorLst	§4.5.3
Comments	cmLst	§4.5.4
Handout Master	handoutMaster	§4.4.1.21
Notes Master	notesMaster	§4.4.1.24
Notes Slide	notes	§4.4.1.23
Presentation	presentation	§4.3.1.24
Presentation Properties	presentationPr	§4.3.1.25
Slide	sld	§4.4.1.35
Slide Layout	sldLayout	§4.4.1.36
Slide Master	sldMaster	§4.4.1.39
Slide Synchronization Data	sldSyncPr	§4.7.1
User-Defined Tags	tagLst	§4.4.3.2
View Properties	viewPr	§4.3.2.18

1.4 DrawingML Part Summary

Part	Root Element	Ref.

Part	Root Element	Ref.
Chart	chartSpace	§5.7.2.29
Chart Drawing	userShapes	§5.7.2.221
Diagram Colors	colorsDef	§5.9.4.3
Diagram Data	dataModel	§5.9.2.10
Diagram Layout Definition	layoutDef	§5.9.2.16
Diagram Style	styleDef	§5.9.5.7
Theme	officeStyleSheet	§5.1.8.9
Theme Override	themeOverride	§5.1.8.12
Table Styles	tblStyleLst	§5.1.4.2.27

1.5 Shared Part Summary

Part	Root Element	Ref.
Additional Characteristics	additionalCharacteristics	§7.7.2.1
Audio	Not applicable	n/a
Bibliography	Sources	§7.6.2.60
Custom XML Data Storage	Not applicable	n/a
Custom XML Data Storage Properties	datastoreItem	§7.5.2.1
Digital Signature Origin	Not applicable	n/a
Digital Signature XML Signature	Signature	Defined in Part 2
Embedded Control Persistence	Not applicable	n/a
Embedded Object	Not applicable	n/a
Embedded Package	Not applicable	n/a
File Properties, Core	coreProperties	Defined in Part 2
File Properties, Custom	Properties	§7.3.2.1
File Properties, Extended	Properties	§7.2.2.21
Font	Not applicable	n/a
Image	Not applicable	n/a
Printer Settings	Not applicable	n/a
Thumbnail	Not applicable	n/a
Video	Not applicable	n/a

End of informative text.

2. WordprocessingML Reference Material

The subordinate subclauses specify the semantics for the XML markup comprising a WordprocessingML document, as defined by Part 1 of this Standard.

2.1 Table of Contents

This subclause is informative.

End of informative text.

2.2 Main Document Story

A WordprocessingML document consists of a compilation of two types of information:

- Properties [*Example*: styles, numbering definitions, etc. *end example*]
- Stories [*Example*: main document, comments, headers, etc. *end example*]

In WordprocessingML, *stories* are defined as each unique region of content within a document into which the user can type. The most important story in a WordprocessingML document is the main document story, which contains the primary contents of the document. The main document story in WordprocessingML is stored inside the body element.

[*Example*: Consider a document with a single paragraph in the main document story. This document would require the following WordprocessingML in its main document part:

```
<w:document>
  <w:body>
    <w:p/>
  </w:body>
</w:document>
```

The fact that the paragraph is inside the body element makes it part of the main document story. *end example*]

2.2.1 background (Document Background)

This element specifies the background information for this document. This background shall be displayed on all pages of the document, behind all other document content.

The child elements of the background element are in the Vector Markup Language (VML) namespace, which allows any valid VML effect to be applied to the document's background.

For solid color fill backgrounds, however, the attributes on this element allow for the specification of use of any valid RGB or theme color value (the latter a reference to the document's themes part).

[*Example*: Consider a document which utilizes a gradient fill background moving between black and the accent5 theme color, as follows:

This background would require the following WordprocessingML markup:

```
<w:background w:color="9BBB59" w:themeColor="accent3">
  <v:background id="_x0000_s1025" o:bwmode="white" fillcolor="#9bbb59 [3206]"
o:targetscreensize="800,600">
    <v:fill color2="fill darken(118)" method="linear sigma" focus="100%"
type="gradientRadial">
      <o:fill v:ext="view" type="gradientCenter" />
    </v:fill>
  </v:background>
</w:background>
```

The resulting background consists of a single color fill of the `accent3` theme color from the themeColor attribute, layered under a `gradientCenter` fill from the fill attribute. *end example*]

Parent Elements
document (§2.2.3); glossaryDocument (§2.12.10)

Child Elements	Subclause
Any element from the urn:schemas-microsoft-com:vml namespace	§6.1
Any element from the urn:schemas-microsoft-com:office:office namespace	§6.2

Attributes	Description
color (Background Color)	Specifies the color for the background of the document. This color may either be presented as a hex value (in RRGGBB format), or auto to allow a consumer to automatically determine the background color as appropriate. If the background specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. [*Example*: Consider a border color with value 2C34FF, as follows: `<w:background … w:color="2C34FF"/>` The background color shall therefore be the color with an RGB value of 44,52,255 (the decimal decoding of the hex value above). *end example*] The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
themeColor (Background Theme Color)	Specifies a theme color to be applied to the current background. The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part, which allows color information to be set centrally in the document. If the color attribute is specified, its value shall be ignored in favor of the color resulting from the use of this attribute with any appropriate themeTint and themeShade attribute value calculations applied. [*Example*: Consider a background configured to use the accent5 theme color, resulting in the following WordprocessingML markup: `<w:background w:color="FFA8A0" w:themeColor="accent5" />` The background has a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, the RGB color value is ignored in favor of the accent5 theme color specified for this document. *end example*] The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).
themeShade	Specifies the shade value applied to the supplied theme color (if any) for this background.

Attributes	Description
(Border Theme Color Shade)	If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 60% applied to a background in a document. This shade is calculated as follows: $$S_{xml} = 0.6 * 255 = 153 = 99(hex)$$ The resulting themeShade value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}^{'} = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L^{'} = 0.53 * 0.75 = 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:background w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this background. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to the document's background. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.4 * 255 = 102 = 66(hex)$$ The resulting themeTint value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6) = 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$ and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting background's color attribute: `<w:background w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>`

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Background">
   <complexContent>
      <extension base="CT_PictureBase">
         <attribute name="color" type="ST_HexColor" use="optional"/>
         <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
         <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
         <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

2.2.2 body (Document Body)

This element specifies the contents of the body of the document - the main document editing surface.

The document body contains what is referred to as *block-level markup* - markup which can exist as a sibling element to paragraphs in a WordprocessingML document.

[*Example*: Consider a document with a single paragraph in the main document story. This document would require the following WordprocessingML in its main document part:

```
<w:document>
  <w:body>
    <w:p/>
  </w:body>
</w:document>
```

The fact that the paragraph is inside the body element makes it part of the main document story. *end example*]

Parent Elements
document (§2.2.3)

Child Elements	Subclause
altChunk (Anchor for Imported External Content)	§2.17.3.1
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3

Child Elements	Subclause
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Block-Level Custom XML Element)	§2.5.1.6
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
del (Deleted Run Content)	§2.13.5.12
ins (Inserted Run Content)	§2.13.5.20
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
p (Paragraph)	§2.3.1.22
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
sdt (Block-Level Structured Document Tag)	§2.5.2.30
sectPr (Document Final Section Properties)	§2.6.18
tbl (Table)	§2.4.36

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Body">
   <sequence>
      <group ref="EG_BlockLevelElts" minOccurs="0" maxOccurs="unbounded"/>
      <element name="sectPr" minOccurs="0" maxOccurs="1" type="CT_SectPr"/>
   </sequence>
</complexType>
```

2.2.3 document (Document)

This element specifies the contents of a main document part in a WordprocessingML document.

[*Example*: Consider the basic structure of the main document part in a basic WordprocessingML document, as follows:

```
<w:document>
  <w:body>
    <w:p/>
  </w:body>
</w:document>
```

All of the contents of the main document part are contained beneath the document element. *end example*]

Parent Elements
Root element of WordprocessingML Main Document part

Child Elements	Subclause
background (Document Background)	§2.2.1
body (Document Body)	§2.2.2

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Document">
   <complexContent>
      <extension base="CT_DocumentBase">
         <sequence>
            <element name="body" type="CT_Body" minOccurs="0" maxOccurs="1"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.3 Paragraphs and Rich Formatting

The basis of a WordprocessingML document is its actual text contents. Those text contents can be stored in many contexts (tables, text boxes, etc.), but the most basic form of text contents in WordprocessingML is the paragraph, specified using the p element (§2.3.1.22).

Within the paragraph, all rich formatting at the paragraph level is stored within the pPr element (§2.3.1.25; §2.3.1.26). [*Note*: Some examples of paragraph properties are alignment, border, hyphenation override, indentation, line spacing, shading, text direction, and widow/orphan control. *end note*]

Within the paragraph, text is grouped into one or more runs, represented by the r element (§2.3.2.23), which define a region of text with a common set of properties.

Just as a paragraph can have rich formatting, so too can a run. All of the elements inside an r element have their properties controlled by a corresponding optional rPr run properties element (§2.7.8.1; §2.3.2.26). [*Note*: Some examples of run properties are bold, underlined, or visible. *end note*]

Within runs, run content is the set of possible objects and characters which can be displayed in the document.

2.3.1 Paragraphs

The most basic unit of block-level content within a WordprocessingML document, *paragraphs* are stored using the p element (§2.3.1.22). A paragraph defines a distinct division of content with a WordprocessingML document which begins on a new line.

[*Example*: Consider the paragraph fragment "*The quick brown fox jumped … "* which is centered on a paragraph. The justification property is a paragraph level property, and therefore is expressed on the paragraph properties as follows:

```
<w:p>
  <w:pPr>
    <w:jc w:val="center"/>
    <w:rPr>
      <w:i/>
    </w:rPr>
  </w:pPr>
  <w:r>
    <w:rPr>
      <w:i/>
    </w:rPr>
    <w:t xml:space="preserve">The quick brown fox jumped … </w:t>
  </w:r>
</w:p>
```

Notice that each run specifies the character formatting information for its contents, and the paragraph specifies the paragraph level formatting (the center-justification). It is also notable that since leading and trailing whitespace is not normally significant in XML; some runs require a designating specifying that their whitespace is significant via the xml:space element. *end example*]

A paragraph's properties are specified via the pPr element (§2.3.1.25; §2.3.1.26). [*Note*: Some examples of paragraph properties are alignment, border, hyphenation override, indentation, line spacing, shading, text direction, and widow/orphan control. *end note*]

2.3.1.1 adjustRightInd (Automatically Adjust Right Indent When Using Document Grid)

This element specifies whether the right indent shall be automatically adjusted for the given paragraph when a document grid has been defined for the current section using the docGrid element (§2.6.5), modifying of the current right indent used on this paragraph.

[*Note*: This setting is used in order to ensure that the line breaking for that paragraph is not determined by the width of the final character on the line. *end note*]

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, its value is assumed to be `true`.

[*Example*: Consider a paragraph in which the right indent on the current paragraph should not be automatically determined based on the character pitch set in the document grid. This setting would be specified using the following WordprocessingML:

```
<w:p>
  <w:pPr>
    …
    <w:adjustRightInd w:val="false" />
  </w:pPr>
  …
</w:p>
```

By explicitly setting the val to `false`, this paragraph will use its specified right indent settings regardless of the presence of the document grid for the parent section. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.1.2 autoSpaceDE (Automatically Adjust Spacing of Latin and East Asian Text)

This element specifies whether inter-character spacing shall automatically be adjusted between regions of Latin text and regions of East Asian text in the current paragraph. These regions shall be determined by the Unicode character values of the text content within the paragraph.

[*Note*: This property is used to ensure that the spacing between regions of Latin text and adjoining East Asian text is sufficient on each side such that the Latin text can be easily read within the East Asian text. *end note*]

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, its value is assumed to be `true`.

[*Example*: Consider a paragraph in which the spacing should not be automatically adjusted based on the presence of Latin and East Asian text. This setting would be specified using the following WordprocessingML:

```
<w:p>
  <w:pPr>
    …
    <w:autoSpaceDE w:val="false" />
  </w:pPr>
  …
</w:p>
```

By explicitly setting the val to `false`, this paragraph shall not automatically adjust the spacing of adjoining Latin and East Asian text. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property:

Attributes	Description
	<w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.1.3 autoSpaceDN (Automatically Adjust Spacing of East Asian Text and Numbers)

This element specifies whether inter-character spacing shall automatically be adjusted between regions of numbers and regions of East Asian text in the current paragraph. These regions shall be determined by the Unicode character values of the text content within the paragraph.

[*Note*: This property is used to ensure that the spacing between regions of numbers and adjoining East Asian text is sufficient on each side such that the numbers can be easily read within the East Asian text. *end note*]

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, its value is assumed to be true.

[*Example*: Consider a paragraph in which the spacing should not be automatically adjusted based on the presence of numbers and East Asian text. This setting would be specified using the following WordprocessingML:

```
<w:p>
  <w:pPr>
    …
    <w:autoSpaceDN w:val="false" />
  </w:pPr>
  …
</w:p>
```

By explicitly setting the val to false, this paragraph will not automatically adjust the spacing of adjoining numbers and East Asian text. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.1.4 bar (Paragraph Border Between Facing Pages)

This element specifies the border which may be displayed on the inside edge of the paragraph when the parent's section settings specify that the section shall be printed using mirrored margins using the mirrorMargins element (§2.15.1.57). [*Note*: This information is present in the WordprocessingML for the purposes of legacy document format compatibility, and it may be removed and/or ignored as required. *end note*]

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then no bar border shall be applied to the current paragraph.

[*Example*: Consider the following paragraph's WordprocessingML definition for its paragraph borders:

```
<w:p>
  <w:pPr>
    <w:pBdr>
      <w:top w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
        w:themeColor="accent2" w:themeTint="33" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="B97034"
        w:themeColor="accent6" w:themeShade="BF" />
      <w:bottom w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
        w:themeColor="accent2" w:themeTint="33" />
```

```
      <w:right w:val="single" w:sz="24" w:space="4" w:color="C3D69B"
        w:themeColor="accent3" w:themeTint="99" />
      <w:bar w:val="single" w:sz="24" w:space="1" w:color="4F81BD"
        w:themeColor="accent1" />
    </w:pBdr>
  </w:pPr>
  <w:r>
    <w:t>Sample paragraph.</w:t>
  </w:r>
</w:p>
```

This paragraph has a single line bar border as defined by the bar element. *end example*]

Parent Elements
pBdr (§2.3.1.24)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or auto to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value auto, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML:

Attributes	Description
	<w:bottom w:frame="true" ... /> This frame's val is true, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: <w:bottom w:shadow="true" ... /> This frame's val is true, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object. When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points. When a document has a page border that is relative to the text extents (using a value of text in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points. [*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows: <w:pgBorders w:offsetFrom="page"> <w:bottom … w:space="24/> </w:pgBorders The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).

sz (Border Width)	Specifies the width of the current border.
	
	If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value.
	
	If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value.
	
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`
	
	The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	
	The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).

themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border.
	
	The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document.
	
	[*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup:
	
	`<w:top … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:bottom … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:left … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:right … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />

Attributes	Description
	The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*] The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).
themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255$$ $$= 102$$ $$= 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L' = 0.53 * 0.75$$ $$= 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634.

Attributes	Description
	This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255 = 153 = 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6) = 0.71$$

Attributes	Description
	Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$ and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <attribute name="val" type="ST_Border" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
   <attribute name="space" type="ST_PointMeasure" use="optional"/>
   <attribute name="shadow" type="ST_OnOff" use="optional"/>
   <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.3.1.5 between (Paragraph Border Between Identical Paragraphs)

This element specifies the border which shall be displayed between each paragraph in a set of paragraphs which have the same set of paragraph border settings.

To determine if any two adjoining paragraphs should have a between border or an individual top and bottom border, the set of borders on the two adjoining paragraphs are compared. If the border information on those two paragraphs is identical for all possible paragraphs borders, then the between border is displayed. Otherwise, each paragraph shall use its bottom and top border, respectively. If this border specifies a space attribute, that value is ignored - this border is always located at the bottom of each paragraph with an identical following paragraph, taking into account any space after the line pitch.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then no between border shall be applied between identical paragraphs.

[*Example*: Consider the following two paragraphs' WordprocessingML definition:

```
<w:p>
  <w:pPr>
    <w:pBdr>
      <w:top w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="B97034"
w:themeColor="accent6" w:themeShade="BF" />
      <w:bottom w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:right w:val="single" w:sz="24" w:space="4" w:color="C3D69B"
w:themeColor="accent3" w:themeTint="99" />
      <w:between w:val="single" w:sz="24" w:space="1" w:color="4F81BD"
w:themeColor="accent1" />
    </w:pBdr>
  </w:pPr>
  <w:r>
    <w:t>First paragraph.</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:pBdr>
      <w:top w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="B97034"
w:themeColor="accent6" w:themeShade="BF" />
      <w:bottom w:val="single" w:sz="24" w:space="0" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:right w:val="single" w:sz="24" w:space="4" w:color="C3D69B"
w:themeColor="accent3" w:themeTint="99" />
```

```
      <w:between w:val="single" w:sz="24" w:space="1" w:color="4F81BD"
w:themeColor="accent1" />
    </w:pBdr>
  </w:pPr>
  <w:r>
    <w:t>Second paragraph.</w:t>
  </w:r>
</w:p>
```

Since the bottom paragraph border is different between the two paragraphs (the bottom space value goes from 1 to 0), these paragraphs do not use the between border, and instead paragraph one uses its bottom border, and paragraph two uses its top border. If those values were identical, then paragraph one would have a between brder below it, and paragraph two would have no top border. *end example*]

Parent Elements
pBdr (§2.3.1.24)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or auto to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value auto, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is

Attributes	Description
	specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is `true`, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object.
	
	When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points.
	
	When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.
	
	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24/>`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the

Attributes	Description
	page offset shall be 24 points. *end example*] The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).

sz (Border Width)	Specifies the width of the current border.
	
	If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value.
	
	If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value.
	
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`
	
	The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	
	The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).

themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border.
	
	The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document.
	
	[*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup:
	
	`<w:top … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:bottom … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:left … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:right … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />

Attributes	Description
	The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*] The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).
themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255$$ $$= 102$$ $$= 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L' = 0.53 * 0.75$$ $$= 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634.

Attributes	Description
	This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255 = 153 = 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6)$$

Attributes	Description
	$= 0.71$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$ and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is single, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <attribute name="val" type="ST_Border" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
   <attribute name="space" type="ST_PointMeasure" use="optional"/>
   <attribute name="shadow" type="ST_OnOff" use="optional"/>
   <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.3.1.6 bidi (Right to Left Paragraph Layout)

This element specifies that this paragraph shall be presented using a right to left direction. This property only affects the set of paragraph-level properties, and shall not affect the layout of text within the contents of this paragraph.

[*Example*: Consider a paragraph with the bidi property set as follows:

```
<w:p>
    <w:pPr>
      <w:bidi/>
    </w:pPr>
    ...
</w:p>
```

This paragraph direction is now right to left, which means that all paragraph properties are displayed right to left (e.g. the paragraph marker glyph (if any) is displayed on the right, and indentation for the first line of the paragraph occurs on the right side of the page). *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.1.7 bottom (Paragraph Border Between Identical Paragraphs)

This element specifies the border which shall be displayed below a set of paragraphs which have the same set of paragraph border settings.

To determine if any two adjoining paragraphs shall have an individual top and bottom border or a between border, the set of borders on the two adjoining paragraphs are compared. If the border information on those

two paragraphs is identical for all possible paragraphs borders, then the between border is displayed. Otherwise, the final paragraph shall use its bottom border and the following paragraph shall use its top border, respectively. If this border specifies a space attribute, that value determines the space after the bottom of the text (ignoring any space below) which should be left before this border is drawn, specified in points.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then no between border shall be applied below identical paragraphs.

[*Example*: Consider the following two paragraphs' WordprocessingML definition:

```
<w:p>
  <w:pPr>
    <w:pBdr>
      <w:top w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="B97034"
w:themeColor="accent6" w:themeShade="BF" />
      <w:bottom w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:right w:val="single" w:sz="24" w:space="4" w:color="C3D69B"
w:themeColor="accent3" w:themeTint="99" />
      <w:between w:val="single" w:sz="24" w:space="1" w:color="4F81BD"
w:themeColor="accent1" />
    </w:pBdr>
  </w:pPr>
  <w:r>
    <w:t>First paragraph.</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:pBdr>
      <w:top w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="B97034"
w:themeColor="accent6" w:themeShade="BF" />
      <w:bottom w:val="single" w:sz="24" w:space="0" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:right w:val="single" w:sz="24" w:space="4" w:color="C3D69B"
w:themeColor="accent3" w:themeTint="99" />
      <w:between w:val="single" w:sz="24" w:space="1" w:color="4F81BD"
w:themeColor="accent1" />
    </w:pBdr>
```

```
    </w:pPr>
    <w:r>
      <w:t>Second paragraph.</w:t>
    </w:r>
  </w:p>
```

Since the paragraph border is different between the two paragraphs (the bottom space value goes from 1 to 0), paragraph one uses its bottom border, which is located one point below the text in that paragraph. *end example*]

Parent Elements
pBdr (§2.3.1.24)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or auto to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value auto, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is true, indicating that the border frame effect shall be applied. *end*

Attributes	Description
	example] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object.
	
	When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points.
	
	When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.
	
	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24/>`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*]
	
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).

sz (Border Width)	Specifies the width of the current border.

Attributes	Description

	If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value.
	
	If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value.
	
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`
	
	The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	
	The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).

themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border.
	
	The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document.
	
	[*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup:
	
	`<w:top … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:bottom … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:left … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:right … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	
	The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).
themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255 = 102 = 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L' = 0.53 * 0.75 = 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$ and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2"`

Attributes	Description
	w:themeShade="BF"/>

end example]

The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106). |
| themeTint (Border Theme Color Tint) | Specifies the tint value applied to the supplied theme color (if any) for this border instance.

If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border.

The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border.

[*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows:

$$T_{xml} = 0.6 * 255$$
$$= 153$$
$$= 99(hex)$$

The resulting themeTint value in the file format would be 99. *end example*]

Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows:

- Convert the color to the HSL color format (values from 0 to 1)
- Modify the luminance factor as follows:

$$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$

- Convert the resultant HSL color to RGB

[*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD.

The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$.

Applying the tint formula with a tint percentage of 60% to the luminance, we get:

$$L' = 0.53 * 0.6 + (1 - .6)$$
$$= 0.71$$

Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get 95B3D7. |

Attributes	Description
	This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
    <attribute name="val" type="ST_Border" use="required"/>
    <attribute name="color" type="ST_HexColor" use="optional"/>
    <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
    <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
    <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
    <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
    <attribute name="space" type="ST_PointMeasure" use="optional"/>
    <attribute name="shadow" type="ST_OnOff" use="optional"/>
    <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.3.1.8 cnfStyle (Paragraph Conditional Formatting)

This element specifies the set of conditional table style formatting properties which have been applied to this paragraph, if this paragraph is contained within a table cell. [*Note*: This property is an optimization which may be used by consumers to determine if a given property on a paragraph is the result of the table style properties vs. direct formatting on the paragraph itself. *end note*]

If this property is specified on a paragraph which is not contained within a table cell, then its contents shall be ignored when reading the contents of the document.

[*Example*: Consider a paragraph in the top right corner of a table with a table style applied. This paragraph would need to specify the following WordprocessingML:

```
<w:p>
  <w:pPr>
    <w:cnfStyle w:val="101000000100" />
    ...
  <//w:pPr>
  ...
</w:p>
```

This paragraph specifies that it has the conditional properties from the table style for the first column, first row, and the NW corner of the parent table by setting the appropriate bits in the val attribute. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description
val (Conditional Formatting Bit Mask)	Specifies the set of conditional formatting properties that have been applied to this object. These properties are expressed using a string serialization of a binary bitmask for each of the following properties (reading from the first character position right): • First Row - Is this the first row of the table? • Last Row - Is this the last row of the table? • First Column - Does this belong to the first column of the table? • Last Column - Does this belong to the last column of the table? • Band 1 Vertical - Does this belong to a column which should receive band 1 formatting? This property specifies whether the cell should receive the formatting specified for odd-numbered columns (e.g. 1,3,5,...) • Band 2 Vertical - Does this belong to a column which should receive band 2 formatting? This property specifies whether the cell should receive the formatting specified for even-numbered columns (e.g. 2,4,6...) • Band 1 Horizontal - Does this receive band 1 formatting? This property specifies whether the cell should receive the formatting specified for odd-numbered rows (e.g. 1,3,5,...) • Band 2 Horizontal - Does this receive band 2 formatting? This property specifies whether the cell should receive the formatting specified for even-numbered rows (e.g. 2,4,6...) • NE Cell - Is this part of the top-right corner of the table? • NW Cell - Is this part of the top-left corner of the table? • SE Cell - Is this part of the bottom-right corner of the table? • SW Cell - Is this part of the bottom-left corner of the table?

Attributes	Description

	For each of these properties, a value of 1 in the specified character position in the string means that the value is true, a value of 0 means false. All values must be specified.
	
	[*Example*: Consider a paragraph in the top right corner of a table with a table style applied. This paragraph would need to specify the following WordprocessingML:
	
	`<w:p>`
	`  <w:pPr>`
	`  <w:cnfStyle w:val="101000000100" />`
	`  ...`
	`  </w:pPr>`
	`  ...`
	`</w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the first column, first row, and the NW corner of the parent table by setting the appropriate bits in the val attribute. *end example*]
	
	The possible values for this attribute are defined by the ST_Cnf simple type (§2.18.11).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Cnf">
    <attribute name="val" type="ST_Cnf" use="required"/>
</complexType>
```

2.3.1.9 contextualSpacing (Ignore Spacing Above and Below When Using Identical Styles)

This element specifies that any space specified before or after this paragraph, specified using the spacing element (§2.3.1.33), should not be applied when the preceding and following paragraphs are of the same paragraph style, affecting the top and bottom spacing respectively. [*Example*: This value is typically used for paragraphs in lists, in which any space between subsequent list items, even if inherited from another style, is not desirable. *end example*]

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then spacing is not ignored. If it is present, then the spacing above or below on this paragraph is subtracted from the spacing which would have been present if contextual spacing was off, never going below zero.

[*Example*: Consider two paragraphs defined as follows:

```
<w:p>
  <w:pPr>
    <w:pStyle w:val="TestParagraphStyle" />
    <w:spacing w:after="200"/>
```

```
      <w:contextualSpacing/>
    </w:pPr>
    …
  </w:p>
  <w:p>
    <w:pPr>
      <w:pStyle w:val="TestParagraphStyle" />
      <w:spacing w:before="240"/>
    </w:pPr>
    …
  </w:p>
```

The first paragraph specifies a spacing after of 10 points, and the second paragraph specifies a spacing before of 12 points, therefore according to the rules on the spacing element, the net paragraph spacing should be 12 points. However, since the first paragraph specifies that its spacing should be omitted between paragraphs of the same style, and the two paragraphs use the same TestParagraphStyle, that value is subtracted from the total, therefore the paragraphs are spaced by 2 points. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.1.10 divId (Associated HTML div ID)

This element specifies that this paragraph should be located within the specified HTML `div` tag when this document is saved in HTML format. This ID is then used to look up the associated div stored in the divs (§2.15.2.8) element. [*Note*: This element is used to preserve the fidelity of existing HTML documents when saved in the WordprocessingML format. *end note*].

If the paragraph does not specify this element, then any `div` referenced by the previous paragraph is closed, and this paragraph shall not belong to any `div` when saved as HTML. If this specified id does not exist in the collection of `div`s the current document, then any `div` referenced by the previous paragraph is closed, and this paragraph shall not belong to any `div` when saved as HTML.

[*Example*: Consider the following WordprocessingML paragraph fragment:

```
<w:p>
  <w:pPr>
    <w:divId w:val="1512645511" />
  </w:pPr>
</w:p>
```

This paragraph specifies that it belongs to the HTML `div` with id `1512645511`, stored in the divs element. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.3.1.11 framePr (Text Frame Properties)

This element specifies information about the current paragraph with regard to *text frames*. *Text frames* are paragraphs of text in a document which are positioned in a separate region or frame in the document, and can be positioned with a specific size and position relative to non-frame paragraphs in the current document.

The first piece of information specified by the framePr element is that the current paragraph is actually part of a text frame in the document. This information is specified simply by the presence of the framePr element in paragraph's properties. If the framePr element is omitted, the paragraph shall not be part of any text frame in the document.

The second piece of information concerns the set of paragraphs which are part of the current text frame in the document. This is determined based on the attributes on the framePr element. If the set of attribute values specified on two adjacent paragraphs is identical, then those two paragraphs shall be considered to be part of the same text frame and rendered within the same frame in the document.

[*Example*: Consider a document in which the following two paragraphs are located adjacent to one another:

```
<w:p>
  <w:pPr>
    <w:framePr w:w="2191" w:h="811" w:hRule="exact" w:hSpace="180"
w:wrap="around" w:vAnchor="text" w:hAnchor="page" w:x="1921"/>
  </w:pPr>
  <w:r>
    <w:t>Paragraph One</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:framePr w:w="2191" w:h="810" w:hRule="exact" w:hSpace="180"
w:wrap="around" w:vAnchor="text" w:hAnchor="page" w:x="1921"/>
  </w:pPr>
  <w:r>
    <w:t>Paragraph Two.</w:t>
  </w:r>
</w:p>
```

These two paragraphs, although each is a part of a text frame due to the presence of the framePr element, are different text frames because of the differing h value - 810 vs. 811. *end example*]

The positioning of the frame relative to the properties stored on its attribute values shall be calculated relative to the next paragraphs in the document which is itself not part of a text frame.

[*Example*: Consider a document in which the following three paragraphs are located adjacent to one another:

```
<w:p>
  <w:pPr>
    <w:framePr w:w="2191" w:h="811" w:hRule="exact" w:hSpace="180"
w:wrap="around" w:vAnchor="text" w:hAnchor="page" w:x="1921"/>
  </w:pPr>
  <w:r>
    <w:t>Paragraph One</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:framePr w:w="2191" w:h="811" w:hRule="exact" w:hSpace="180"
w:wrap="around" w:vAnchor="text" w:hAnchor="page" w:x="1921"/>
  </w:pPr>
  <w:r>
    <w:t>Paragraph Two.</w:t>
  </w:r>
</w:p>
<w:p/>
```

The first two paragraphs form a single text frame, which is anchored using its attribute values relative to the first non-frame paragraph following it (the third paragraph in the example). *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description
anchorLock (Lock Frame Anchor to Paragraph)	Specifies that the frame shall always remain in the same logical position relative to the non-frame paragraphs which precede and follow it in this document. This means that consumers which modify this document shall ensure that this text frame remains directly above the non-frame paragraph which it is currently above, by adjusting the frame's positioning properties as needed as the paragraph is moved throughout the document rather than moving the frame's logical location within the paragraphs in the document, if that would be more appropriate. If this attribute is omitted, then this frame shall not have a locked anchor position. [*Example*: Consider the following WordprocessingML paragraph contained in a text

Attributes	Description

	frame:
	
	`<w:p>`
	`  <w:pPr>`
	`  <w:framePr w:w="2419" w:h="2189" w:hRule="exact"`
	`w:hSpace="187" w:wrap="around" w:vAnchor="text"`
	`w:hAnchor="page" w:x="1643" w:y="73" w:anchorLock="1"/>`
	`  </w:pPr>`
	`  <w:r>`
	`  <w:t>Text Frame Content.</w:t>`
	`  </w:r>`
	`</w:p>`
	
	This text frame has a locked anchor using the anchorLock attribute. If the text frame is moved down in the document, the text frame properties must be adjusted to be relative to the parent paragraph's same logical position - the paragraph cannot be relocated in the document, which results in changes to the frame's properties as follows:
	
	`<w:p>`
	`  <w:pPr>`
	`  <w:framePr w:w="2419" w:h="2189" w:hRule="exact"`
	`w:hSpace="187" w:wrap="around" w:vAnchor="text"`
	`w:hAnchor="page" w:x="1643" w:y="-5247" w:anchorLock="1"/>`
	`  </w:pPr>`
	`  <w:r>`
	`  <w:t>Text Frame Content.</w:t>`
	`  </w:r>`
	`</w:p>`
	
	The non-frame paragraph was relocated 5320 twentieths of a point below its original location in the document, and the frame's vertical positioning properties were adjusted to ensure its logical location within the paragraph ordering was constant while its visual location was changed. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

dropCap (Drop Cap Frame)	Specifies that the current frame contains a drop cap to be located at the beginning of the next non-frame paragraph in the document. Its contents shall be used to specify how that drop cap should be positioned relative to that paragraph. If this attribute is omitted, then this frame shall not be considered a drop cap frame. [*Note*: Although a drop cap is simply a text frame, this element is used to determine how the cap should be positioned relative to the following non-frame paragraph in relative terms (see possible values), rather than relying on absolute sizing. *end note*] [*Example*: Consider the following paragraph containing a text frame which should be

Attributes	Description

	positioned as a drop cap:
	
	`<w:p>`
	`  <w:pPr>`
	`  <w:framePr w:dropCap="margin" w:lines="3" w:hSpace="432" w:wrap="around" w:vAnchor="text" w:hAnchor="page" />`
	`  </w:pPr>`
	`  <w:r>`
	`  <w:t>A</w:t>`
	`  </w:r>`
	`</w:p>`
	
	The dropCap attribute specifies a value of `margin`, so this drop cap will be placed outside of the text margin before the start of the current text. *end example*]
	
	The possible values for this attribute are defined by the ST_DropCap simple type (§2.18.24).

h (Frame Height)	Specifies the frame's height.
	
	This height is expressed in twentieths of a point.
	
	If this attribute is omitted, then its value shall be assumed to be `0`.
	
	The meaning of the value of the h attribute is defined based on the value of the hRule attribute for this text frame as follows:
	• If the value of hRule is `auto`, then the frame's height should be automatically determined based on the height of its contents. This value is ignored.
	• If the value of hRule is `atLeast`, then the frame's height should be at least the value of this attribute.
	• If the value of hRule is `exact`, then the frame's height should be exactly the value of this attribute.
	
	[*Example*: Consider the following paragraph containing a text frame:
	
	`<w:p>`
	`  <w:pPr>`
	`  <w:framePr w:w="2419" w:h="2189" w:hRule="atLeast" w:hspace="187" w:wrap="around" w:vanchor="text" w:hanchor="page" w:x="1643" w:y="73" />`
	`  </w:pPr>`
	`  <w:r>`
	`  <w:t>Text Frame Content.</w:t>`
	`  </w:r>`
	`</w:p>`
	
	The h attribute specifies a value of 2189 twentieths of a point, so this text frame will be a

Attributes	Description
	minimum of 2189 twentieths of a point high regardless of its contents, since its hRule value is set to atLeast. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

hAnchor (Frame Horizontal Positioning Base)	Specifies the base object from which the horizontal positioning in the x attribute should be calculated.
	
	A text frame may be horizontally positioned relative to:
	• The vertical edge of the page before any runs of text (the left edge for left-to-right paragraphs, the right edge for right-to-left paragraphs)
	• The vertical edge of the text margin before any runs of text (the left edge for left-to-right paragraphs, the right edge for right-to-left paragraphs)
	• The vertical edge of the text margin for the column in which the anchor paragraph is located
	
	If this attribute is omitted, then its value shall be assumed to be page.
	
	[*Example*: Consider a text frame which should be positioned one inch to the right of its column in a left-to-right document. This text frame would be specified using the following WordprocessingML:
	
	`<w:pPr>`
	`  <w:framePr … w:x="1440" w:hAnchor="column" />`
	`</w:pPr>`
	
	These frame properties specify that they are relative to the anchor paragraph's column, and that relative to that column, the frame should be 1440 twentieths of a point in the direction of the flow of text (right, in this case). *end example*]
	
	The possible values for this attribute are defined by the ST_HAnchor simple type (§2.18.40).

hRule (Frame Height Type)	Specifies the meaning of the height specified for this frame. The meaning of the value of the h attribute is defined based on the value of the hRule attribute for this text frame as follows: • If the value of hRule is auto, then the frame's height should be automatically determined based on the height of its contents. The h value is ignored. • If the value of hRule is atLeast, then the frame's height should be at least the value the h attribute. • If the value of hRule is exact, then the frame's height should be exactly the value of the h attribute. If this attribute is omitted, then its value shall be assumed to be auto.

Attributes	Description

	[*Example*: Consider the following paragraph containing a text frame:
	
	`<w:p>`
	`  <w:pPr>`
	`  <w:framePr w:w="2419" w:h="2189" w:hRule="atLeast"`
	`w:hSpace="187" w:wrap="around" w:vAnchor="text"`
	`w:hAnchor="page" w:x="1643" w:y="73" />`
	`  </w:pPr>`
	`  <w:r>`
	`  <w:t>Text Frame Content.</w:t>`
	`  </w:r>`
	`</w:p>`
	
	The h attribute specifies a value of 2189 twentieths of a point, so this text frame will be a minimum of 2189 twentieths of a point high regardless of its contents, since its hRule value is set to atLeast. *end example*]
	
	The possible values for this attribute are defined by the ST_HeightRule simple type (§2.18.42).

hSpace (Horizontal Frame Padding)	Specifies the minimum distance which shall be maintained between the current text frame and any non-frame text which has been allowed to flow around this object when the wrap attribute on this text frame is set to around.
	
	This distance is expressed in twentieths of a point.
	
	If the wrap value is not set to around, this value shall be ignored. If this attribute is omitted, its value shall be assumed to be 0.
	
	[*Example*: Consider a text frame which should have a minimum of a one-half inch spacing from any non-frame text on its left and right sides. This constraint would be specified using the following WordprocessingML:
	
	`<w:pPr>`
	`  <w:framePr ...w:hSpace="720" w:wrap="around" />`
	`</w:pPr>`
	
	The wrap value of around allows text to wrap around this text frame, and the hSpace attribute specifies that the spacing between text and this frame shall be a minimum of 720 twentieths of a point. *end example*]
	
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

lines (Drop Cap Vertical Height in Lines)	Specifies the number of lines in the non-frame paragraph to which this text frame is anchored which should be used to calculate the drop cap's height. If the current frame is not a drop cap (the parent framePr element does not have the

Attributes	Description

	dropCap attribute), this value is ignored. If the current text frame is a dropped cap and this attribute is present, then any other vertical positioning information shall be ignored.
	
	If this attribute is omitted, then its value shall be considered to be 1.
	
	[*Example*: Consider the following paragraph containing a text frame which should be positioned as a drop cap:
	
	`<w:p>`
	`  <w:pPr>`
	`  <w:framePr w:dropCap="margin" w:lines="3" w:hSpace="432" w:wrap="around" w:vAnchor="text" w:hAnchor="page" W:y="400" w:yAlign="text" />`
	`  </w:pPr>`
	`  <w:r>`
	`  <w:t>O</w:t>`
	`  </w:r>`
	`</w:p>`
	
	Since this frame is being used as a dropped cap, the y and yAlign attributes are ignored and the height of the drop cap is the first three lines of the anchor paragraph. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

vAnchor (Frame Vertical Positioning Base)	Specifies the base object from which the horizontal positioning in the y attribute should be calculated.
	
	A text frame may be horizontally positioned relative to:
	• The horizontal edge of the page before any runs of text (the top edge for top-to-bottom sections, the bottom for bottom-to-top sections)
	• The horizontal edge of the text margin before any runs of text (the top edge for top-to-bottom sections, the bottom for bottom-to-top sections)
	• The horizontal edge of the page before any runs of text (the top edge for top-to-bottom sections, the bottom for bottom-to-top sections)
	
	If this attribute is omitted, then its value shall be assumed to be page.
	
	[*Example*: Consider a text frame which should be positioned two inches below the page top in a top-to-bottom document. This text frame would be specified using the following WordprocessingML:
	
	`<w:pPr>`
	`  <w:framePr … w:y="2880" w:vAnchor="page" />`
	`</w:pPr>`

Attributes	Description
	These frame properties specify that they are relative to the anchor page, and that relative to that column, the frame should be 2880 twentieths of a point in the direction of the flow of text (down, in this case). *end example*] The possible values for this attribute are defined by the ST_VAnchor simple type (§2.18.109).

vSpace (Vertical Frame Padding)	Specifies the minimum distance which shall be maintained between the current text frame and any non-frame text which is above or below this text frame.
	
	This distance is expressed in twentieths of a point.
	
	If this attribute is omitted, its value shall be assumed to be 0.
	
	[*Example*: Consider a text frame which should have a minimum of a one-half inch spacing from any non-frame text on its top and bottom sides. This constraint would be specified using the following WordprocessingML:
	
	`<w:pPr>`
	`  <w:framePr … w:vSpace="720" />`
	`</w:pPr>`
	
	The vspace attribute specifies that the spacing between text and this frame shall be a minimum of 720 twentieths of a point. *end example*]
	
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

w (Frame Width)	Specifies the exact value for this text frame's width.
	
	This value is specified in twentieths of a point.
	
	When this attribute is present, the text frame shall be rendered to the exact width specified. If this attribute is omitted, the text frame width shall be automatically determined by the maximum line width of the content within the text frame.
	
	[*Example*: Consider the following WordprocessingML fragment specifying a text frame:
	
	`<w:p>`
	`  <w:pPr>`
	`  <w:framePr w:w="2419" w:h="2189" w:hRule="atLeast"`
	`w:hspace="187" w:wrap="around" w:vanchor="text"`
	`w:hanchor="page" w:x="1643" w:y="73" />`
	`  </w:pPr>`
	`  <w:r>`
	`  <w:t>Text Frame Content.</w:t>`
	`  </w:r>`
	`</w:p>`

Attributes	Description
	This text frame specifies that its width shall be exactly 2419 twips. If this attribute was removed, the text frame would be rendered at the width of the content `Text Frame Content`. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

wrap (Text Wrapping Around Frame)	Specifies the type of text wrapping which should be allowed around the contents of this text frame. This attribute determines if non-frame text shall be allowed to flow around the contents of this frame.
	
	If this attribute is omitted, its value shall be assumed to be `around`.
	
	[*Example*: Consider the following WordprocessingML fragment specifying a text frame:
	
	`<w:p>`
	`  <w:pPr>`
	`  <w:framePr w:w="2419" w:h="2189" w:hRule="atLeast" w:hSpace="187" w:wrap="around" w:vAnchor="text" w:hAnchor="page" w:x="1643" w:y="73" />`
	`  </w:pPr>`
	`  <w:r>`
	`  <w:t>Text Frame Content.</w:t>`
	`  </w:r>`
	`</w:p>`
	
	This text frame specifies that when the frame is rendered on the page, any non-text frame paragraphs which would normally flow onto the same lines shall be allowed to do so. *end example*]
	
	The possible values for this attribute are defined by the ST_Wrap simple type (§2.18.113).

x (Absolute Horizontal Position)	Specifies an absolute horizontal position for the text frame. This absolute position is specified relative to the horizontal anchor specified by the hAnchor attribute for this text frame. This value is expressed in twentieths of a point. If it is positive, then the text frame is positioned after the anchor object in the direction of horizontal text flow in this document. If it is negative, then the text frame is positioned before the anchor object in the direction of horizontal text flow in this document. If the xAlign attribute is also specified, then this value is ignored. If this attribute is omitted, then its value shall be assumed to be `0`. [*Example*: Consider the following WordprocessingML fragment specifying a text frame:

Attributes	Description
	<w:p> <w:pPr> <w:framePr w:w="2419" w:h="2189" w:hRule="atLeast" w:hSpace="187" w:wrap="around" w:vAnchor="text" w:hAnchor="page" w:x="1643" w:y="73" /> </w:pPr> <w:r> <w:t>Text Frame Content.</w:t> </w:r> </w:p> This text frame specifies that it should be located exactly 1643 twentieths of a point after the vertical edge of the page (from the hAnchor attribute). *end example*] The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§2.18.88).
xAlign (Relative Horizontal Position)	Specifies a relative horizontal position for the text frame. This relative position is specified relative to the horizontal anchor specified by the hAnchor attribute for this text frame. If omitted, this attribute is not specified and the value of the x attribute determines the absolute horizontal position of the text frame. If specified, the position for this attribute supersede any value which is specified in the x attribute, and that value is ignored. [*Example*: Consider the following WordprocessingML fragment specifying a text frame: <w:p> <w:pPr> <w:framePr w:w="2419" w:h="2189" w:hRule="atLeast" w:hSpace="187" w:wrap="around" w:vAnchor="text" w:hAnchor="page" w:x="1643" w:xAlign="left" w:y="73" /> </w:pPr> <w:r> <w:t>Text Frame Content.</w:t> </w:r> </w:p> This text frame specifies that it has a horizontal placement of exactly 1643 twentieths of a point relative to the page, but that exact placement is overridden by the presence of the xAlign attribute to place the frame on the left side of the page. *end example*] The possible values for this attribute are defined by the ST_XAlign simple type (§2.18.114).
y (Absolute Vertical Position)	Specifies an absolute vertical position for the text frame. This absolute position is specified relative to the vertical anchor specified by the vAnchor attribute for this text frame.

Attributes	Description

	This value is expressed in twentieths of a point. If it is positive, then the text frame is positioned after the anchor object in the direction of vertical text flow in this document. If it is negative, then the text frame is positioned before the anchor object in the direction of vertical text flow in this document.
	
	If the yAlign attribute is also specified, then this value is ignored. If this attribute is omitted, then its value shall be assumed to be 0.
	
	[*Example*: Consider the following WordprocessingML fragment specifying a text frame:
	
	`<w:p>`
	`  <w:pPr>`
	`  <w:framePr w:w="2419" w:h="2189" w:hRule="atLeast"`
	`w:hSpace="187" w:wrap="around" w:vAnchor="text"`
	`w:hAnchor="page" w:x="1643" w:y="73" />`
	`  </w:pPr>`
	`  <w:r>`
	`  <w:t>Text Frame Content.</w:t>`
	`  </w:r>`
	`</w:p>`
	
	This text frame specifies that it should be located exactly 79 twentieths of a point below the top vertical edge of the anchor's paragraph's text (from the vAnchor attribute), assuming that the vertical text direction is top to bottom. *end example*]
	
	The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§2.18.88).

yAlign (Relative Vertical Position)	Specifies a relative vertical position for the text frame. This relative position is specified relative to the vertical anchor specified by the vAnchor attribute for this text frame.
	
	If omitted, this attribute is not specified and the value of the y attribute determines the absolute horizontal position of the text frame. If specified, the position for this attribute supersedes any value which is specified in the y attribute, and that value is ignored, unless the vAnchor is set to text, in which case any relative positioning is not allowed, and is itself ignored.
	
	[*Example*: Consider the following WordprocessingML fragment specifying a text frame:
	
	`<w:p>`
	`  <w:pPr>`
	`  <w:framePr w:w="2419" w:h="2189" w:hRule="atLeast"`
	`w:hSpace="187" w:wrap="around" w:vAnchor="margin"`
	`w:hAnchor="page" w:x="1643" w:y="73" w:yAlign="center" />`
	`  </w:pPr>`
	`  <w:r>`

Attributes	Description
	<w:t>Text Frame Content.</w:t> </w:r> </w:p> This text frame specifies that it has a vertical placement of exactly 73 twentieths of a point relative to the top margin, but that exact placement is overridden by the presence of the yAlign attribute to place the frame in the center of the margin. *end example*] The possible values for this attribute are defined by the ST_YAlign simple type (§2.18.115).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FramePr">
   <attribute name="dropCap" type="ST_DropCap" use="optional"/>
   <attribute name="lines" type="ST_DecimalNumber" use="optional"/>
   <attribute name="w" type="ST_TwipsMeasure" use="optional"/>
   <attribute name="h" type="ST_TwipsMeasure" use="optional"/>
   <attribute name="vSpace" type="ST_TwipsMeasure" use="optional"/>
   <attribute name="hSpace" type="ST_TwipsMeasure" use="optional"/>
   <attribute name="wrap" type="ST_Wrap" use="optional"/>
   <attribute name="hAnchor" type="ST_HAnchor" use="optional"/>
   <attribute name="vAnchor" type="ST_VAnchor" use="optional"/>
   <attribute name="x" type="ST_SignedTwipsMeasure" use="optional"/>
   <attribute name="xAlign" type="ST_XAlign" use="optional"/>
   <attribute name="y" type="ST_SignedTwipsMeasure" use="optional"/>
   <attribute name="yAlign" type="ST_YAlign" use="optional"/>
   <attribute name="hRule" type="ST_HeightRule" use="optional"/>
   <attribute name="anchorLock" type="ST_OnOff" use="optional"/>
</complexType>
```

2.3.1.12 ind (Paragraph Indentation)

This element specifies the set of indentation properties applied to the current paragraph.

Indentation settings are overriden on an individual basis - if any single attribute on this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If any single attribute on this element is never specified in the style hierarchy, then no indentation of that type is applied to the paragraph.

[*Example*: Consider a paragraph which should have a one inch indentation from the text margins on both the left and the right sides, except for the first line in each paragraph, which should only be indented one quarter of an inch from the text margin (on the side which begins the flow of text for this paragraph). This set of indentations is specified using the following WordprocessingML:

```
<w:pPr>
  <w:ind w:left="1440" w:right="1440" w:hanging="1080" />
</w:pPr>
```

This set of indentation properties specifies that a 1440 twentieths of a point indentation should be provided on both the left and the right side of the text margins for this paragraph, and that a 1080 twentieths of a point hanging indent (towards the text margin) should be applied to the text in the first paragraph, giving it a net one-quarter inch indent from the text margin. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description

firstLine (Additional First Line Indentation)	Specifies the additional indentation which shall be applied to the first line of the parent paragraph. This additional indentation is specified relative to the paragraph indentation which is specified for all other lines in the parent paragraph.
	
	The firstLine and hanging attributes are mutually exclusive, if both are specified, then the firstLine value is ignored. If the firstLineChars attribute is also specified, then this value is ignored. If this attribute is omitted, then its value shall be assumed to be zero (if needed).
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:ind w:left="1440" w:right="720" w:firstLine="1440" />`
	`</w:pPr>`
	
	This set of indentations specifies that the first line should be indented 1440 twentieths of a point (one inch) from the indentation specified for all remaining paragraphs, which is the 1440 twentieths of a point, as specified by the left attribute. This gives the first line a two inch indentation from the text margin. *end example*]
	
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

firstLineChars (Additional First Line Indentation in Character Units)	Specifies the additional indentation which shall be applied to the first line of the parent paragraph. This additional indentation is specified relative to the paragraph indentation which is specified for all other lines in the parent paragraph. It is specified in one hundredths of a character unit. The firstLineChars and hangingChars attributes are mutually exclusive, if both are specified, then the firstLineChars value is ignored. If the firstLine attribute is also specified, then this value supersedes its other value. If this attribute is omitted, then its value shall be assumed to be zero (if needed). [*Example*: Consider the following WordprocessingML fragment: `<w:pPr>`

Attributes	Description
	<w:ind w:left="1440" w:right="720" w:firstLineChars="140" /> </w:pPr> This set of indentations specifies that the first line should be indented 140 hundredths of a character units from the indentation specified for all remaining paragraphs, which is the 1440 twentieths of a point specified by the left attribute. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
hanging (Indentation Removed from First Line)	Specifies the indentation which shall be removed from the first line of the parent paragraph, by moving the indentation on the first line back towards the beginning of the direction of text flow. This indentation is specified relative to the paragraph indentation which is specified for all other lines in the parent paragraph. The firstLine and hanging attributes are mutually exclusive, if both are specified, then the firstLine value is ignored. If the hangingChars attribute is also specified, then this value is ignored. If this attribute is omitted, its value shall be assumed to be zero (if needed). [*Example*: Consider the following WordprocessingML fragment: <w:pPr> <w:ind w:left="1440" w:right="720" w:hanging="720" /> </w:pPr> This set of indentations specifies that the first line should be indented 720 twentieths of a point (one inch) towards the text margin from the indentation specified for all remaining paragraphs, which is the 1440 twentieths of a point specified by the left attribute. This gives the first line a one-half inch indentation from the text margin. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).
hangingChars (Indentation Removed From First Line in Character Units)	Specifies the indentation which shall be removed from the first line of the parent paragraph, by moving the indentation on the first line back towards the beginning of the direction of text flow. This indentation is specified relative to the paragraph indentation which is specified for all other lines in the parent paragraph. It is specified in one hundredths of a character unit. The firstLineChars and hangingChars attributes are mutually exclusive, if both are specified, then the firstLine value is ignored. If the hanging attribute is also specified,

Attributes	Description

	then its value is superseded by this value. If this attribute is omitted, its value shall be assumed to be zero (if needed).
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:ind w:left="1440" w:right="720" w:hangingChars="100" />`
	`</w:pPr>`
	
	This set of indentations specifies that the first line should be indented one character unit towards the text margin from the indentation specified for all remaining paragraphs, which is the 1440 twentieths of a point specified by the left attribute. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

left (Left Indentation)	Specifies the indentation which shall be placed between the left text margin for this paragraph and the left edge of that paragraph's content in a left to right paragraph, and the right text margin and the right edge of that paragraph's text in a right to left paragraph. If the mirrorIndents property (§2.3.1.18) is specified for this paragraph, then this indent is used for the inside page edge - the right page edge for odd numbered pages and the left page edge for even numbered pages.
	
	If this attribute is omitted, its value shall be assumed to be zero.
	
	All other values for this element are relative to the text margin, Negative values are defined such that the text is moved past the text margin, positive values move the text inside the text margin. This value may be superseded for the first line only via use of the firstLine or hanging attributes. As well, if the leftChars attribute is specified, then this value is ignored.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:ind w:left="720" w:right="2880" />`
	`</w:pPr>`
	
	This set of paragraph indentations specifies that this paragraph's text should be indented 720 twentieths of a point (one half inch) from the left text margin in this document, assuming this is a left to right paragraph. *end example*]
	
	The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§2.18.88).

leftChars (Left Indentation in Character Units)	Specifies the indentation which shall be placed between the left text margin for this paragraph and the left edge of that paragraph's content in a left to right paragraph, and the right text margin and the right edge of that paragraph's text in a right to left paragraph. If the mirrorIndents property (§2.3.1.18) is specified for this paragraph, then

Attributes	Description

	this indent is used for the inside page edge - the right page edge for odd numbered pages and the left page edge for even numbered pages.
	
	This value is specified in hundredths of a character unit.
	
	If this attribute is omitted, its value shall be assumed to be zero.
	
	All other values for this element are relative to the text margin, Negative values are defined such that the text is moved past the text margin, positive values move the text inside the text margin. This value may be superseded for the first line only via use of the firstLine or hanging attributes. As well, if the left attribute is specified, then its value is ignored, and is superseded by this value.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:ind w:leftChars="250" />`
	`</w:pPr>`
	
	This set of paragraph indentations specifies that this paragraph's text should be indented two and a half character units from the left text margin in this document. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

right (Right Indentation)	Specifies the indentation which shall be placed between the right text margin for this paragraph and the right edge of that paragraph's content in a left to right paragraph, and the left text margin and the left edge of that paragraph's text in a right to left paragraph. If the mirrorIndents property (§2.3.1.18) is specified for this paragraph, then this indent is used for the outside page edge - the left page edge for odd numbered pages and the right page edge for even numbered pages.
	
	If this attribute is omitted, its value shall be assumed to be zero.
	
	All other values for this element are relative to the text margin, Negative values are defined such that the text is moved past the text margin, positive values move the text inside the text margin. As well, if the rightChars attribute is specified, then this value is ignored.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:ind w:left="720" w:right="-1440" />`
	`</w:pPr>`
	
	This set of paragraph indentations specifies that this paragraph's text should be indented 1440 twentieths of a point (one inch) into the right text margin in this document,

Attributes	Description
	assuming this is a left to right paragraph. *end example*] The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§2.18.88).

rightChars (Right Indentation in Character Units)	Specifies the indentation which shall be placed between the right text margin for this paragraph and the right edge of that paragraph's content in a left to right paragraph, and the left text margin and the left edge of that paragraph's text in a right to left paragraph. If the mirrorIndents property (§2.3.1.18) is specified for this paragraph, then this indent is used for the outside page edge - the left page edge for odd numbered pages and the right page edge for even numbered pages.
	
	This value is specified in hundredths of a character unit.
	
	If this attribute is omitted, its value shall be assumed to be zero.
	
	All other values for this element are relative to the right text margin, Negative values are defined such that the text is moved past the text margin, positive values move the text inside the text margin. As well, if the right attribute is specified, then its value is ignored, and is superseded by this value.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:ind w:rightChars="250" />`
	`</w:pPr>`
	
	This set of paragraph indentations specifies that this paragraph's text should be indented two and a half character units from the right text margin in this document, assuming this is a left to right paragraph. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Ind">
   <attribute name="left" type="ST_SignedTwipsMeasure" use="optional"/>
   <attribute name="leftChars" type="ST_DecimalNumber" use="optional"/>
   <attribute name="right" type="ST_SignedTwipsMeasure" use="optional"/>
   <attribute name="rightChars" type="ST_DecimalNumber" use="optional"/>
   <attribute name="hanging" type="ST_TwipsMeasure" use="optional"/>
   <attribute name="hangingChars" type="ST_DecimalNumber" use="optional"/>
   <attribute name="firstLine" type="ST_TwipsMeasure" use="optional"/>
   <attribute name="firstLineChars" type="ST_DecimalNumber" use="optional"/>
</complexType>
```

2.3.1.13 jc (Paragraph Alignment)

This element specifies the paragraph alignment which shall be applied to text in this paragraph.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then no alignment is applied to the paragraph.

[*Example*: Consider a paragraph which should be right justified to the right page side paragraph extents within a document. This constraint is specified in the following WordprocessingML content:

```
<w:pPr>
  <w:jc w:val="right" />
</w:pPr>
```

The paragraph is now `right` justified on the page. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description

val (Alignment Type)	Specifies the justification which should be applied to the parent object within a document.
	
	The possible values (see below) for this attribute are always specified relative to the page, and do not change semantic from right-to-left and left-to-right documents.
	
	[*Example*: Consider the following WordprocessingML fragment for a paragraph in a document:
	
	`<w:pPr>`
	`  <w:jc w:val="right" />`
	`</w:pPr>`
	
	This paragraph is now right justified on the page, regardless of the paragraph or section settings. *end example*]
	
	The possible values for this attribute are defined by the ST_Jc simple type (§2.18.50).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Jc">
   <attribute name="val" type="ST_Jc" use="required"/>
</complexType>
```

2.3.1.14 keepLines (Keep All Lines On One Page)

This element specifies that when rendering this document in a page view, all lines for this page are maintained on a single page whenever possible.

This means that if the contents of the current paragraph would normally span across two pages due to the placement of the paragraph's text, all lines in this paragraph shall be moved onto the next page to ensure they are displayed together. If this is not possible because all lines in the paragraph would exceed a single page in any case, then lines in this paragraph shall start on a new page, with page breaks as needed afterwards.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then this property shall not be applied.

[*Example*: Consider a WordprocessingML document in which a code fragment (such as the schema fragments in this document) are defined such that they should never be broken across a page boundary in order to improve readability. This constraint would be specified using the following paragraph properties in WordprocessingML:

```
<w:pPr>
  <w:keepLines />
  ...
</w:pPr>
```

This setting ensures that the schema fragment will be displayed on one page if possible. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

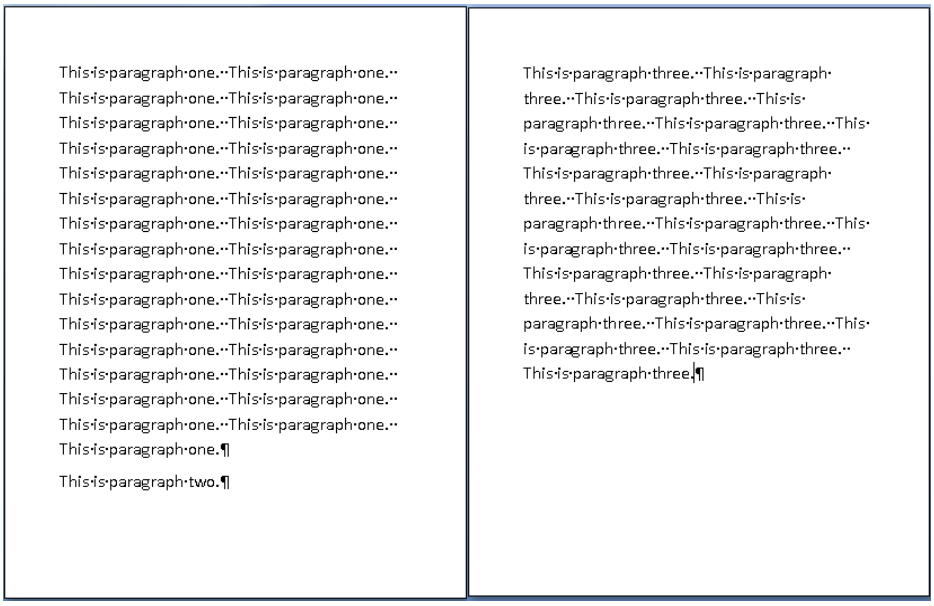
2.3.1.15 keepNext (Keep Paragraph With Next Paragraph)

This element specifies that when rendering this document in a paginated view, the contents of this paragraph are at least partly rendered on the same page as the following paragraph whenever possible.

This means that if the contents of the current paragraph would normally be completely rendered on a different page than the following paragraph (because only one of the two paragraphs would fit on the remaining space on the first page), then both paragraphs shall be rendered on a single page. This property can be chained between multiple paragraphs to ensure that all paragraphs are rendered on a single page without any intervening page boundaries. If this is not possible the entire set of paragraphs that are grouped together using this property would exceed a single page in any case, then the set of "keep with next" paragraphs shall start on a new page, with page breaks as needed afterwards.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then this property shall not be applied.

[*Example*: Consider the following document with three paragraphs:

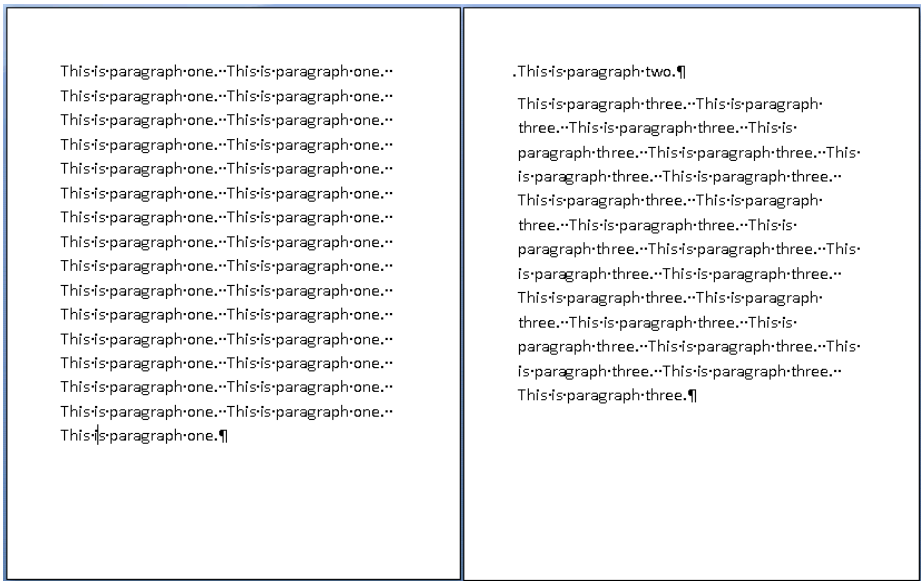


As shown above, the second paragraph and third paragraph are being rendered on two separate pages. However, a producer can specify that the second paragraph should not be displayed without any part of the third by setting the keepNext element as follows:

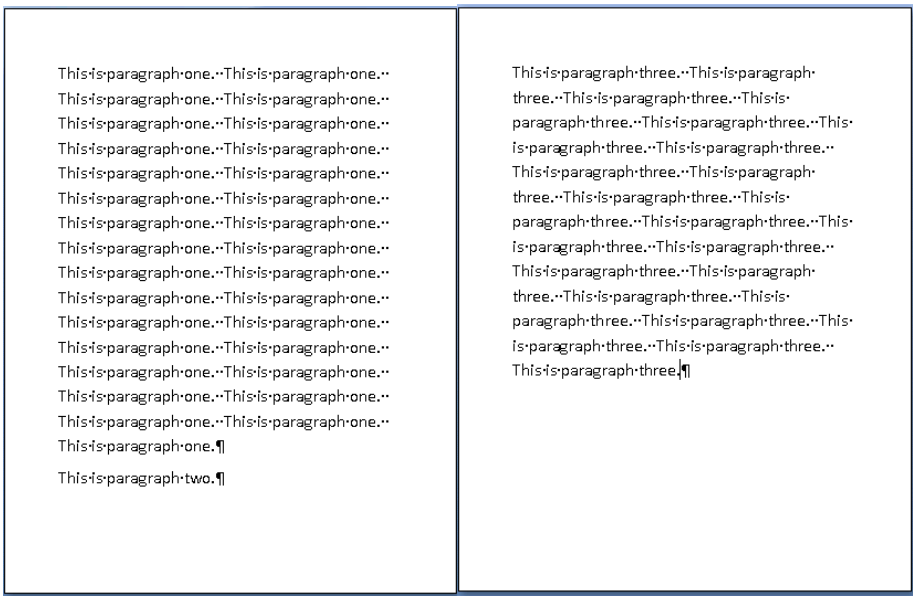
```
<w:pPr>
  <w:keepNext/>
```

```
    ...
  </w:pPr>
```

This would ensure that the second paragraph is displayed on the same page as the third paragraph:



However, if the first paragraph was also set to keepNext, then this set of grouped paragraphs would exceed a page, therefore the set of paragraphs would be grouped and start on the first page, resulting in the second paragraph reappearing on page one:



Since the paragraphs cannot all be put on one page, they all start on page one and flow as needed. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.1.16 kinsoku (Use East Asian Typography Rules for First and Last Character per Line)

This element specifies whether East Asian typography and line-breaking rules shall be applied to text in this paragraph to determine which characters may begin and end each line. This property only applies to Chinese PRC, Chinese Taiwan, and Japanese text in this paragraph.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then this property shall be applied to Chinese PRC, Chinese Taiwan, and Japanese text in this paragraph.

If these rules are set on the current paragraph, then the following rules are applied to the all first and last characters in the paragraph except the first and last character in the paragraph. By default, the following settings are used for kinsoku paragraphs:

Chinese (Simplified)

Annex A. Cannot start a line:

!%),.:;>?]}¢¨° • ˇ ¯ — ‖ ’ ” … ‰ ′ ″ › ℃ ： 、 。 〃 〉 》 」 』 】 〕 〗 ″ ﹀ ﹂ ﹀ ﹀ ﹏ ） ｝ 〕 ！ " % ′ ） ， ． ： ； ？ ］ ‘ | } ～ ¢

Annex B. Cannot end a line:

$([{£¥·‘“〈《「『【〔〖〝｛（〈$.[{

Chinese (Traditional)

Annex C. Cannot start a line:

!),.:;?]}¢• －—

’ ” •‥…… ′ － 、 。〉 》 」 』 】 ） ″ ： || ｜ ﹀﹀﹀﹀﹀﹀﹀﹏ ，、 ． ； ： ？

！ ） ｝ ） ！ ） ， ． ： ； ？ ］ ｜ ｝ 、

Annex D. Cannot end a line:

([{£¥ ‘ “ 〝 〈 《 「 『 【 〔 〝﹀﹀﹀﹀﹀﹀﹀﹀（ ｛ （ （ ｛

Japanese

Annex E. Cannot start a line:

!%),.:;?]}¢° ’ ” ‰′ ″ ℃、。々〉 》 」 』 】 〕 ゛ ゜ヽゝ・ ヽゝ！ ％），． ： ；

？ ］ ｝ 。」、・ﾞﾟ ₡

Annex F. Cannot end a line:

$([\{£¥ ‘ “ 〈 《 「 『 【 〔 $ （ ［ {「₤￥

Korean

Annex G. Cannot start a line:

!%),.:;?]}¢° ’ ” ′ ″ ℃〉 》 」 』 】 〕 ！ ％），． ： ； ？ ］ } ₡

Annex H. Cannot end a line:

$([\{£¥ ‘ “ 〈 《 「 『 【 〔 $ （ ［ {₤￥₩

If the strictFirstAndLastChars property (§2.15.1.83) is set in the Document Settings part, then the following settings supersede the defaults for Japanese:

Annex I. Cannot start a line:

!%),.:;?]}¢° ’ ” ‰′ ″ ℃、。々〉 》 」 』 】 〕 ぁ ぃ ぅ ぇ ぉ っ ゃ ゅ ょ ゎ ゛ ゜ヽゝ

ァ ィ ゥ ェ ォ ッ ャ ュ ョ ヮ カ ケ・ ー ヽゝ！ ％），． ： ； ？ ］ } 。」、・ｱｲｳｴｵﾔﾕﾖﾂｰﾞﾟ ₡

Annex J. Cannot end a line:

$([\{£¥ ‘ “ 〈 《 「 『 【 〔 $ （ ［ {「₤￥

If the noLineBreaksBefore property (§2.15.1.59) is set in the Document Settings part, then the characters it specifies cannot begin a line for the specified language. If the noLineBreaksAfter property (§2.15.1.58) is set in the Document Settings part, then the characters it specifies cannot end a line for the specified language. In both cases, those settings shall supersede the defaults specified above.

[*Example*: Consider a document with a paragraph which should not use the kinsoku line breaking properties. This paragraph would define the following WordprocessingML:

```
<w:pPr>
  <w:kinsoku w:val="off" />
</w:pPr>
```

This paragraph would now be exempt from any kinsoku line breaking rules, and the characters specified above are allowed to begin and end lines as they normally would. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.1.17 left (Left Paragraph Border)

This element specifies the border which shall be displayed on the left side of the page around the specified paragraph.

To determine if any two adjoining paragraphs should have a left border which spans the full line height or not, the left border shall be drawn between the top border or between border at the top (whichever would be rendered for the current paragraph), and the bottom border or between border at the bottom (whichever would be rendered for the current paragraph).

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then no left border shall be applied.

[*Example*: Consider the following two paragraphs' WordprocessingML definition:

```
<w:p>
  <w:pPr>
```

```
    <w:pBdr>
      <w:top w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="B97034"
w:themeColor="accent6" w:themeShade="BF" />
      <w:bottom w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:right w:val="single" w:sz="24" w:space="4" w:color="C3D69B"
w:themeColor="accent3" w:themeTint="99" />
      <w:between w:val="single" w:sz="24" w:space="1" w:color="4F81BD"
w:themeColor="accent1" />
    </w:pBdr>
  </w:pPr>
  <w:r>
    <w:t>First paragraph.</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:pBdr>
      <w:top w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="B97034"
w:themeColor="accent6" w:themeShade="BF" />
      <w:bottom w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:right w:val="single" w:sz="24" w:space="4" w:color="C3D69B"
w:themeColor="accent3" w:themeTint="99" />
      <w:between w:val="single" w:sz="24" w:space="1" w:color="4F81BD"
w:themeColor="accent1" />
    </w:pBdr>
  </w:pPr>
  <w:r>
    <w:t>Second paragraph.</w:t>
  </w:r>
</w:p>
```

Since the paragraph border set is identical between the two paragraphs, the paragraphs are connected by a between border. These paragraphs will therefore draw the left border between the top and between borders for the first paragraph, and the between and bottom borders for the second paragraph. *end example*]

Parent Elements
pBdr (§2.3.1.24)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or auto to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value auto, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is true, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the

Attributes	Description
	following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object.
	
	When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points.
	
	When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.
	
	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24/>`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*]
	
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).

sz (Border Width)	Specifies the width of the current border. If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value. If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value. [*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:

Attributes	Description
	<w:top w:val="dashed" w:sz="24" …/> <w:left w:val="dashed" w:sz="24" …/> <w:bottom w:val="dashed" w:sz="24" …/> <w:right w:val="dashed" w:sz="24" …/> The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*] The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).
themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border. The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document. [*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup: <w:top … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:bottom … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:left … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:right … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*] The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).
themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is

Attributes	Description
	calculated as follows: $$S_{xml} = 0.4 * 255$$ $$= 102$$ $$= 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}^{'} = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L^{'} = 0.53 * 0.75$$ $$= 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border.

Attributes	Description
	The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255$$ $$= 153$$ $$= 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6)$$ $$= 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object.

Attributes	Description
	This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <attribute name="val" type="ST_Border" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
   <attribute name="space" type="ST_PointMeasure" use="optional"/>
   <attribute name="shadow" type="ST_OnOff" use="optional"/>
   <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.3.1.18 mirrorIndents (Use Left/Right Indents as Inside/Outside Indents)

This element specifies whether the paragraph indents should be interpreted as mirrored indents. When this element is present, the left indent shall become the inside indent and the right indent shall become the outside indent.

If the mirrorIndents property is specified for this paragraph, then the inside page edge is the right page edge for odd numbered pages and the left page edge for even numbered pages. Conversely, the outside page edge is the left page edge for odd numbered pages and the right page edge for even numbered pages.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then this property shall not be applied.

[*Example*: Consider a paragraph on the first page of a document which should have a one inch indentation from the text margins on the inside edge when the resulting document is printed and bound. This means that the paragraph will have a one inch right border if it is on an odd numbered page, and a one inch left border if it is on an even numbered page. This set of indentations is specified using the following WordprocessingML:

```
<w:pPr>
  <w:ind w:left="1440" />
  <w:mirrorIndents />
```

```
</w:pPr>
```

This set of indentation properties specifies that a 1440 twip indentation should be provided on the left side of the text margins for this paragraph. However, since the mirrorIndents property is set, the left indent is really the inside indent, and if this paragraph is on page one, shall result in a one inch right indent from the text margin. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.1.19 numPr (Numbering Definition Instance Reference)

This element specifies that the current paragraph references a *numbering definition instance* in the current document.

The presence of this element specifies that the paragraph will inherit the properties specified by the numbering definition in the num element (§2.9.16) at the level specified by the level specified in the lvl element (§2.9.7) and shall have an associated number positioned before the beginning of the text flow in this paragraph. When this element appears as part of the paragraph formatting for a paragraph style, then any numbering level defined using the ilvl element shall be ignored, and the pStyle element (§2.9.25) on the associated abstract numbering definition shall be used instead.

[*Example*: Consider a paragraph in a document which should be associated with level 4 of a numbering definition with ID 0. Associating the paragraph with this numbering definition would be specified using the following WordprocessingML:

```
<w:pPr>
  <w:numPr>
    <w:ilvl w:val="4" />
    <w:numId w:val="0" />
  </w:numPr>
</w:pPr>
```

The numPr element specifies that this paragraph shall contain numbering information, and its children specify that the numbering definition for that numbering information shall have a numId of 0 and an ilvl of 4 within that numbering definition. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Child Elements	Subclause
ilvl (Numbering Level Reference)	§2.9.3
ins (Inserted Numbering Properties)	§2.13.5.19
numberingChange (Previous Paragraph Numbering Properties)	§2.13.5.30
numId (Numbering Definition Instance Reference)	§2.9.19

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NumPr">
   <sequence>
      <element name="ilvl" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="numId" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="numberingChange" type="CT_TrackChangeNumbering" minOccurs="0"/>
      <element name="ins" type="CT_TrackChange" minOccurs="0"/>
   </sequence>
</complexType>
```

2.3.1.20 outlineLvl (Associated Outline Level)

This element specifies the *outline level* which shall be associated with the current paragraph in the document. The *outline level* specifies an integer which defines the level of the associated text. This level shall not affect the appearance of the text in the document, but shall be used to calculate the TOC field (§2.16.5.75) if the appropriate field switches have been set, and may be used by consumers to provide additional application behavior.

The outline level of text in the document (specified using the val attribute) may be from 0 to 9, where 9 specifically indicates that there is no outline level specifically applied to this paragraph. If this element is omitted, then the outline level of the content is assumed to be 9 (no level).

[*Example*: Consider a paragraph in a document which has outline level 1 applied to it. This paragraph would specify the following WordprocessingML:

```
<w:pPr>
  <w:outlineLvl w:val="0" />
</w:pPr>
```

This paragraph is now of outline level 1, and if a table of contents field is inserted that utilizes outlines levels, the text in this paragraph will be at level one in the TOC. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.3.1.21 overflowPunct (Allow Punctuation to Extent Past Text Extents)

This element specifies that punctuation characters which appear at the end of text in any line in this paragraph shall be allowed to extend one character past the text extents (indents/margins) as needed in order to ensure that they are not displayed as hanging punctuation. *Hanging punctuation* is defined as punctuation which appears on a different line than the text which it logically would appear with.

Omitting this element sets its value to `true`.

[*Example*: Consider a WordprocessingML document with the following string at the end of a line:

```
"This is some text in quotation marks"
```

Typically, if the text extents would normally fall between the letter s and the closing quotation mark, the quotation mark would be allowed to extend past the end of the line by one character even though the punctuation is not part of the word `marks` (since the omission of overflowPunct is equivalent to setting its val attribute to `true`).

However, if this behavior should not be applied to this paragraph, a producer can specify this by setting the property in the WordprocessingML:

```
<w:pPr>
  <w:overflowPunct w:val="0" />
</w:pPr>
```

The line would now break after the letter `s`, regardless of the fact that the next character is a quotation mark. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.1.22 p (Paragraph)

This element specifies a paragraph of content in the document.

The contents of a paragraph in a WordprocessingML document shall consist of any combination of the following four types of content:

- Paragraph properties
- Annotations (bookmarks, comments, revisions)
- Custom markup
- Run level content (fields, hyperlinks, runs)

[*Example*: Consider a basic WordprocessingML document with a single paragraph. This paragraph would be expressed as follows:

```
<w:document>
  <w:body>
    <w:p>
      <w:r>
        <w:t>Text</w:t>
      </w:r>
      <w:fldSimple w:instr="AUTHOR">
        <w:r>
          <w:t>Author Name</w:t>
        </w:r>
      </w:fldSimple>
    </w:p>
  </w:body>
</w:document>
```

The p element is the container for all of the content in the paragraph, which in this example includes both a text run and a simple field. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.6); docPartBody (§2.12.6); endnote (§2.11.2); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); sdtContent (§2.5.2.32); tc (§2.4.62); txbxContent (§2.17.1.1)

Child Elements	Subclause
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Inline-Level Custom XML Element)	§2.5.1.5

Child Elements	Subclause
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
del (Deleted Run Content)	§2.13.5.12
fldSimple (Simple Field)	§2.16.21
hyperlink (Hyperlink)	§2.16.24
ins (Inserted Run Content)	§2.13.5.20
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
pPr (Paragraph Properties)	§2.3.1.26
proofErr (Proofing Error Anchor)	§2.13.8.1
r (Text Run)	§2.3.2.23
sdt (Inline-Level Structured Document Tag)	§2.5.2.29
smartTag (Inline-Level Smart Tag)	§2.5.1.9
subDoc (Anchor for Subdocument Location)	§2.17.2.1

Attributes	Description
rsidDel (Revision Identifier for Paragraph Deletion)	Specifies a unique identifier used to track the editing session when the paragraph was deleted from the main document. All rsid* attributes throughout this document with the same value, if present, must indicate that those regions were modified during the same editing session (time between

Attributes	Description
	subsequent save actions). A producer may choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications in this document. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).
rsidP (Revision Identifier for Paragraph Properties)	This attribute specifies a unique identifier used to track the editing session when the paragraph's properties were last modified in this document. All rsid* attributes throughout this document with the same value, if present, must indicate that those regions were modified during the same editing session (time between subsequent save actions). A producer may choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications in this document. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).
rsidR (Revision Identifier for Paragraph)	This attribute specifies a unique identifier used to track the editing session when the paragraph was added to the main document. All rsid* attributes throughout this document with the same value, if present, must indicate that those regions were modified during the same editing session (time between subsequent save actions). A producer may choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications in this document. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).
rsidRDefault (Default Revision Identifier for Runs)	This attribute specifies a unique identifier used for all runs in this paragraph which do not explicitly declare an rsidR attribute. This attribute allows consumers to optimize the locations where rsid* values are written in this document. All rsid* attributes throughout this document with the same value, if present, must indicate that those regions were modified during the same editing session (time between subsequent save actions). A producer may choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications in this document.

Attributes	Description
	The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).
rsidRPr (Revision Identifier for Paragraph Glyph Formatting)	This attribute specifies a unique identifier used to track the editing session when the glyph character representing the paragraph mark was last modified in the main document. All rsid* attributes throughout this document with the same value, if present, must indicate that those regions were modified during the same editing session (time between subsequent save actions). A producer may choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications in this document. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_P">
    <sequence>
        <element name="pPr" type="CT_PPr" minOccurs="0"/>
        <group ref="EG_PContent" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="rsidRPr" type="ST_LongHexNumber"/>
    <attribute name="rsidR" type="ST_LongHexNumber"/>
    <attribute name="rsidDel" type="ST_LongHexNumber"/>
    <attribute name="rsidP" type="ST_LongHexNumber"/>
    <attribute name="rsidRDefault" type="ST_LongHexNumber"/>
</complexType>
```

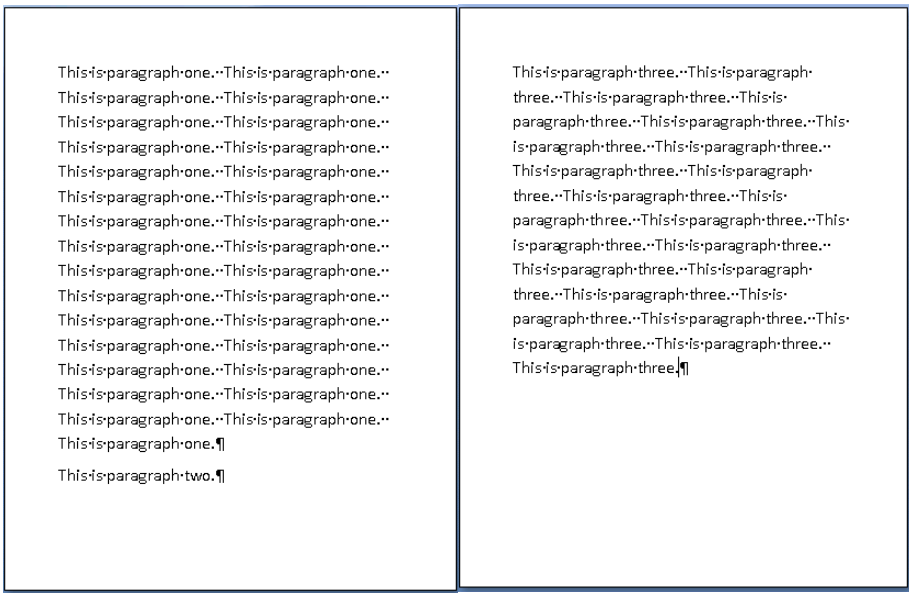
2.3.1.23 pageBreakBefore (Start Paragraph on Next Page)

This element specifies that when rendering this document in a paginated view, the contents of this paragraph are rendered on the start of a new page in the document.

This means that if the contents of the current paragraph would normally be rendered on the middle of a page in the host document, then the paragraph shall be rendered on a new page as if the paragraph was preceded by a page break in the WordprocessingML contents of the document. This property supersedes any use of the keepNext property, so that if any paragraph wishes to be on the same page as this paragraph, they will still be separated by a page break.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then this property shall not be applied.

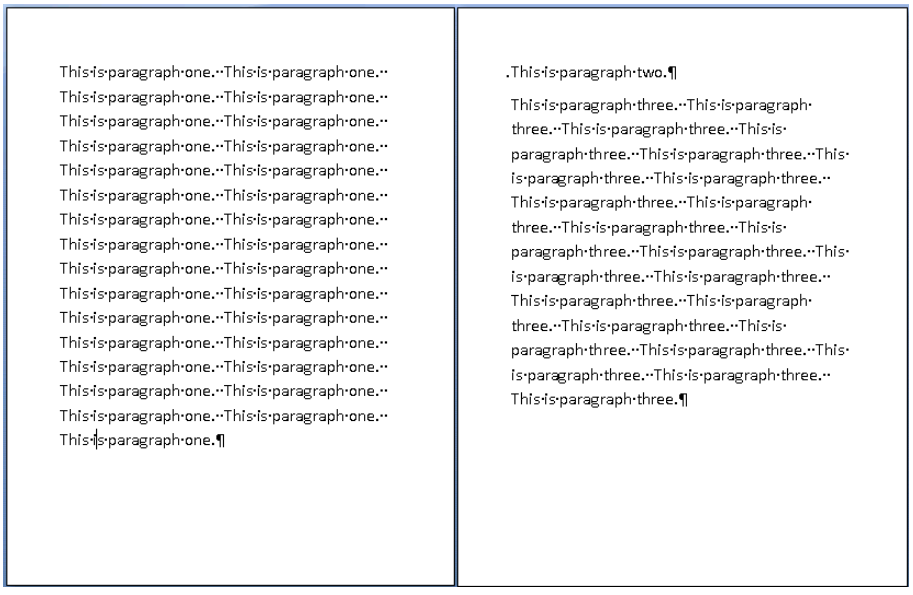
[*Example*: Consider the following document with three paragraphs:



As shown above, the second paragraph is rendered at the bottom of page one. However, a producer can specify that the second paragraph should be displayed at the top of a new page by setting the pageBreakBefore element as follows:

```
<w:pPr>
  <w:pageBreakBefore/>
</w:pPr>
```

This would ensure that the second paragraph is displayed on a new page:



Since the paragraph is specified to start on a new page, it begins page two even though it could have fit on page one. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.1.24 pBdr (Paragraph Borders)

This element specifies the borders for the parent paragraph. Each child element shall specify a specific type of border (left, right, bottom, top, and between).

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then no paragraph borders shall be applied.

[*Example*: Consider a pairing of paragraphs which have a three point red border around them, and a six point border between them. These paragraphs would each have the following set of paragraph borders:

```
<w:pBdr>
  <w:top w:val="single" w:sz="24" w:space="1" w:color="FF0000" />
  <w:left w:val="single" w:sz="24" w:space="4" w:color="FF0000" />
  <w:bottom w:val="single" w:sz="24" w:space="1" w:color="FF0000" />
  <w:right w:val="single" w:sz="24" w:space="4" w:color="FF0000" />
```

```
    <w:between w:val="single" w:sz="48" w:space="1" w:color="4D5D2C" />
  </w:pBdr>
```

The resulting paragraphs have identical pBdr values, therefore they would use the top, left, bottom, and right borders around them as a units, and the between border between each other. This matching heuristic is further discussed in the child elements of the pBdr element. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Child Elements	Subclause
bar (Paragraph Border Between Facing Pages)	§2.3.1.4
between (Paragraph Border Between Identical Paragraphs)	§2.3.1.5
bottom (Paragraph Border Between Identical Paragraphs)	§2.3.1.7
left (Left Paragraph Border)	§2.3.1.17
right (Right Paragraph Border)	§2.3.1.28
top (Paragraph Border Above Identical Paragraphs)	§2.3.1.42

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PBdr">
   <sequence>
      <element name="top" type="CT_Border" minOccurs="0"/>
      <element name="left" type="CT_Border" minOccurs="0"/>
      <element name="bottom" type="CT_Border" minOccurs="0"/>
      <element name="right" type="CT_Border" minOccurs="0"/>
      <element name="between" type="CT_Border" minOccurs="0"/>
      <element name="bar" type="CT_Border" minOccurs="0"/>
   </sequence>
</complexType>
```

2.3.1.25 pPr (Previous Paragraph Properties)

This element specifies a set of paragraph properties which shall be attributed to a revision by a particular author and at a particular time. This element contains the set of properties which have been tracked as a specific set of revisions by one author.

[*Example*: Consider a paragraph which should have a set of paragraph formatting properties that were added with revision tracking turned on. This set of revised properties is specified in the paragraph properties as follows:

```
<w:p>
  <w:pPr>
    <w:pBdr>
      <w:bottom w:val="single" w:sz="8" w:space="4" w:color="4F81BD" />
```

```
      </w:pBdr>
      <w:pPrChange w:author="user1" … >
        <w:pPr>
          <w:spacing w:after="300" />
          <w:contextualSpacing />
        </w:pPr>
      </w:pPrChange>
    </w:pPr>
  </w:p>
```

The pPr element under pPrChange specifies the properties which are applied to the current paragraph with revision tracking turned on - in this case, spacing after the paragraph using the spacing element (§2.3.1.33), and that spacing should be ignored for paragraphs above/below of the same style using the contextualSpacing element (§2.3.1.9). *end example*]

Parent Elements
pPrChange (§2.13.5.31)

Child Elements	Subclause
adjustRightInd (Automatically Adjust Right Indent When Using Document Grid)	§2.3.1.1
autoSpaceDE (Automatically Adjust Spacing of Latin and East Asian Text)	§2.3.1.2
autoSpaceDN (Automatically Adjust Spacing of East Asian Text and Numbers)	§2.3.1.3
bidi (Right to Left Paragraph Layout)	§2.3.1.6
cnfStyle (Paragraph Conditional Formatting)	§2.3.1.8
contextualSpacing (Ignore Spacing Above and Below When Using Identical Styles)	§2.3.1.9
divId (Associated HTML div ID)	§2.3.1.10
framePr (Text Frame Properties)	§2.3.1.11
ind (Paragraph Indentation)	§2.3.1.12
jc (Paragraph Alignment)	§2.3.1.13
keepLines (Keep All Lines On One Page)	§2.3.1.14
keepNext (Keep Paragraph With Next Paragraph)	§2.3.1.15
kinsoku (Use East Asian Typography Rules for First and Last Character per Line)	§2.3.1.16
mirrorIndents (Use Left/Right Indents as Inside/Outside Indents)	§2.3.1.18
numPr (Numbering Definition Instance Reference)	§2.3.1.19
outlineLvl (Associated Outline Level)	§2.3.1.20
overflowPunct (Allow Punctuation to Extent Past Text Extents)	§2.3.1.21
pageBreakBefore (Start Paragraph on Next Page)	§2.3.1.23
pBdr (Paragraph Borders)	§2.3.1.24

Child Elements	Subclause
pStyle (Referenced Paragraph Style)	§2.3.1.27
shd (Paragraph Shading)	§2.3.1.31
snapToGrid (Use Document Grid Settings for Inter-Line Paragraph Spacing)	§2.3.1.32
spacing (Spacing Between Lines and Above/Below Paragraph)	§2.3.1.33
suppressAutoHyphens (Suppress Hyphenation for Paragraph)	§2.3.1.34
suppressLineNumbers (Suppress Line Numbers for Paragraph)	§2.3.1.35
suppressOverlap (Prevent Text Frames From Overlapping)	§2.3.1.36
tabs (Set of Custom Tab Stops)	§2.3.1.38
textAlignment (Vertical Character Alignment on Line)	§2.3.1.39
textboxTightWrap (Allow Surrounding Paragraphs to Tight Wrap to Text Box Contents)	§2.3.1.40
textDirection (Paragraph Text Flow Direction)	§2.3.1.41
topLinePunct (Compress Punctuation at Start of a Line)	§2.3.1.43
widowControl (Allow First/Last Line to Display on a Separate Page)	§2.3.1.44
wordWrap (Allow Line Breaking At Character Level)	§2.3.1.45

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PPrBase">
   <sequence>
      <element name="pStyle" type="CT_String" minOccurs="0"/>
      <element name="keepNext" type="CT_OnOff" minOccurs="0"/>
      <element name="keepLines" type="CT_OnOff" minOccurs="0"/>
      <element name="pageBreakBefore" type="CT_OnOff" minOccurs="0"/>
      <element name="framePr" type="CT_FramePr" minOccurs="0"/>
      <element name="widowControl" type="CT_OnOff" minOccurs="0"/>
      <element name="numPr" type="CT_NumPr" minOccurs="0"/>
      <element name="suppressLineNumbers" type="CT_OnOff" minOccurs="0"/>
      <element name="pBdr" type="CT_PBdr" minOccurs="0"/>
      <element name="shd" type="CT_Shd" minOccurs="0"/>
      <element name="tabs" type="CT_Tabs" minOccurs="0"/>
      <element name="suppressAutoHyphens" type="CT_OnOff" minOccurs="0"/>
      <element name="kinsoku" type="CT_OnOff" minOccurs="0"/>
      <element name="wordWrap" type="CT_OnOff" minOccurs="0"/>
      <element name="overflowPunct" type="CT_OnOff" minOccurs="0"/>
      <element name="topLinePunct" type="CT_OnOff" minOccurs="0"/>
      <element name="autoSpaceDE" type="CT_OnOff" minOccurs="0"/>
      <element name="autoSpaceDN" type="CT_OnOff" minOccurs="0"/>
      <element name="bidi" type="CT_OnOff" minOccurs="0"/>
      <element name="adjustRightInd" type="CT_OnOff" minOccurs="0"/>
      <element name="snapToGrid" type="CT_OnOff" minOccurs="0"/>
      <element name="spacing" type="CT_Spacing" minOccurs="0"/>
      <element name="ind" type="CT_Ind" minOccurs="0"/>
      <element name="contextualSpacing" type="CT_OnOff" minOccurs="0"/>
      <element name="mirrorIndents" type="CT_OnOff" minOccurs="0"/>
      <element name="suppressOverlap" type="CT_OnOff" minOccurs="0"/>
      <element name="jc" type="CT_Jc" minOccurs="0"/>
      <element name="textDirection" type="CT_TextDirection" minOccurs="0"/>
      <element name="textAlignment" type="CT_TextAlignment" minOccurs="0"/>
      <element name="textboxTightWrap" type="CT_TextboxTightWrap" minOccurs="0"/>
      <element name="outlineLvl" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="divId" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="cnfStyle" type="CT_Cnf" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

2.3.1.26 pPr (Paragraph Properties)

This element specifies a set of paragraph properties which shall be applied to the contents of the parent paragraph after all style/numbering/table properties have been applied to the text. These properties are defined as *direct formatting*, since they are directly applied to the paragraph and supersede any formatting from styles.

[*Example*: Consider a paragraph which should have a set of paragraph formatting properties. This set of properties is specified in the paragraph properties as follows:

```
<w:p>
  <w:pPr>
    <w:pBdr>
```

```
      <w:bottom w:val="single" w:sz="8" w:space="4" w:color="4F81BD" />
    </w:pBdr>
    <w:spacing w:after="300" />
    <w:contextualSpacing />
  </w:pPr>
</w:p>
```

The pPr element specifies the properties which are applied to the current paragraph - in this case, a bottom paragraph border using the bottom element (§2.3.1.7), spacing after the paragraph using the spacing element (§2.3.1.33), and that spacing should be ignored for paragraphs above/below of the same style using the contextualSpacing element (§2.3.1.9). *end example*]

Parent Elements
p (§2.3.1.22)

Child Elements	Subclause
adjustRightInd (Automatically Adjust Right Indent When Using Document Grid)	§2.3.1.1
autoSpaceDE (Automatically Adjust Spacing of Latin and East Asian Text)	§2.3.1.2
autoSpaceDN (Automatically Adjust Spacing of East Asian Text and Numbers)	§2.3.1.3
bidi (Right to Left Paragraph Layout)	§2.3.1.6
cnfStyle (Paragraph Conditional Formatting)	§2.3.1.8
contextualSpacing (Ignore Spacing Above and Below When Using Identical Styles)	§2.3.1.9
divId (Associated HTML div ID)	§2.3.1.10
framePr (Text Frame Properties)	§2.3.1.11
ind (Paragraph Indentation)	§2.3.1.12
jc (Paragraph Alignment)	§2.3.1.13
keepLines (Keep All Lines On One Page)	§2.3.1.14
keepNext (Keep Paragraph With Next Paragraph)	§2.3.1.15
kinsoku (Use East Asian Typography Rules for First and Last Character per Line)	§2.3.1.16
mirrorIndents (Use Left/Right Indents as Inside/Outside Indents)	§2.3.1.18
numPr (Numbering Definition Instance Reference)	§2.3.1.19
outlineLvl (Associated Outline Level)	§2.3.1.20
overflowPunct (Allow Punctuation to Extent Past Text Extents)	§2.3.1.21
pageBreakBefore (Start Paragraph on Next Page)	§2.3.1.23
pBdr (Paragraph Borders)	§2.3.1.24
pPrChange (Revision Information for Paragraph Properties)	§2.13.5.31
pStyle (Referenced Paragraph Style)	§2.3.1.27

Child Elements	Subclause
rPr (Run Properties for the Paragraph Mark)	§2.3.1.29
sectPr (Section Properties)	§2.6.19
shd (Paragraph Shading)	§2.3.1.31
snapToGrid (Use Document Grid Settings for Inter-Line Paragraph Spacing)	§2.3.1.32
spacing (Spacing Between Lines and Above/Below Paragraph)	§2.3.1.33
suppressAutoHyphens (Suppress Hyphenation for Paragraph)	§2.3.1.34
suppressLineNumbers (Suppress Line Numbers for Paragraph)	§2.3.1.35
suppressOverlap (Prevent Text Frames From Overlapping)	§2.3.1.36
tabs (Set of Custom Tab Stops)	§2.3.1.38
textAlignment (Vertical Character Alignment on Line)	§2.3.1.39
textboxTightWrap (Allow Surrounding Paragraphs to Tight Wrap to Text Box Contents)	§2.3.1.40
textDirection (Paragraph Text Flow Direction)	§2.3.1.41
topLinePunct (Compress Punctuation at Start of a Line)	§2.3.1.43
widowControl (Allow First/Last Line to Display on a Separate Page)	§2.3.1.44
wordWrap (Allow Line Breaking At Character Level)	§2.3.1.45

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PPr">
   <complexContent>
      <extension base="CT_PPrBase">
         <sequence>
            <element name="rPr" type="CT_ParaRPr" minOccurs="0"/>
            <element name="sectPr" type="CT_SectPr" minOccurs="0"/>
            <element name="pPrChange" type="CT_PPrChange" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.3.1.27 pStyle (Referenced Paragraph Style)

This element specifies the style ID of the paragraph style which shall be used to format the contents of this paragraph.

This formatting is applied at the following location in the *style hierarchy*:

- Document defaults
- Table styles
- Numbering styles
- Paragraph styles (this element)
- Character styles
- Direct Formatting

This means that all properties specified in the style element (§2.7.3.17) with a styleId which corresponds to the value in this element's val attribute are applied to the paragraph at the appropriate level in the hierarchy.

If this element is omitted, or it references a style which does not exist, then no paragraph style shall be applied to the current paragraph. As well, this property is ignored if the paragraph properties are part of a paragraph style.

[*Example*: Consider the following WordprocessingML fragment:

```
<w:pPr>
  <w:pStyle w:val=”TestParagraphStyle” />
  <w:ind w:left=”1440” />
</w:pPr>
```

This paragraph specifies that it will inherit all of the paragraph properties specified by the paragraph style with a styleId of `TestParagraphStyle`, which will then have any indentation properties overridden with a left indentation of `1440` twentieths of a point, and no indentation for any other value. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.3.1.28 right (Right Paragraph Border)

This element specifies the border which shall be displayed on the right side of the page around the specified paragraph.

To determine if any two adjoining paragraphs should have a right border which spans the full line height or not, the right border shall be drawn between the top border or between border at the top (whichever would be rendered for the current paragraph), and the bottom border or between border at the bottom (whichever would be rendered for the current paragraph).

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then no right border shall be applied.

[*Example*: Consider the following two paragraphs' WordprocessingML definition:

```
<w:p>
  <w:pPr>
    <w:pBdr>
      <w:top w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="B97034"
w:themeColor="accent6" w:themeShade="BF" />
      <w:bottom w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:right w:val="single" w:sz="24" w:space="4" w:color="C3D69B"
w:themeColor="accent3" w:themeTint="99" />
      <w:between w:val="single" w:sz="24" w:space="1" w:color="4F81BD"
w:themeColor="accent1" />
    </w:pBdr>
  </w:pPr>
  <w:r>
    <w:t>First paragraph.</w:t>
  </w:r>
</w:p>
<w:p>
```

```
  <w:pPr>
    <w:pBdr>
      <w:top w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="B97034"
w:themeColor="accent6" w:themeShade="BF" />
      <w:bottom w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:right w:val="single" w:sz="24" w:space="4" w:color="C3D69B"
w:themeColor="accent3" w:themeTint="99" />
      <w:between w:val="single" w:sz="24" w:space="1" w:color="4F81BD"
w:themeColor="accent1" />
    </w:pBdr>
  </w:pPr>
  <w:r>
    <w:t>Second paragraph.</w:t>
  </w:r>
</w:p>
```

Since the paragraph border set is identical between the two paragraphs, the paragraphs are connected by a between border. These paragraphs will therefore draw the right border between the top and between borders for the first paragraph, and the between and bottom borders for the second paragraph. *end example*]

Parent Elements
pBdr (§2.3.1.24)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or auto to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value auto, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value.

Attributes	Description
	The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is `true`, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object. When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points. When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.

Attributes	Description

	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24/>`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*]
	
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).

sz (Border Width)	Specifies the width of the current border.
	
	If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value.
	
	If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value.
	
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`
	
	The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	
	The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).

themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border. The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document. [*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup:

Attributes	Description
	<w:top … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:bottom … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:left … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:right … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*] The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).
themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255 = 102 = 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D.

Attributes	Description
	The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $L^{'} = 0.53 * 0.75$ $= 0.39698$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $T_{xml} = 0.6 * 255$ $= 153$ $= 99(hex)$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $\mathrm{L}^{'} = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$ • Convert the resultant HSL color to RGB

Attributes	Description
	[*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `4F81BD`. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $L' = 0.53 * 0.6 + (1 - .6)$ $= 0.71$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get `95B3D7`. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <attribute name="val" type="ST_Border" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
   <attribute name="space" type="ST_PointMeasure" use="optional"/>
   <attribute name="shadow" type="ST_OnOff" use="optional"/>
   <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.3.1.29 rPr (Run Properties for the Paragraph Mark)

This element specifies the set of run properties applied to the glyph used to represent the physical location of the paragraph mark for this paragraph. This paragraph mark, being a physical character in the document, can be formatted, and therefore must be capable of representing this formatting like any other character in the document.

If this element is not present, the paragraph mark is unformatted, as with any other run of text.

[*Example*: Consider a run of text displayed as follows, including a display format using the pilcrow sign ¶ for the paragraph mark glyph:

```
This is some text and the paragraph mark.¶
```

If we format the display formatting for the paragraph mark by making it red and giving it a 72 point font size, then the WordprocessingML shall reflect this formatting on the paragraph as follows:

```
<w:pPr>
  <w:rPr>
    <w:color w:val="FF0000" />
    <w:sz w:val="144" />
  </w:rPr>
</w:pPr>
```

The paragraph glyph's formatting is stored in the rPr element under the paragraph properties, since there is no run saved for the paragraph mark itself. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Child Elements	Subclause
b (Bold)	§2.3.2.1
bCs (Complex Script Bold)	§2.3.2.2

Child Elements	Subclause
bdr (Text Border)	§2.3.2.3
caps (Display All Characters As Capital Letters)	§2.3.2.4
color (Run Content Color)	§2.3.2.5
cs (Use Complex Script Formatting on Run)	§2.3.2.6
del (Deleted Paragraph)	§2.13.5.13
dstrike (Double Strikethrough)	§2.3.2.7
eastAsianLayout (East Asian Typography Settings)	§2.3.2.8
effect (Animated Text Effect)	§2.3.2.9
em (Emphasis Mark)	§2.3.2.10
emboss (Embossing)	§2.3.2.11
fitText (Manual Run Width)	§2.3.2.12
highlight (Text Highlighting)	§2.3.2.13
i (Italics)	§2.3.2.14
iCs (Complex Script Italics)	§2.3.2.15
imprint (Imprinting)	§2.3.2.16
ins (Inserted Paragraph)	§2.13.5.18
kern (Font Kerning)	§2.3.2.17
lang (Languages for Run Content)	§2.3.2.18
moveFrom (Move Source Paragraph)	§2.13.5.22
moveTo (Move Destination Paragraph)	§2.13.5.25
noProof (Do Not Check Spelling or Grammar)	§2.3.2.19
oMath (Office Open XML Math)	§2.3.2.20
outline (Display Character Outline)	§2.3.2.21
position (Vertically Raised or Lowered Text)	§2.3.2.22
rFonts (Run Fonts)	§2.3.2.24
rPrChange (Revision Information for Run Properties on the Paragraph Mark)	§2.13.5.33
rStyle (Referenced Character Style)	§2.3.2.27
rtl (Right To Left Text)	§2.3.2.28
shadow (Shadow)	§2.3.2.29
shd (Run Shading)	§2.3.2.30
smallCaps (Small Caps)	§2.3.2.31
snapToGrid (Use Document Grid Settings For Inter-Character Spacing)	§2.3.2.32
spacing (Character Spacing Adjustment)	§2.3.2.33
specVanish (Paragraph Mark Is Always Hidden)	§2.3.2.34

Child Elements	Subclause
strike (Single Strikethrough)	§2.3.2.35
sz (Font Size)	§2.3.2.36
szCs (Complex Script Font Size)	§2.3.2.37
u (Underline)	§2.3.2.38
vanish (Hidden Text)	§2.3.2.39
vertAlign (Subscript/Superscript Text)	§2.3.2.40
w (Expanded/Compressed Text)	§2.3.2.41
webHidden (Web Hidden Text)	§2.3.2.42

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ParaRPr">
   <sequence>
      <group ref="EG_ParaRPrTrackChanges" minOccurs="0"/>
      <group ref="EG_RPrBase" minOccurs="0"/>
      <element name="rPrChange" type="CT_ParaRPrChange" minOccurs="0"/>
   </sequence>
</complexType>
```

2.3.1.30 rPr (Previous Run Properties for the Paragraph Mark)

This element specifies a set of run properties applied to the glyph used to represent the physical location of the paragraph mark for this paragraph which shall be attributed to a revision by a particular author and at a particular time. This element contains the set of properties which have been tracked as a specific set of revisions by one author.

[*Example*: Consider a run which has a set of run formatting properties that were added with revision tracking turned on. This set of revised properties is specified in the run properties as follows:

```
<w:p>
  <w:pPr>
    <w:rPr>
      <w:b />
      <w:imprint />
      <w:lang w:val="en-ca" />
      <w:rPrChange w:author="user1">
        <w:rPr>
          <w:i />
          <w:dstrike w:val="false" />
        </w:rPr>
      </w:rPrChange>
    </w:rPr>
  </w:pPr>
</w:p>
```

The rPr element under rPrChange specifies the properties which are applied to the run representing the paragraph mark before the revision tracking was turned on - in this case, italics using the i element (§2.3.2.14), and that any double strikethrough which was applied based on the style hierarchy shall be turned off using the dstrike element (§2.3.2.7). *end example*]

Parent Elements
rPrChange (§2.13.5.33)

Child Elements	Subclause
b (Bold)	§2.3.2.1
bCs (Complex Script Bold)	§2.3.2.2
bdr (Text Border)	§2.3.2.3
caps (Display All Characters As Capital Letters)	§2.3.2.4
color (Run Content Color)	§2.3.2.5
cs (Use Complex Script Formatting on Run)	§2.3.2.6
del (Deleted Paragraph)	§2.13.5.13
dstrike (Double Strikethrough)	§2.3.2.7
eastAsianLayout (East Asian Typography Settings)	§2.3.2.8
effect (Animated Text Effect)	§2.3.2.9
em (Emphasis Mark)	§2.3.2.10
emboss (Embossing)	§2.3.2.11
fitText (Manual Run Width)	§2.3.2.12
highlight (Text Highlighting)	§2.3.2.13
i (Italics)	§2.3.2.14
iCs (Complex Script Italics)	§2.3.2.15
imprint (Imprinting)	§2.3.2.16
ins (Inserted Paragraph)	§2.13.5.18
kern (Font Kerning)	§2.3.2.17
lang (Languages for Run Content)	§2.3.2.18
moveFrom (Move Source Paragraph)	§2.13.5.22
moveTo (Move Destination Paragraph)	§2.13.5.25
noProof (Do Not Check Spelling or Grammar)	§2.3.2.19
oMath (Office Open XML Math)	§2.3.2.20
outline (Display Character Outline)	§2.3.2.21
position (Vertically Raised or Lowered Text)	§2.3.2.22
rFonts (Run Fonts)	§2.3.2.24

Child Elements	Subclause
rStyle (Referenced Character Style)	§2.3.2.27
rtl (Right To Left Text)	§2.3.2.28
shadow (Shadow)	§2.3.2.29
shd (Run Shading)	§2.3.2.30
smallCaps (Small Caps)	§2.3.2.31
snapToGrid (Use Document Grid Settings For Inter-Character Spacing)	§2.3.2.32
spacing (Character Spacing Adjustment)	§2.3.2.33
specVanish (Paragraph Mark Is Always Hidden)	§2.3.2.34
strike (Single Strikethrough)	§2.3.2.35
sz (Font Size)	§2.3.2.36
szCs (Complex Script Font Size)	§2.3.2.37
u (Underline)	§2.3.2.38
vanish (Hidden Text)	§2.3.2.39
vertAlign (Subscript/Superscript Text)	§2.3.2.40
w (Expanded/Compressed Text)	§2.3.2.41
webHidden (Web Hidden Text)	§2.3.2.42

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ParaRPrOriginal">
   <sequence>
      <group ref="EG_ParaRPrTrackChanges" minOccurs="0"/>
      <group ref="EG_RPrBase" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

2.3.1.31 shd (Paragraph Shading)

This element specifies the shading applied to the contents of the paragraph.

This shading consists of three components:

- Background Color
- (optional) Pattern
- (optional) Pattern Color

The resulting shading is applied by setting the background color behind the paragraph, then applying the pattern color using the mask supplied by the pattern over that background.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then no paragraph shading shall be applied.

[*Example*: Consider a paragraph which shall have a background consisting of a theme color `accent3` with a theme color `accent6` overlaid using a 20% fill pattern. This requirement is specified using the following WordprocessingML:

```
<w:pPr>
  <w:shd w:val="pct20" w:themeColor="accent6" w:themeFill="accent3" />
</w:pPr>
```

The resulting paragraph will use the background color `accent3` under the foreground pattern color `accent6` as specified by the `pct20` pattern mask. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description
color (Shading Pattern Color)	Specifies the color used for any foreground pattern specified for this shading using the val attribute. This color may either be presented as a hex value (in RRGGBB format), or `auto` to allow a consumer to automatically determine the foreground shading color as appropriate. If the shading style (the val attribute) specifies the use of no shading format or is omitted, then this property has no effect. Also, if the shading specifies the use of a theme color via the themeColor attribute, then this value is superseded by the theme color value. If this attribute is omitted, then its value shall be assumed to be `auto`. [*Example*: Consider a shading of type `pct20` with a foreground color value of `auto`, as follows: `<w:shd w:val="pct20"... w:color="auto"/>` The foreground color for this shading pattern therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the shading color can be distinguished against the page's background color. *end example*] The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
fill (Shading Background Color)	Specifies the color used for the background for this shading. This color may either be presented as a hex value (in RRGGBB format), or `auto` to allow a consumer to automatically determine the background shading color as appropriate. If this attribute is omitted, then its value shall be assumed to be `auto`.

Attributes	Description
	[*Example*: Consider a shading using a background color of hex value C3D69B, using the following WordprocessingML: `<w:shd w:fill="C3D69B" />` The background color for this shading therefore will be a color with a hex value of C3D69B. *end example*] If the shading specifies the use of a theme color via the themeFill attribute, then this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).

themeColor (Shading Pattern Theme Color)	Specifies a theme color which should be applied to any foreground pattern specified for this shading using the val attribute.
	
	The specified theme color is a reference to one of the predefined theme colors, located in the document's themes part, which allows for color information to be set centrally in the document.
	
	If this element is omitted, then no theme color is applied, and the color attribute shall be used to determine the shading pattern color.
	
	[*Example*: Consider a paragraph which shall have a background consisting of a theme color accent3 with a theme color accent6 overlaid using a 20% fill pattern. This requirement is specified using the following WordprocessingML:
	
	`<w:pPr>`
	`  <w:shd w:val="pct20" w:themeColor="accent6"`
	`w:themeFill="accent3" />`
	`</w:pPr>`
	
	The resulting paragraph will use the foreground pattern color accent6 in the region specified by the pct20 pattern mask. *end example*]
	
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).

themeFill (Shading Background Theme Color)	Specifies a theme color which should be applied to the background for this shading. The specified theme color is a reference to one of the predefined theme colors, located in the document's themes part, which allows for color information to be set centrally in the document. If this element is omitted, then no theme color is applied, and the color attribute shall be

Attributes	Description
	used to determine the shading background color. [*Example*: Consider a paragraph which shall have a background consisting of a theme color `accent3` with a theme color `accent6` overlaid using a 20% fill pattern. This requirement is specified using the following WordprocessingML: `<w:shd w:val="pct20" w:themeColor="accent6" w:themeFill="accent3" />` The resulting shading will use the background color specified by the `accent3` theme color. *end example*] The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).
themeFillShade (Shading Background Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this shading color. If the themeFillShade is supplied, then it is applied to the RGB value of the themeFill color (from the theme part) to determine the final color applied to this border. The themeFillShade value is stored as a hex encoding of the shade value (from 0 to 255) applied to the current border. [*Example*: Consider a shade of 40% applied to a background shading color in a document. This shade is calculated as follows: $S_{xml} = 0.4 * 255$ $= 102$ $= 66(hex)$ The resulting themeFillShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `C0504D`. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$.

Attributes	Description
	Applying the shade formula with a shade percentage of 75% to the luminance, we get: $L^{'} = 0.53 * 0.75$ $= 0.39698$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting shading's fill attribute: `<w:shd w:fill="943634" w:themeFill="accent2" w:themeFillShade="BF" />` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeFillTint (Shading Background Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this shading instance. If the themeFillTint is supplied, then it is applied to the RGB value of the themeFill color (from the theme part) to determine the final color applied to this border. The themeFillTint value is stored as a hex encoding of the tint value (from 0 to 255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $T_{xml} = 0.6 * 255$ $= 153$ $= 99(hex)$ The resulting themeFillTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $\mathrm{L}^{'} = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color,

Attributes	Description
	whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $L^{'} = 0.53 * 0.6 + (1 - .6)$ $= 0.71$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting shading's fill attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:fill="95B3D7" w:themeFillColor="accent2" w:themeFillTint="99" />` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeShade (Shading Pattern Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this shading color. If the themeShade is supplied, then it is applied to the RGB value of the themeColor color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0 to 255) applied to the current border. [*Example*: Consider a shade of 40% applied to a background shading color in a document. This shade is calculated as follows: $S_{xml} = 0.4 * 255$ $= 102$ $= 66(hex)$ Te resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $\mathrm{L}^{'} = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$

Attributes	Description
	• Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `C0504D`. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $L' = 0.53 * 0.75$ $= 0.39698$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get `943634`. This transformed value can be seen in the resulting background's color attribute: `<w:shd w:color="943634" w:themeColor="accent2" w:themeShade="BF" />` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Shading Pattern Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this shading instance. If the themeTint is supplied, then it is applied to the RGB value of the themeColor color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0 to 255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $T_{xml} = 0.6 * 255$ $= 153$ $= 99(hex)$ The resulting themeTint value in the file format would be `99`. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1)

Attributes	Description
	• Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `4F81BD`. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6)$$ $$= 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get `95B3D7`. This transformed value can be seen in the resulting shading's color attribute: `<w:shd w:color="95B3D7" w:themeColor="accent2" w:themeTint="99" />` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Shading Pattern)	Specifies the pattern which shall be used to lay the pattern color over the background color for this paragraph shading. This pattern consists of a mask which is applied over the background shading color to get the locations where the pattern color should be shown. Each of these possible masks are shown in the simple type values referenced below. [*Example*: Consider a shaded paragraph which uses a 10 percent foreground fill, resulting in the following WordprocessingML: `<w:shd w:val="pct10" .../>` This shading val is `pct10`, indicating that the border style is a 10 percent foreground fill mask. *end example*] The possible values for this attribute are defined by the ST_Shd simple type (§2.18.85).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Shd">
   <attribute name="val" type="ST_Shd" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="fill" type="ST_HexColor" use="optional"/>
   <attribute name="themeFill" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeFillTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeFillShade" type="ST_UcharHexNumber" use="optional"/>
</complexType>
```

2.3.1.32 snapToGrid (Use Document Grid Settings for Inter-Line Paragraph Spacing)

This element specifies whether the current paragraph should use the document grid lines per page settings defined in the docGrid element (§2.6.5) when laying out the contents in the paragraph. This setting determines whether the additional line pitch specified in the document grid shall be added to each line in this paragraph as specified by the document grid.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then the paragraph shall use the document grid to lay out text when a document grid is defined for this document.

[*Example*: Consider two single-spaced paragraphs in a section with a document grid set to allow 15 lines per page. This document grid would effectively specifies that an additional line pitch of 45.6 points shall be added to each line in order to ensure that the resulting page contains only 15 lines of text.

If this property is set on the first paragraph, but turned off on the second paragraph, as follows:

```
<w:p>
  <w:pPr>
    <w:snapToGrid w:val="off" />
  </w:pPr>
  ...
</w:p>
<w:p>
  ...
</w:p>
```

The resulting document shall have 45.6 points of additional line pitch added to each line in paragraph two, but zero lines of additional line pitch added to each line in paragraph one, since the snapToGrid property is turned off. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.1.33 spacing (Spacing Between Lines and Above/Below Paragraph)

This element specifies the inter-line and inter-paragraph spacing which shall be applied to the contents of this paragraph when it is displayed by a consumer.

If this element is omitted on a given paragraph, each of its values is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then the paragraph shall have no spacing applied to its lines, or above and below its contents.

[*Example*: Consider the following WordprocessingML paragraph:

```
<w:pPr>
  <w:spacing w:after="200" w:line="276" w:lineRule="auto" />
</w:pPr>
```

This paragraph specifies that it shall have at least 200 twentieths of a point after the last line in each paragraph, and that the spacing in each line should be automatically calculated based on a 1.15 times (276 divided by 240) the normal single spacing calculation. *end example*]

When determining the spacing between any two paragraphs, a consumer shall use the maximum of the inter-line spacing in each paragraph, the spacing after the first paragraph and the spacing before the second paragraph to determine the net spacing between the paragraphs.

[*Example*: Consider two consecutive single-spaced paragraphs in a document, the first of which specifies spacing below of 12 points, the second of which specifies spacing above of 4 points. These constraints are expressed using the following WordprocessingML:

```
<w:p>
  <w:pPr>
    <w:spacing w:after="240" />
  </w:pPr>
  …
</w:p>
<w:p>
  <w:pPr>
    <w:spacing w:before="80" />
  </w:pPr>
  …
</w:p>
```

The resulting spacing between the first and second paragraph will be 12 points, since that is the largest spacing requested between the two paragraphs. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description

after (Spacing Below Paragraph)	Specifies the spacing that should be added after the last line in this paragraph in the document in absolute units.
	
	If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then the paragraph shall have no spacing applied below its contents.
	
	If the afterLines attribute or the afterAutoSpacing attribute is also specified, then this attribute value is ignored.
	
	[*Example*: Consider the following WordprocessingML paragraph:
	
	`<w:p>`
	`  <w:pPr>`
	`  <w:spacing w:after="240" />`
	`  </w:pPr>`
	`  …`
	`</w:p>`

Attributes	Description
	This paragraph shall have a minimum spacing below its final lines of 240 twentieths of a point, although the actual spacing may be determined by the inter-line spacing or the spacing above the following paragraph, if either are greater. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

afterAutospacing (Automatically Determine Spacing Below Paragraph)	Specifies whether a consumer shall automatically determine the spacing after this paragraph based on its contents.
	
	This automatic spacing shall match the spacing which would be applied to the paragraph in an HTML document where no explicit spacing before/after is specified.
	
	If this attribute is specified, then any value in after or afterLines is ignored, and the spacing is automatically determined by the consumer.
	
	If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then automatic spacing is turned off (not applied).
	
	[*Example*: Consider a paragraph in a document whose spacing below shall automatically be determined by the consumer based on the paragraph's contents. This constraint would be specified by the following WordprocessingML:
	
	`<w:pPr>`
	`  <w:spacing .. w:afterAutospacing="on" />`
	`</w:pPr>`
	
	The resulting paragraph shall have the spacing below its last line determined automatically by the consumer to match an HTML document as specified. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

afterLines (Spacing Below Paragraph in Line Units)	Specifies the spacing that should be added after the last line in this paragraph in the document in line units.
	
	The value of this attribute is specified in one hundredths of a line.
	
	If the afterAutoSpacing attribute is also specified, then this attribute value is ignored. If this setting is never specified in the style hierarchy, then its value shall be zero (if needed).
	
	[*Example*: Consider the following WordprocessingML paragraph:
	
	`<w:p>`
	`  <w:pPr>`
	`  <w:spacing w:afterLines="300" />`

Attributes	Description
	</w:pPr> ... </w:p> This paragraph shall have a minimum spacing below its final lines of 3 lines, although the actual spacing may be determined by the inter-line spacing or the spacing above the following paragraph, if either are greater. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
before (Spacing Above Paragraph)	Specifies the spacing that should be added above the first line in this paragraph in the document in absolute units. If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then the paragraph shall have no spacing applied above its contents. If the beforeLines attribute or the beforeAutoSpacing attribute is also specified, then this attribute value is ignored. [*Example*: Consider the following WordprocessingML paragraph: <w:p> <w:pPr> <w:spacing w:before="80" /> </w:pPr> ... </w:p> This paragraph shall have a minimum spacing above its first line of 80 twentieths of a point, although the actual spacing may be determined by the inter-line spacing or the spacing below the last line in the preceding paragraph, if either are greater. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).
beforeAutospacing (Automatically Determine Spacing Above Paragraph)	Specifies whether a consumer shall automatically determine the spacing before this paragraph based on its contents. This automatic spacing shall match the spacing which would be applied to the paragraph in an HTML document where no explicit spacing before/after is specified. If this attribute is specified, then any value in before or beforeLines is ignored, and the spacing is automatically determined by the consumer. If this element is omitted on a given paragraph, its value is determined by the setting

Attributes	Description

	previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then automatic spacing is turned off (not applied).
	
	[*Example*: Consider a paragraph in a document whose spacing above shall automatically be determined by the consumer based on the paragraph's contents. This constraint would be specified by the following WordprocessingML:
	
	`<w:pPr>`
	`  <w:spacing … w:beforeAutospacing="on" />`
	`</w:pPr>`
	
	The resulting paragraph shall have the spacing above its first line determined automatically by the consumer to match an HTML document as specified. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

beforeLines (Spacing Above Paragraph IN Line Units)	Specifies the spacing that should be added before the first line in this paragraph in the document in line units.
	
	The value of this attribute is specified in one hundredths of a line.
	
	If the beforeAutoSpacing attribute is also specified, then this attribute value is ignored. If this setting is never specified in the style hierarchy, then its value shall be zero (if needed).
	
	[*Example*: Consider the following WordprocessingML paragraph:
	
	`<w:p>`
	`  <w:pPr>`
	`  <w:spacing w:beforeLines="100" />`
	`  </w:pPr>`
	`  ...`
	`</w:p>`
	
	This paragraph shall have a minimum spacing above its first line of 1 line, although the actual spacing may be determined by the inter-line spacing or the spacing below the preceding paragraph, if either are greater. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

line (Spacing Between Lines in Paragraph)	This attribute specifies the amount of vertical spacing between lines of text within this paragraph. If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then no line spacing

Attributes	Description

	shall be applied to lines within this paragraph.
	
	If the value of the lineRule attribute is either `atLeast` or `exactly`, then the value of this attribute shall be interpreted as twentieths of a point. When the value of the lineRule attribute is either `exactly`, the text shall be positioned as follows within that line height:
	• When the line height is too small, the text shall be positioned at the bottom of the line (i.e. clipped from the top down)
	• When the line height is too large, the text shall be centered in the available space.
	
	If the value of the lineRule attribute is `auto`, then the value of the line attribute shall be interpreted as 240ths of a line, in the manner described by the simple type's values.
	
	[Example: Consider the following WordprocessingML paragraph which should have an inter-line spacing of 1.15 times the line height. This constraint would be specified using the following WordprocessingML:
	
	`<w:pPr>`
	`  <w:spacing w:line="276" w:lineRule="auto" />`
	`</w:pPr>`
	
	The lineRule attribute value of `auto` specifies that the value of the line attribute is to be interpreted in 240ths of a single line height, which means that the net spacing is 276/240ths of a line or 1.15 lines tall. *end example*]
	
	The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§2.18.88).

lineRule (Type of Spacing Between Lines)	Specifies how the spacing between lines is calculated as stored in the line attribute.
	
	If this attribute is omitted, then it shall be assumed to be of a value `auto` if a line attribute value is present.
	
	If the value of this attribute is either `atLeast` or `exactly`, then the value of the line attribute shall be interpreted as twentieths of a point, in the manner described by the simple type's values.
	
	If the value of this attribute is `auto`, then the value of the line attribute shall be interpreted as 240ths of a line, in the manner described by the simple type's values.
	
	[*Example*: Consider the following WordprocessingML paragraph which should have an inter-line spacing of 1.15 times the line height. This constraint would be specified using the following WordprocessingML:
	
	`<w:pPr>`
	`  <w:spacing w:line="276" w:lineRule="auto" />`

Attributes	Description
	</w:pPr> The lineRule attribute value of auto specifies that the value of the line attribute is to be interpreted in 240ths of a single line height. *end example*] The possible values for this attribute are defined by the ST_LineSpacingRule simple type (§2.18.55).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Spacing">
    <attribute name="before" type="ST_TwipsMeasure" use="optional"/>
    <attribute name="beforeLines" type="ST_DecimalNumber" use="optional"/>
    <attribute name="beforeAutospacing" type="ST_OnOff" use="optional"/>
    <attribute name="after" type="ST_TwipsMeasure" use="optional"/>
    <attribute name="afterLines" type="ST_DecimalNumber" use="optional"/>
    <attribute name="afterAutospacing" type="ST_OnOff" use="optional"/>
    <attribute name="line" type="ST_SignedTwipsMeasure" use="optional"/>
    <attribute name="lineRule" type="ST_LineSpacingRule" use="optional"/>
</complexType>
```

2.3.1.34 suppressAutoHyphens (Suppress Hyphenation for Paragraph)

This element specifies whether any hyphenation shall be performed on this paragraph by the consumer when requested using the autoHyphenation element (§2.15.1.10) in the document's settings. This element specifies whether the current paragraph should be exempted from any hyphenation which is applied by the consumer on this document.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then the default hyphenation settings for the document, as specified in the autoHyphenation element, shall apply to the contents of this paragraph.

[*Example*: Consider a document which shall be hyphenated automatically by a consumer, since it has the autoHyphenation element set to true in its document settings. If this paragraph should be exempted from that hyphenation pass, this requirement would be specified using the following WordprocessingML:

```
<w:pPr>
  <w:suppressAutoHyphens />
</w:pPr>
```

The paragraph would then be exempted from hyphenation by a consumer at display time, regardless of the hyphenation settings for the document. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.1.35 suppressLineNumbers (Suppress Line Numbers for Paragraph)

This element specifies whether line numbers shall be calculated for lines in this paragraph by the consumer when line numbering is requested using the lnNumType element (§2.6.8) in the paragraph's parent section settings. This element specifies whether the current paragraph's lines should be exempted from line numbering which is applied by the consumer on this document, not just suppressing the display of the numbering, but removing these lines from the line numbering calculation.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then the default line number settings for the section, as specified in the lnNumType element shall apply to each line of this paragraph.

[*Example*: Consider a document with three paragraphs, each of which are displayed on five lines , all contained in a section which has the lnNumType element specified. If the second paragraph should be exempted from that line numbering, this requirement would be specified using the following WordprocessingML:

```
<w:pPr>
  <w:suppressLineNumbers />
</w:pPr>
```

The paragraph would then be exempted from line by a consumer at display time, which would result in paragraph one using line numbers one through five, the second paragraph having no line numbers, and the third paragraph using line numbers six through ten. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.1.36 suppressOverlap (Prevent Text Frames From Overlapping)

This element specifies whether a text frame which intersects another text frame at display time shall be allowed to overlap the contents of the other text frame. If a text frame cannot overlap other text frames, it shall be repositioned when displayed to prevent this overlap as needed.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then overlap shall be allowed between a text frame which intersects another text frame displayed at the same location.

[*Example*: Consider a document with two text frames which are allowed to overlap each other. If the second text frame should overlap the contents of another text frame, that constraint would be specified via the following WordprocessingML:

```
<w:p>
  …
</w:p>
<w:p>
```

```
    <w:pPr>
      <w:framePr … />
      <w:suppressOverlap />
    </w:pPr>
    …
  </w:p>
```

The resulting text frame with the suppressOverlap property specified would never overlap any intersecting text frames. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.1.37 tab (Custom Tab Stop)

This element specifies a single custom tab stop within a set of custom tab stops applied as part of a set of customized paragraph properties in a document.

[*Example*: Consider a custom tab stops at 1.5" in a WordprocessingML document. This tab stop would be contained within a tab element defining the tab stop as follows:

```
<w:tab w:val="left" w:pos="2160" />
```

The tab element specifies all of the properties for the customized tab stop for the current paragraph property set. *end example*]

Parent Elements
tabs (§2.3.1.38)

Attributes	Description
leader (Tab Leader Character)	Specifies the character which shall be used to fill in the space created by a tab which ends at this custom tab stop. This character shall be repeated as required to completely fill the tab spacing generated by the tab character. If this attribute is omitted, then no tab leader character shall be used. [*Example*: Consider a tab stop which should be preceded by a sequence of underscore characters, as follows: _____Text at the tab stop This tab stop would have a leader attribute value of `underscore`, indicating that the tab stop shall be preceded by underscore characters as needed to fill the tab spacing. *end example*] The possible values for this attribute are defined by the ST_TabTlc simple type (§2.18.92).
pos (Tab Stop Position)	Specifies the position of the current custom tab stop with respect to the current page margins. Negative values are valid and move the tab stop into the current page margin the specified amount. [*Example*: Consider a custom tab stops at 1.5" in a WordprocessingML document. This tab stop would be contained within a tab element defining the tab stop as follows: `<w:tab w:val="left" w:pos="2160" />` The pos attribute specifies that this custom tab stop shall be located `2160` points (1.5 inches) inside the starting text margin. *end example*] The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§2.18.88).
val (Tab Stop Type)	Specifies the type of custom tab stop, which determines the behavior of the tab stop and the alignment which shall be applied to text entered at the current custom tab stop. The value of `clear` is unique and specifies that this tab stop shall be removed when the document is next edited by a consumer which supports rendering the document

Attributes	Description
	contents. [*Example*: Consider a custom tab stops at 1.5" in a WordprocessingML document. This tab stop would be contained within a tab element defining the tab stop as follows: `<w:tab w:val="left" w:pos="2160" />` The val attribute specifies that this custom tab stop shall align all text entered at its location to its left. *end example*] The possible values for this attribute are defined by the ST_TabJc simple type (§2.18.91).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TabStop">
   <attribute name="val" type="ST_TabJc" use="required"/>
   <attribute name="leader" type="ST_TabTlc" use="optional"/>
   <attribute name="pos" type="ST_SignedTwipsMeasure" use="required"/>
</complexType>
```

2.3.1.38 tabs (Set of Custom Tab Stops)

This element specifies a sequence of custom tab stops which shall be used for any tab characters in the current paragraph.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then no custom tab stops shall be used for this paragraph.

As well, this property is additive - tab stops at each level in the style hierarchy are added to each other to determine the full set of tab stops for the paragraph. A hanging indent specified via the hanging attribute on the ind element (§2.3.1.12) shall also always implicitly create a custom tab stop at its location.

[*Example*: Consider a paragraph which contains two custom tab stops at 1.5" and 3.5", respectively. These two tab stops would be contained within a tabs element defining the set of tab stops of the paragraph as follows:

```
<w:pPr>
  <w:tabs>
    <w:tab w:val="left" w:pos="2160" />
    <w:tab w:val="left" w:pos="5040" />
  </w:tabs>
</w:pPr>
```

The tabs element specifies all of the customized tab stops for the current paragraph. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Child Elements	Subclause
tab (Custom Tab Stop)	§2.3.1.37

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Tabs">
   <sequence>
      <element name="tab" type="CT_TabStop" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

2.3.1.39 textAlignment (Vertical Character Alignment on Line)

This element specifies the vertical alignment of all text on each line displayed within a paragraph. If the line height (before any added spacing) is larger than one or more characters on the line, all characters will be aligned to each other as specified by this element.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then the vertical alignment of all characters on the line shall be automatically determined by the consumer.

[*Example*: Consider a paragraph of text of different font sizes, as follows:

If the text on this paragraph shall be aligned based on the top point of the maximum character height, that requirement would be specified as follows in the WordprocessingML:

```
<w:pPr>
  <w:textAlignment w:val="top" />
</w:pPr>
```

The resulting text would be top aligned, as follows:

The characters are all aligned to the maximum character extent on the line. *end example*]

Parent Elements

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description

val (Vertical Character Alignment Position)	Specifies the type of vertical alignment which shall be used to align the characters on each line in the current paragraph.
	
	[*Example*: Consider a paragraph of text of different font sizes which shall be aligned based on the baseline point of each character in each line. This requirement would be specified as follows in the WordprocessingML:
	
	`<w:pPr>`
	`  <w:textAlignment w:val="baseLine" />`
	`</w:pPr>`
	
	The resulting text would be aligned to the baseline for each character on the line. *end example*]
	
	The possible values for this attribute are defined by the ST_TextAlignment simple type (§2.18.98).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextAlignment">
   <attribute name="val" type="ST_TextAlignment" use="required"/>
</complexType>
```

2.3.1.40 textboxTightWrap (Allow Surrounding Paragraphs to Tight Wrap to Text Box Contents)

This element specifies whether, for paragraphs in a text box, the surrounding text shall be allowed to overlap with the empty text box boundaries and tight wrap to the extents of the text within the text box.

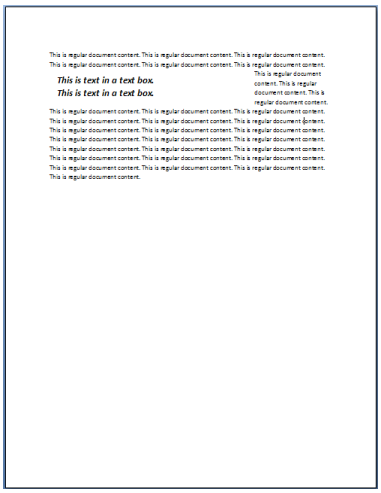
This element shall only be read for paragraphs which are contained within a text box (have a txbxContent ancestor), ignored otherwise.

If the parent text box does not meet the following three criteria, then this property has no effect:

- The text box wrapping must be set to `tight`
- The text box border must not be set
- The text box shading must not be set

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then paragraphs in a text box have no tight wrapping overrides, and text shall wrap to the extents of the text box.

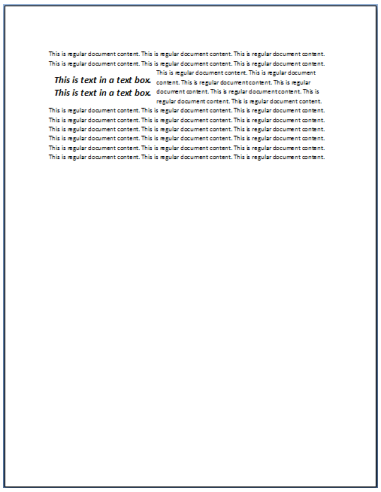
[*Example*: Consider a document with a tight wrapped text box which extends two-thirds of the way across the page, as follows:



The surrounding text is tightly wrapped to the extents of the text box. If the consumer shall tight wrap to the extents of the text, that requirement would be specified using the following WordprocessingML:

```
<w:pPr>
  <w:textboxTightWrap w:val="all" />
</w:pPr>
```

This would result in the following display of the content:



The resulting paragraphs within the textbox use the textboxTightWrap element to specify that text should be tightly wrapped to the paragraph's extents. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description

val (Lines to Tight Wrap to Paragraph Extents)	Specifies the lines in the parent paragraph which shall allow the text to be tight wrapped to the paragraph (and not the text box) extents when displaying the document.
	
	[*Example*: Consider a paragraph in a text box which meets the criteria specified above which shall allow wrapping to the text extents on its first line only. That requirement would be specified using the following WordprocessingML:
	
	`<w:pPr>`
	`  <w:textboxTightWrap w:val="firstLineOnly" />`
	`</w:pPr>`
	
	The resulting paragraph would allow text to tightly wrap to the contents of its first line only. All other lines would wrap to the text box's extents. *end example*]
	
	The possible values for this attribute are defined by the ST_TextboxTightWrap simple type (§2.18.99).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextboxTightWrap">
   <attribute name="val" type="ST_TextboxTightWrap" use="required"/>
</complexType>
```

2.3.1.41 textDirection (Paragraph Text Flow Direction)

This element specifies the direction of the text flow for this paragraph.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then the paragraph shall inherit the text flow settings from the parent section.

[*Example*: Consider a document with a paragraph in which text should flow bottom to top vertically, and left to right horizontally. This setting would be specified with the following WordprocessingML:

```
<w:pPr>
  <w:textFlow w:val="btLr" />
</w:pPr>
```

The textFlow element specifies via the `btLr` value in the val attribute that the text flow should go bottom to top, and left to right. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description

Attributes	Description

val (Direction of Text Flow)	Specifies the direction of the text flow for this object.
	
	[*Example*: Consider a document with a section in which text shall flow bottom to top vertically, and left to right horizontally. This setting requires the following WordprocessingML:
	
	`<w:sectPr>`
	`…`
	`<w:textDirection w:val="btLr" />`
	`</w:sectPr>`
	
	The textDirection element specifies via the `btLr` value in the val attribute that the text flow shall go bottom to top, and left to right. *end example*]
	
	The possible values for this attribute are defined by the ST_TextDirection simple type (§2.18.100).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextDirection">
   <attribute name="val" type="ST_TextDirection" use="required"/>
</complexType>
```

2.3.1.42 top (Paragraph Border Above Identical Paragraphs)

This element specifies the border which shall be displayed above a set of paragraphs which have the same set of paragraph border settings.

To determine if any two adjoining paragraphs shall have an individual top and bottom border or a between border, the set of borders on the two adjoining paragraphs are compared. If the border information on those two paragraphs is identical for all possible paragraphs borders, then the between border is displayed. Otherwise, the final paragraph shall use its bottom border and the following paragraph shall use its top border, respectively. If this border specifies a space attribute, that value determines the space above the text (ignoring any spacing above) which should be left before this border is drawn, specified in points.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then no between border shall be applied above identical paragraphs.

[*Example*: Consider the following two paragraphs' WordprocessingML definition:

```
<w:p>
  <w:pPr>
    <w:pBdr>
      <w:top w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
```

```
      <w:left w:val="single" w:sz="24" w:space="4" w:color="B97034"
w:themeColor="accent6" w:themeShade="BF" />
      <w:bottom w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:right w:val="single" w:sz="24" w:space="4" w:color="C3D69B"
w:themeColor="accent3" w:themeTint="99" />
      <w:between w:val="single" w:sz="24" w:space="1" w:color="4F81BD"
w:themeColor="accent1" />
    </w:pBdr>
  </w:pPr>
  <w:r>
    <w:t>First paragraph.</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:pBdr>
      <w:top w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="B97034"
w:themeColor="accent6" w:themeShade="BF" />
      <w:bottom w:val="single" w:sz="24" w:space="0" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:right w:val="single" w:sz="24" w:space="4" w:color="C3D69B"
w:themeColor="accent3" w:themeTint="99" />
      <w:between w:val="single" w:sz="24" w:space="1" w:color="4F81BD"
w:themeColor="accent1" />
    </w:pBdr>
  </w:pPr>
  <w:r>
    <w:t>Second paragraph.</w:t>
  </w:r>
</w:p>
```

Since the paragraph border is different between the two paragraphs (the bottom space value goes from 1 to 0), paragraph two uses its top border, which is located one point above the text in that paragraph. *end example*]

Parent Elements
pBdr (§2.3.1.24)

Attributes	Description
color (Border Color)	Specifies the color for this border.

Attributes	Description
	This color may either be presented as a hex value (in RRGGBB format), or auto to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value auto, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is true, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />`

Attributes	Description
	This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object.
	
	When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points.
	
	When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.
	
	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24/>`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*]
	
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).

sz (Border Width)	Specifies the width of the current border.
	
	If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value.
	
	If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value.
	
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`

Attributes	Description
	`<w:right w:val="dashed" w:sz="24" …/>` The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*] The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).

themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border.
	
	The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document.
	
	[*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup:
	
	`<w:top … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />`
	`<w:bottom … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />`
	`<w:left … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />`
	`<w:right … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />`
	
	The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*]
	
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).

themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255 = 102$$

Attributes	Description
	$= 66(hex)$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $\mathrm{L}^{'} = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $L^{'} = 0.53 * 0.75$ $= 0.39698$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated

Attributes	Description
	as follows: $$T_{xml} = 0.6 * 255 \\ = 153 \\ = 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6) \\ = 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style.

Attributes	Description
	[*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
    <attribute name="val" type="ST_Border" use="required"/>
    <attribute name="color" type="ST_HexColor" use="optional"/>
    <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
    <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
    <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
    <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
    <attribute name="space" type="ST_PointMeasure" use="optional"/>
    <attribute name="shadow" type="ST_OnOff" use="optional"/>
    <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.3.1.43 topLinePunct (Compress Punctuation at Start of a Line)

This element specifies whether punctuation shall be compressed when it appears as the first character in a line, allowing subsequent characters on the line to be move in accordingly.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then punctuation shall not be compressed in this paragraph, even when it appears at the start of a line.

[*Example*: Consider a paragraph which should allow punctuation at the start of a line to be compressed, in order to prevent it from taking up unnecessary space. This constraint is specified using the following WordprocessingML:

```
<w:pPr>
  <w:topLinePunct w:val="on" />
</w:pPr>
```

The topLinePunct element specifies that this compression shall be allowed when displaying this paragraph. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

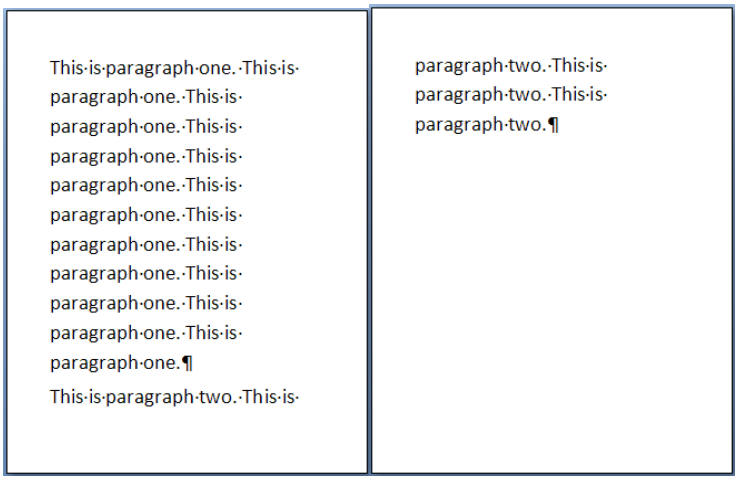
2.3.1.44 widowControl (Allow First/Last Line to Display on a Separate Page)

This element specifies whether a consumer shall prevent a single line of this paragraph from being displayed on a separate page from the remaining content at display time by moving the line onto the following page.

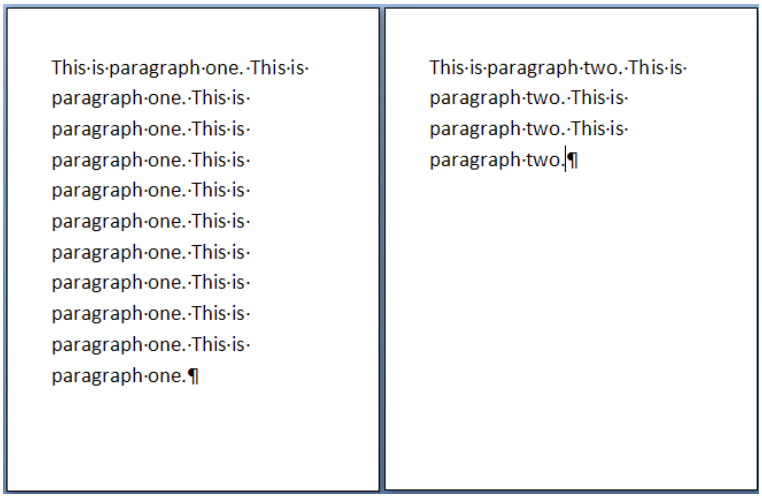
When displaying a paragraph in a page, it is sometimes the case that the first line of that paragraph would display as the last line on one page, and all subsequent lines would display on the following page. This property ensures that a consumer shall move the single line to the following page as well to prevent having one line on its own page. As well, if a single line appears at the top of a page, a consumer shall move the preceding line onto the following page as well, to prevent a single line from being displayed on a separate page.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then this paragraph shall prevent a single line from being shown on a separate page whenever it would normally occur.

[*Example*: Consider a document with a paragraph which shall be shown on four lines at display time. If this paragraph would normally be laid out with its first line at the bottom of one page, and its following lines on the next page, as follows:



This property would ensure that the default behavior for each paragraph prevented this, by moving this line onto the following paragraph as follows:



However, if this default is overridden by specifying the following WordprocessingML:

```
<w:pPr>
  <w:widowControl w:val="off" />
</w:pPr>
```

The specifying of the widowControl element with value `off` means that the consumer displaying this document shall not move the first line onto a separate page if it would be separated from all other lines (the first picture above). *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.1.45 wordWrap (Allow Line Breaking At Character Level)

This element specifies whether a consumer shall break Latin text which exceeds the text extents of a line by breaking the word across two lines (breaking on the character level) or by moving the word to the following line (breaking on the word level).

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then this paragraph shall break Latin words at the word level, not the character level when it is displayed.

[*Example*: Consider a paragraph whose first line ends with the word world, where the text extents for that line would normally fall between the letter o and the letter r. If this element is omitted, a producer would normally move the entire word world to the following line, since the word does not fit within the first line's text extents. However, if this document should allow words to be broken at the character level, that constraint would be specified as follows:

```
<w:pPr>
  <w:wordWrap w:val="off" />
</w:pPr>
```

The resulting paragraph specifies that wordWrap is turned off, therefore the word "world" would be broken into two lines between the exact two characters (o and r) that match the text extents. *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.3.1.25); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.2 Run

The next level of the document hierarchy is the *run*, which defines a region of text with a common set of properties, represented by the r element (§2.3.2.23). An r element allows the producer to specify a single set of formatting properties, applying the same information to all the contents of the run.

Just as a paragraph can have properties, so too can a run. All of the elements inside an r element have their properties controlled by a corresponding optional rPr run properties element (§2.7.8.1; §2.3.2.26), which must be the first child of the r element. In turn, the rPr element is a container for a set of property elements that are applied to the rest of the children of the r element. [*Note*: The elements inside the rPr container element allow the consumer to control whether the content in the following run content is bold, underlined, or visible, for example. *end note*]

[*Example*: Consider the following run within a WordprocessingML document:

```
<w:r>
  <w:rPr>
    <w:b/>
    <w:i/>
  </w:rPr>
  <w:t>quick</w:t>
</w:r>
```

The run specifies two formatting properties in its run contents: bold and italic. These properties are therefore applied to all content within this run. *end example*]

2.3.2.1 b (Bold)

This element specifies whether the bold property shall be applied to all non-complex script characters in the contents of this run when displayed in a document.

This formatting property is a *toggle property*, which specifies that its behavior differs between its use within a style definition and its use as direct formatting. When used as part of a style definition, setting this property shall toggle the current state of that property as specified up to this point in the hierarchy (i.e. applied to not applied, and vice versa). Setting it to `false` (or an equivalent) shall result in the current setting remaining unchanged. However, when used as direct formatting, setting this property to `true` or `false` shall set the absolute state of the resulting property.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then bold shall not be applied to non-complex script characters.

[*Example*: Consider a run of text which shall have the b property explicitly turned off for the non complex script contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:b w:val="false"/>
</w:rPr>
```

This run explicitly declares that the b property is `false` for the non-complex script contents of this run. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element.

Attributes	Description
	A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.2.2 bCs (Complex Script Bold)

This element specifies whether the bold property shall be applied to all complex script characters in the contents of this run when displayed in a document.

This formatting property is a *toggle property*, which specifies that its behavior differs between its use within a style definition and its use as direct formatting. When used as part of a style definition, setting this property shall toggle the current state of that property as specified up to this point in the hierarchy (i.e. applied to not applied, and vice versa). Setting it to false (or an equivalent) shall result in the current setting remaining unchanged. However, when used as direct formatting, setting this property to true or false shall set the absolute state of the resulting property.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then bold shall not be applied to complex script characters.

[*Example*: Consider a run of text which shall have the bCs property (bold) explicitly turned on for the complex script contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:bCs w:val="true"/>
</w:rPr>
```

This run explicitly declares that the bCs property is true , so bold is turned on for the complex script contents of this run. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.2.3 bdr (Text Border)

This element specifies information about the border applied to the text in the current run.

The first piece of information specified by the bdr element is that the current shall have a border when displayed. This information is specified simply by the presence of the bdr element in run's properties.

The second piece of information concerns the set of runs which share the current run border. This is determined based on the attributes on the bdr element. If the set of attribute values specifies on two adjacent runs is identical, then those two runs shall be considered to be part of the same run border group and rendered within the same set of borders in the document.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then no run border shall be applied to the text in this run.

[*Example*: Consider a document in which the following two runs are located adjacent to one another:

```
<w:r>
  <w:rPr>
```

```
    <w:bdr w:val="single" w:sz="36" w:space="0" w:color="B8CCE4"
w:themeColor="accent1" w:themeTint="66" />
  </w:rPr>
  <w:t xml:space="preserve">run one</w:t>
</w:r>
<w:r >
  <w:rPr>
    <w:b />
    <w:bdr w:val="single" w:sz="36" w:space="0" w:color="B8CCE4"
w:themeColor="accent1" w:themeTint="66" />
  </w:rPr>
  <w:t>run two</w:t>
</w:r>
```

These two runs, although each is distinct, are combined when rendering the text border because the bdr elements are identical between the two runs. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or auto to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value auto, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text.

Attributes	Description
	If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is `true`, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object.
	
	When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points.
	
	When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.
	
	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24/>`

Attributes	Description
	</w:pgBorders The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*] The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).
sz (Border Width)	Specifies the width of the current border. If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value. If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value. [*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup: <w:top w:val="dashed" w:sz="24" …/> <w:left w:val="dashed" w:sz="24" …/> <w:bottom w:val="dashed" w:sz="24" …/> <w:right w:val="dashed" w:sz="24" …/> The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*] The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).
themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border. The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document. [*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup: <w:top … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:bottom … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />

Attributes	Description

	`<w:left … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />`
	`<w:right … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />`
	
	The borders have a color with an RGB value of `FFA8A0`, however, because the themeColor attribute is specified, that value is ignored in favor of the `accent2` theme color specified for this document. *end example*]
	
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).

themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255 = 102 = 66(hex)$$ The resulting themeShade value in the file format would be `66`. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `C0504D`. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L' = 0.53 * 0.75$$

Attributes	Description
	$= 0.39698$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $T_{xml} = 0.6 * 255$ $= 153$ $= 99(hex)$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD.

Attributes	Description
	The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $L' = 0.53 * 0.6 + (1 - .6)$ $= 0.71$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is single, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <attribute name="val" type="ST_Border" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
   <attribute name="space" type="ST_PointMeasure" use="optional"/>
   <attribute name="shadow" type="ST_OnOff" use="optional"/>
   <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.3.2.4 caps (Display All Characters As Capital Letters)

This element specifies that any lowercase characters in this text run shall be formatted for display only as their capital letter character equivalents. This property does not affect any non-alphabetic character in this run, and does not change the Unicode character for lowercase text, only the method in which it is displayed.

This formatting property is a *toggle property*, which specifies that its behavior differs between its use within a style definition and its use as direct formatting. When used as part of a style definition, setting this property shall toggle the current state of that property as specified up to this point in the hierarchy (i.e. applied to not applied, and vice versa). Setting it to `false` (or an equivalent) shall result in the current setting remaining unchanged. However, when used as direct formatting, setting this property to `true` or `false` shall set the absolute state of the resulting property.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the characters are not formatted as capital letters.

This element shall not be present with the smallCaps (§2.3.2.31) property on the same run, since they are mutually exclusive in terms of appearance.

[*Example*: Consider the words `Hello World`, which shall be displayed in all capital letters in a document. This constraint is specified as follows in the WordprocessingML:

```
<w:r>
  <w:rPr>
    <w:caps w:val="true" />
  </w:rPr>
  <w:t>Hello World</w:t>
</w:r>
```

This run will display as `HELLO WORLD`, even though the lowercase characters are used in the run contents due to the use of the caps element. If this property is removed, the original character forms will be displayed (they are not lost). *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted.

Attributes	Description
	A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.2.5 color (Run Content Color)

This element specifies the color which shall be used to display the contents of this run in the document.

This color may be explicitly specified, or set to allow the consumer to automatically choose an appropriate color based on the background color behind the run's content.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the characters are set to allow the consumer to automatically choose an appropriate color based on the background color behind the run's content.

[*Example*: Consider a run of text which should be displayed using the `accent3` theme color from the document's Theme part. This requirement would be specified as follows in the resulting WordprocessingML:

```
<w:rPr>
  <w:color w:themeColor=”accent3” />
</w:rPr>
```

The color attribute specifies that the run shall use the `accent3` theme color. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
themeColor (Run Content Theme	Specifies a theme color which should be applied to the current run.

Attributes	Description

Color)	The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows for color information to be set centrally in the document.
	
	If the themeColor attribute is specified, then the val attribute is ignored for this run.
	
	[*Example*: Consider a run of text which should be displayed using the accent3 theme color from the document's Theme part. This requirement would be specified as follows in the resulting WordprocessingML:
	
	`<w:rPr>`
	`  <w:color w:themeColor="accent3" />`
	`</w:rPr>`
	
	The color attribute specifies that the run shall use the accent3 theme color. *end example*]
	
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).

themeShade (Run Content Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this run's contents. If the themeShade is supplied, then it is applied to the RGB value of the theme color to determine the final color applied to this run. The themeShade value is stored as a hex encoding of the shade value (from 0 to 255) applied to the current border. [*Example*: Consider a shade of 40% applied to a run in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255$$ $$= 102$$ $$= 66(hex)$$ Te resulting themeShade value in the file format would be 66. *end example*] Given a input red, green, or blue color value C (from 0-255), an output color value of C' (from 0-255), and a shade value S (from 0-100), the shade is applied as follows: $$C' = \left(1 - \frac{S}{100}\right) \mathrm{C}$$ [*Example*: Consider a document with a run using the accent6 theme color, whose RGB value (in RRGGBB hex format) is F79646.

Attributes	Description
	The hex value for the green component is 96 - 150 in decimal. Applying the shade formula with shade of 50%, the output decimal value of the green component is 75, or a hex value of 4B. This transformed value can be seen in the resulting run color WordprocessingML's val attribute: `<w:color w:val="7B4B23" w:themeColor="accent6" w:themeShade="80" />` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Run Content Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this run's contents. If the themeTint is supplied, then it is applied to the RGB value of the theme color to determine the final color applied to this run. The themeTint value is stored as a hex encoding of the tint value (from 0 to 255) applied to the current border. [*Example*: Consider a tint of 60% applied to a run in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255 = 153 = 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given a input red, green, or blue color value C (from 0-255), an output color value of C' (from 0-255), and a tint value T (from 0-100), the tint is applied as follows: $$C' = \left(1 - \frac{T}{100}\right)(255 - \mathrm{C}) + \mathrm{C}$$ [*Example*: Consider a document with a run using the accent1 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The hex value for the green component is 50 - 80 in decimal. Applying the tint formula with tint of 60%, the output decimal value of the green component is 150, or a hex value of 96. This transformed value can be seen in the resulting run color's WordprocessingML val attribute: `<w:color w:val="D99694" w:themeColor="accent1" w:themeTint="99" />`

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).

val (Run Content Color)	Specifies the color for this run.
	
	This color may either be presented as a hex value (in RRGGBB format), or `auto` to allow a consumer to automatically determine the run color as appropriate.
	
	If the run specifies the use of a theme color via the themeColor attribute, then this value is superseded by the theme color value.
	
	[*Example*: Consider a run color with value `auto`, as follows:
	
	`<w:rPr>`
	`  <w:color ... w:val="auto" />`
	`</w:rPr>`
	
	This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the run contents can be distinguished against the page's background color. *end example*]
	
	The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <attribute name="val" type="ST_HexColor" use="required"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
</complexType>
```

2.3.2.6 cs (Use Complex Script Formatting on Run)

This element specifies whether the contents of this run shall be treated as complex script text regardless of their Unicode character values when determining the formatting for this run.

This means that a consumer shall use the complex script formatting applied to the run [*Example*: The bCs value (§2.3.2.2), not the b value (§2.3.2.1). *end example*] when determining the resulting formatting properties.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the run contents are set to complex script based on the Unicode character positions of the content.

[*Example*: Consider the following run of English text in a WordprocessingML document:

```
<w:r>
  <w:rPr>
    <w:bCs/>
    <w:i/>
    <w:cs/>
  </w:rPr>
  <w:t>some English text</w:t>
</w:r>
```

This run has bold applied to complex script characters, and italics applied to non-complex script characters. However, since the cs property is set, the text in this run shall be treated as complex script text when determining the resulting formatting. Therefore, the run will have bold formatting, but no italic formatting when displayed. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.2.7 dstrike (Double Strikethrough)

This element specifies that the contents of this run shall be displayed with two horizontal lines through each character displayed on the line.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then double strikethrough shall not be applied to the contents of this run.

This element shall not be present with the strike (§2.3.2.35) property on the same run, since they are mutually exclusive in terms of appearance.

[*Example*: Consider a run of text which shall have the dstrike property explicitly turned on for the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:dstrike w:val="true"/>
</w:rPr>
```

This run explicitly declares that the dstrike property is `true`, so the contents of this run will have two horizontal strikethrough lines. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.2.8 eastAsianLayout (East Asian Typography Settings)

This element specifies any East Asian typography settings which shall be applied to the contents of the run.

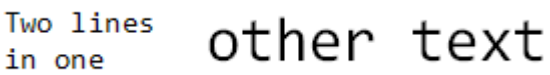
The specific typography settings represented by this element include the *two lines in one* and *horizontal in vertical* text options.

The *two lines in one* setting specifies that the characters in this run should be written out on a single line in the document by creating two sub-lines within the regular line, and laying out this text equally between those sub lines.

[*Example*: Consider a paragraph with the text `two lines in one`, which shall be displayed within a single logical line in the document. This constraint would be specified as follows in the WordprocessingML:

```
<w:r>
  <w:rPr>
    <w:eastAsianLayout w:id="1" w:combine="on" />
  </w:rPr>
  <w:t>two lines in one</w:t>
</w:r>
```

The resulting text would be displayed on two sub lines within the other text on this line, like this:



end example]

The *horizontal in vertical* setting specifies that characters in this run should be rendered with a 90 degree rotation to the left from all other contents of the line when displayed in the document, while keeping the text on the same line as all other text in the paragraph.

[*Example*: Consider a paragraph with the text `this word is vertical`, of which the word `vertical` shall be displayed vertically within the document. This constraint would be specified as follows in the WordprocessingML:

```
<w:r>
  <w:rPr>
    <w:eastAsianLayout w:id="2" w:vert="on" />
  </w:rPr>
  <w:t>vertical</w:t>
</w:r>
```

The resulting text would be displayed with a 90 degree rotation from the other text content. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description

combine (Two Lines in One)	Specifies whether the contents of the current run should be combined into one line using the two lines in one logic described above in the parent element.
	
	If this attribute is omitted, then this run shall not be displayed on two sub lines.
	
	[*Example*: Consider a paragraph with the text `two lines in one`, which shall be displayed within a single logical line in the document. This constraint would be specified as follows in the WordprocessingML:
	
	`<w:r>`
	`  <w:rPr>`
	`  <w:eastAsianLayout w:id="1" w:combine="on" />`
	`  </w:rPr>`
	`  <w:t>two lines in one</w:t>`
	`</w:r>`
	
	The resulting text would be displayed on two sub lines within the other text on this line. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

combineBrackets (Display Brackets Around Two Lines in One)	Specifies that the two lines in one text should be enclosed within a pair of brackets when displayed. This attribute's values determine the bracket style to put around combined text.
	
	If this attribute is not specified, then no brackets shall be placed around this content when displayed in the document. If the combine attribute is not specified, then this attribute is ignored.
	
	[*Example*: Consider a paragraph with the text `two lines in one`, which shall be displayed within a single logical line in the document and enclosed in curly brackets. This constraint would be specified as follows in the WordprocessingML:
	
	`<w:r>`
	`  <w:rPr>`
	`  <w:eastAsianLayout w:id="1" w:combine="on"`
	`w:combineBrackets="curly"/>`
	`  </w:rPr>`
	`  <w:t>two lines in one</w:t>`
	`</w:r>`
	
	The resulting text would be displayed on two sub lines within the other text on this line and enclosed within curly brackets when displayed. *end example*]
	
	The possible values for this attribute are defined by the ST_CombineBrackets simple type (§2.18.13).

id (East Asian	Specifies a unique ID which shall b used to link multiple runs containing eastAsianLayout

Attributes	Description

Typography Run ID)	element to each other to ensure that their contents are correctly displayed in the document.
	
	This means that multiple runs which are broken apart due to differences in formatting can be identified as belonging to the same grouping in terms of eastAsianLayout properties, although they are separated into multiple runs of text.
	
	[*Example*: Consider the following three runs in a document:
	
	`<w:r>`
	`  <w:rPr>`
	`  <w:asianLayout w:id="-1552701694" w:combine="lines" w:combineBrackets="curly" />`
	`  </w:rPr>`
	`  <w:t>two</w:t>`
	`</w:r>`
	`<w:r>`
	`  <w:rPr>`
	`  <w:u w:val="single" w:color="4F81BD" w:themeColor="accent1" />`
	`  <w:asianLayout w:id="-1552701694" w:combine="lines" w:combineBrackets="curly" />`
	`  </w:rPr>`
	`  <w:t>lines in</w:t>`
	`</w:r>`
	`<w:r>`
	`  <w:rPr>`
	`  <w:asianLayout w:id="-1552701694" w:combine="lines" w:combineBrackets="curly" />`
	`  </w:rPr>`
	`  <w:t>one</w:t>`
	`</w:r>`
	
	Although there are three runs of content, all three regions shall be combined into a single two lines in one region based on the identical value used in the id attribute for all three runs. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

vert (Horizontal in Vertical (Rotate Text))	Specifies that characters in this run should be rendered with a 270 degree rotation to the left from all other contents of the line when displayed in the document as described above. If this attribute is omitted, then the contents of this run shall not be rotated with respect to the normal text flow. [*Example*: Consider a paragraph with the text `this word is vertical`, of which the

Attributes	Description

	word `vertical` shall be displayed vertically within the document. This constraint would be specified as follows in the WordprocessingML:
	
	`<w:r>`
	`  <w:rPr>`
	`  <w:eastAsianLayout w:id="2" w:vert="on" />`
	`  </w:rPr>`
	`  <w:t>vertical</w:t>`
	`</w:r>`
	
	The resulting text would be displayed with a 270 degree rotation from the other text content. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

vertCompress (Compress Rotated Text to Line Height)	Specifies whether the rotated text shall be compressed at display time in order to ensure that it fits into the existing line height without increasing the overall height of the line.
	
	If the vert attribute is not specified, then this attribute is ignored. If this attribute is omitted, then text shall not be compressed in order to fit into the existing height of the line when it is rotated.
	
	[*Example*: Consider a paragraph with the text `this word is vertical`, of which the word `vertical` shall be displayed vertically within the document but shall not change the height of the line. This constraint would be specified as follows in the WordprocessingML:
	
	`<w:r>`
	`  <w:rPr>`
	`  <w:eastAsianLayout w:id="2" w:vert="true"`
	`vertCompress="true" />`
	`  </w:rPr>`
	`  <w:t>vertical</w:t>`
	`</w:r>`
	
	The resulting text would be compressed in order to fit the height of the line as defined by all non-compressed characters. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EastAsianLayout">
   <attribute name="id" type="ST_DecimalNumber" use="optional"/>
   <attribute name="combine" type="ST_OnOff" use="optional"/>
   <attribute name="combineBrackets" type="ST_CombineBrackets" use="optional"/>
   <attribute name="vert" type="ST_OnOff" use="optional"/>
   <attribute name="vertCompress" type="ST_OnOff" use="optional"/>
</complexType>
```

2.3.2.9 effect (Animated Text Effect)

This element specifies an animated text effect which should be displayed when rendering the contents of this run. This effect is rendered around the extents of the text in the run in the same location as a run border with zero pixels of padding would be rendered (if such a run border was present).

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then no text effect shall be applied to the contents of this run.

[*Example*: Consider a run of text which shall have an animated text effect consisting of multiple colored flashing lights (see possible attribute values for descriptions of each effect). This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:effect w:val="lights"/>
</w:rPr>
```

This run explicitly declares that the effect property is `lights`, so the contents of this run will have an animated lights text effect. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description

val (Animated Text Effect Type)	Specifies the type of animated text effect which shall be applied to this text run.
	
	[*Example*: Consider a run of text which shall have an animated text effect consisting of multiple colored flashing lights. This constraint is specified using the following WordprocessingML:
	
	`<w:rPr>`
	`  <w:effect w:val="lights"/>`
	`</w:rPr>`
	
	This run explicitly declares a type of text effect, using the val property, of `lights`, so the contents of this run will have the animated lights text effect. *end example*]
	
	The possible values for this attribute are defined by the ST_TextEffect simple type (§2.18.101).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextEffect">
   <attribute name="val" type="ST_TextEffect" use="required"/>
</complexType>
```

2.3.2.10 em (Emphasis Mark)

This element specifies the emphasis mark which shall be displayed for each non-space character in this run. An *emphasis mark* is an additional character that is rendered above or below the main character glyph as specified by the contents of the val attribute.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then no emphasis mark shall be added to each character in the contents of this run.

[*Example*: Consider a run of text which shall have a dot underneath each character as an emphasis mark. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:em w:val="dot"/>
</w:rPr>
```

This run explicitly declares that the emphasis mark type is dot, so the contents of this run will have a dot emphasis mark above each character. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description

val (Emphasis Mark Type)	Specifies the emphasis mark type used for each character in this run.
	
	[*Example*: Consider a run of text which shall have a dot underneath each character as an emphasis mark. This constraint is specified using the following WordprocessingML:
	
	`<w:rPr>`
	`  <w:em w:val="dot"/>`
	`</w:rPr>`
	
	This run explicitly declares that the em type is dot, so the contents of this run will have a dot emphasis mark beneath each character. *end example*]
	
	The possible values for this attribute are defined by the ST_Em simple type (§2.18.28).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Em">
   <attribute name="val" type="ST_Em" use="required"/>
</complexType>
```

2.3.2.11 emboss (Embossing)

This element specifies that the contents of this run should be displayed as if embossed, which makes text appear as if it is raised off the page in relief.

This formatting property is a *toggle property*, which specifies that its behavior differs between its use within a style definition and its use as direct formatting. When used as part of a style definition, setting this property shall toggle the current state of that property as specified up to this point in the hierarchy (i.e. applied to not applied, and vice versa). Setting it to `false` (or an equivalent) shall result in the current setting remaining unchanged. However, when used as direct formatting, setting this property to `true` or `false` shall set the absolute state of the resulting property.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then embossing shall not be applied to the contents of this run.

This element shall not be present with either the imprint (§2.3.2.16) or outline (§2.3.2.21) properties on the same run, since they are mutually exclusive in terms of appearance.

[*Example*: Consider a run of text which shall have the emboss property explicitly turned on for the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:emboss w:val="true"/>
</w:rPr>
```

This run explicitly declares that the emboss property is `true`, so the contents of this run will appear embossed. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted.

Attributes	Description
	A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.2.12 fitText (Manual Run Width)

This element specifies that the contents of this run shall not be automatically displayed based on the width of its contents, rather its contents shall be resized to fit the width specified by the val attribute. This expansion/contraction shall be performed by equally increasing/decreasing the size of each character in this run's contents when displayed.

If this element is omitted, then the contents of this run shall be displayed based on the size of its contents.

[*Example*: Consider a document with a run which shall be displayed in exactly one-half inch of space, regardless of its contents. This constraint would be specified using the following WordprocessingML:

```
<w:r>
  <w:rPr>
    <w:fitText w:id="50" w:val="720" />
  </w:rPr>
  <w:t>This text shall be displayed in one-half of an inch.</w:t>
</w:r>
```

The resulting run contents shall be displayed in exactly `720` twentieths of a point (one half of an inch) when displayed in a document. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
id (Fit Text Run ID)	Specifies a unique ID which shall be used to link multiple contiguous runs containing fitText elements to each other to ensure that their contents are correctly merged into

Attributes	Description

	the specified width in the document.
	
	This means that multiple runs which are broken apart due to differences in formatting can be identified as belonging to the same grouping in terms of fitText properties, although they are multiple runs of text in the WordprocessingML.
	
	If the runs are not contiguous, then the id attribute is ignored, and the runs are not linked.
	
	If this attribute is omitted, then this run has no id and shall not be linked with any other run in the parent paragraph.
	
	[*Example*: Consider the following three runs in a document, which should be fit into exactly one inch at display time:
	
	`<w:r>`
	`  <w:rPr>`
	`  <w:fitText w:id="99" w:val="1440" />`
	`  </w:rPr>`
	`  <w:t>fit this into</w:t>`
	`</w:r>`
	`<w:r>`
	`  <w:rPr>`
	`  <w:b/>`
	`  <w:fitText w:id="99" w:val="1440" />`
	`  </w:rPr>`
	`  <w:t>one</w:t>`
	`</w:r>`
	`<w:r>`
	`  <w:rPr>`
	`  <w:fitText w:id="99" w:val="1440" />`
	`  </w:rPr>`
	`  <w:t>inch</w:t>`
	`</w:r>`
	
	Although there are three runs of content, all three regions shall be combined into a single fit text region (e.g. they all fit into one inch, rather than one inch each) based on the identical value used in the id attribute for all three runs. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

val (Value)	This attribute specifies the exact width of space which this run shall be fit into when displayed in the document. [*Example*: Consider a document with a run which shall be displayed in exactly one-half inch of space, regardless of its contents. This constraint would be specified using the

Attributes	Description

	following WordprocessingML:
	
	`<w:r>`
	`  <w:rPr>`
	`  <w:fitText w:id="50" w:val="720" />`
	`  </w:rPr>`
	`  <w:t>This text shall be displayed in one-half of an inch.</w:t>`
	`  </w:r>`
	
	The resulting run contents shall be displayed in exactly 720 twentieths of a point (one half of an inch) when displayed in a document. *end example*]
	
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FitText">
   <attribute name="val" type="ST_TwipsMeasure" use="required"/>
   <attribute name="id" type="ST_DecimalNumber" use="optional"/>
</complexType>
```

2.3.2.13 highlight (Text Highlighting)

This element specifies a highlighting color which is applied as a background behind the contents of this run.

If this run has any background shading specified using the shd element (§2.3.2.30), then the background shading shall be superseded by the highlighting color when the contents of this run are displayed.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then text highlighting shall not be applied to the contents of this run.

[*Example*: Consider a run within a paragraph which has run shading applied as well as yellow text highlighting using the highlight element. This formatting is specified using the following WordprocessingML:

```
<w:rPr>
  <w:highlight w:val="yellow" />
  <w:shd w:themeFill="accent2" w:themeFillTint="66" />
</w:rPr>
```

The resulting run would have yellow highlighting visible over its contents, as the highlighting supersedes the shading for the contents of the run. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30);

Parent Elements
rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description

val (Highlighting Color)	Specifies the color of the text highlighting which shall be applied to the contents of this run.
	
	[*Example*: Consider a text run which shall be displayed with colored text highlighting. This highlighting would be specified using the following WordprocessingML:
	
	`<w:rPr>`
	`  <w:highlight w:val="red" />`
	`</w:rPr>`
	
	The resulting text highlighting would be `red`, as this is the color specified by the val attribute. *end example*]
	
	The possible values for this attribute are defined by the ST_HighlightColor simple type (§2.18.46).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Highlight">
   <attribute name="val" type="ST_HighlightColor" use="required"/>
</complexType>
```

2.3.2.14 i (Italics)

This element specifies whether the italic property should be applied to all non-complex script characters in the contents of this run when displayed in a document.

This formatting property is a *toggle property*, which specifies that its behavior differs between its use within a style definition and its use as direct formatting. When used as part of a style definition, setting this property shall toggle the current state of that property as specified up to this point in the hierarchy (i.e. applied to not applied, and vice versa). Setting it to `false` (or an equivalent) shall result in the current setting remaining unchanged. However, when used as direct formatting, setting this property to `true` or `false` shall set the absolute state of the resulting property.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy* .If this element is never applied in the style hierarchy, then italics shall not be applied to non-complex script characters.

[*Example*: Consider a run of text which shall have the i property explicitly turned on for the non-complex script contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:i />
</w:rPr>
```

This run explicitly declares that the i property is `true` for the non-complex script contents of this run. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.2.15 iCs (Complex Script Italics)

This element specifies whether the italic property should be applied to all complex script characters in the contents of this run when displayed in a document.

This formatting property is a *toggle property*, which specifies that its behavior differs between its use within a style definition and its use as direct formatting. When used as part of a style definition, setting this property shall toggle the current state of that property as specified up to this point in the hierarchy (i.e. applied to not applied, and vice versa). Setting it to `false` (or an equivalent) shall result in the current setting remaining unchanged. However, when used as direct formatting, setting this property to `true` or `false` shall set the absolute state of the resulting property.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then italics shall not be applied to complex script characters.

[*Example*: Consider a run of text which shall have the iCs property explicitly turned on for the complex script contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:iCs w:val="true"/>
</w:rPr>
```

This run explicitly declares that the iCs property is `true`, so italics are turned on for the complex script contents of this run. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.2.16 imprint (Imprinting)

This element specifies that the contents of this run should be displayed as if imprinted, which makes text appear to be imprinted or pressed into page (also referred to as 'engrave').

This formatting property is a *toggle property*, which specifies that its behavior differs between its use within a style definition and its use as direct formatting. When used as part of a style definition, setting this property shall toggle the current state of that property as specified up to this point in the hierarchy (i.e. applied to not applied, and vice versa). Setting it to `false` (or an equivalent) shall result in the current setting remaining unchanged. However, when used as direct formatting, setting this property to `true` or `false` shall set the absolute state of the resulting property.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then imprinting shall not be applied to the contents of this run.

This element shall not be present with either the emboss (§2.3.2.11) or outline (§2.3.2.21) properties on the same run, since they are mutually exclusive in terms of appearance.

[*Example*: Consider a run of text which shall have the imprint property explicitly turned on for the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:imprint w:val="true"/>
</w:rPr>
```

This run explicitly declares that the imprint property is `true`, so the contents of this run will appear imprinted. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.2.17 kern (Font Kerning)

This element specifies whether font kerning shall be applied to the contents of this run. If it is specified, then kerning shall be automatically adjusted when displaying characters in this run as needed.

The val attribute specifies the smallest font size which shall have its kerning automatically adjusted if this setting is specified. If the font size in the sz element (§2.3.2.36) is smaller than this value, then no font kerning shall be performed.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then font kerning shall not be applied to the contents of this run.

[*Example*: Consider the following WordprocessingML run which has font kerning properties specified:

```
<w:r>
  <w:rPr>
    <w:sz w:val="22" />
    <w:kern w:val="28" />
  </w:rPr>
</w:r>
```

Even though font kerning is turned on via the kern element, the contents of this run shall not be kerned because that settings only applied to font sizes of 14 points (28 half-points) or larger. If the kern element's val attribute was less than or equal to the sz element's val attribute, then kerning would be applied:

```
<w:r>
  <w:rPr>
    <w:sz w:val="22" />
    <w:kern w:val="22" />
  </w:rPr>
</w:r>
```

end example]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description

Attributes	Description

val (Half Point Measurement)	Specifies a positive measurement specified in half-points (1/144 of an inch).
	
	The contents of this attribute value are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:rPr>`
	`  <w:sz w:val="28" />`
	`</w:rPr>`
	
	The value of the val attribute is the font size of the run's contents.
	
	However, consider the following fragment:
	
	`<w:rPr>`
	`  <w:kern w:val="30" />`
	`</w:rPr>`
	
	In this case, the value in the val attribute is the minimum size for which font characters shall be automatically kerned.
	
	In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_HpsMeasure simple type (§2.18.48).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HpsMeasure">
   <attribute name="val" type="ST_HpsMeasure" use="required"/>
</complexType>
```

2.3.2.18 lang (Languages for Run Content)

This element specifies the languages which shall be used to check spelling and grammar (if requested) when processing the contents of this run.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the languages for the contents of this run shall be automatically determined based on their contents using any method desired.

[*Example*: Consider a run which contains both Latin and complex script characters in its contents. If those contents should be interpreted as French (Canada) and Hebrew, respectively, that requirement would be specified as follows in the resulting WordprocessingML:

```
<w:r>
  <w:rPr>
```

```
      <w:lang w:val="fr-CA" w:bidi="he-IL" />
    </w:rPr>
  </w:r>
```

The resulting run specifies that any complex script contents shall be spell and grammar checked as if they were Hebrew, and any Latin character contents shall be spell and grammar checked as if they were French (Canada). *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description

bidi (Complex Script Language)	Specifies the language which shall be used when processing the contents of this run which use complex script characters, as determined by the Unicode character values of the run content.
	
	If this attribute is omitted, then the languages for the contents of this run using complex script characters shall be automatically determined based on their contents using any appropriate method.
	
	[*Example*: Consider a run which contains complex script characters in its contents. If those contents should be interpreted as Hebrew, that requirement would be specified as follows in the resulting WordprocessingML:
	
	`<w:r>`
	`  <w:rPr>`
	`  <w:lang w:bidi="he-IL" />`
	`  </w:rPr>`
	`</w:r>`
	
	The resulting run specifies that any complex script contents shall be spell and grammar checked using a Hebrew dictionary and grammar engine, if one is available. *end example*]
	
	The possible values for this attribute are defined by the ST_Lang simple type (§2.18.51).

eastAsia (East Asian Language)	Specifies the language which shall be used when processing the contents of this run which use East Asian characters, as determined by the Unicode character values of the run content. If this attribute is omitted, then the languages for the contents of this run using East Asian characters shall be automatically determined based on their contents using any appropriate method. [*Example*: Consider a run which contains East Asian characters in its contents. If those contents should be interpreted as Korean, that requirement would be specified as

Attributes	Description

	follows in the resulting WordprocessingML:
	
	`<w:r>`
	`  <w:rPr>`
	`  <w:lang w:bidi="ko-KR" />`
	`  </w:rPr>`
	`</w:r>`
	
	The resulting run specifies that any complex script contents shall be spell and grammar checked using a Korean dictionary and grammar engine, if one is available. *end example*]
	
	The possible values for this attribute are defined by the ST_Lang simple type (§2.18.51).

val (Latin Language)	Specifies the language which shall be used to check spelling and grammar (if requested) when processing the contents of this run which use Latin characters, as determined by the Unicode character values of the run content.
	
	If this attribute is omitted, then the languages for the contents of this run using Latin characters shall be automatically determined based on their contents using any appropriate method.
	
	[*Example*: Consider a run which contains Latin characters in its contents. If those contents should be interpreted as English (Canada), that requirement would be specified as follows in the resulting WordprocessingML:
	
	`<w:r>`
	`  <w:rPr>`
	`  <w:lang w:bidi="en-CA" />`
	`  </w:rPr>`
	`</w:r>`
	
	The resulting run specifies that any complex script contents shall be spell and grammar checked using a English (Canada) dictionary and grammar engine, if one is available. *end example*]
	
	The possible values for this attribute are defined by the ST_Lang simple type (§2.18.51).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Language">
   <attribute name="val" type="ST_Lang" use="optional"/>
   <attribute name="eastAsia" type="ST_Lang" use="optional"/>
   <attribute name="bidi" type="ST_Lang" use="optional"/>
</complexType>
```

2.3.2.19 noProof (Do Not Check Spelling or Grammar)

This element specifies that the contents of this run shall not report any errors when the document is scanned for spelling and grammar. [*Note*: It is entirely at the consumer's/producer's discretion whether this is done by not checking the region for spelling and grammar, or simply by suppressing the results. *end note*]

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then spelling and grammar error shall not be suppressed on the contents of this run.

[*Example*: Consider a run of text which shall not ever have spelling or grammar errors reported for the contents of the run, for example, the XML fragments included in this Specification. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:noProof w:val="true"/>
</w:rPr>
```

This run explicitly declares that the noProof property is `true`, so the contents of this run will never report spelling or grammar errors. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.2.20 oMath (Office Open XML Math)

This element specifies that this run contains WordprocessingML which shall be handled as though it was Office Open XML Math.

[*Rationale*: Like other run properties may be applied to the glyph representing the paragraph mark, it is possible to create an Office Open XML Math equation on an empty paragraph as well. Since that paragraph mark must be defined by WordprocessingML, it is not possible to store the paragraph using the Office Open XML Math markup. Instead, this run property is stored on the paragraph mark's run properties to indicate that the paragraph mark is part of an Office Open XML Math equation. For example, the first paragraph below is stored as Office Open XML Math:

The paragraph must be a p (§2.3.1.22) element, but that would mean the data loss of the Math markup when saving as a WordprocessingML package. In order to prevent that data loss, this property stores the Math property as a run property. *end rationale*]

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then this run shall not be treated as Office Open XML Math.

This property may be applied to any run, but that should only introduce the semantic that the run is math in the user interface, and shall not change the appearance of the text.

[*Example*: Consider a paragraph in WordprocessingML where the paragraph mark glyph (the pilcrow mark - ¶) has been formatted as Math. Since this mark is not an actual run, it cannot be written out in the Office Open XML Math syntax, and must be written out as a property on the actual run as follows:

```
<w:pPr>
  <w:rPr>
    <w:oMath />
  </w:rPr>
</w:pPr>
```

This property is therefore used to roundtrip the math setting on this paragraph mark character. *end example*]

Parent Elements

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.2.21 outline (Display Character Outline)

This element specifies that the contents of this run should be displayed as if they have an outline, by drawing a one pixel wide border around the inside and outside borders of each character glyph in the run.

This formatting property is a *toggle property*, which specifies that its behavior differs between its use within a style definition and its use as direct formatting. When used as part of a style definition, setting this property shall toggle the current state of that property as specified up to this point in the hierarchy (i.e. applied to not applied, and vice versa). Setting it to false (or an equivalent) shall result in the current setting remaining unchanged. However, when used as direct formatting, setting this property to true or false shall set the absolute state of the resulting property.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then outline shall not be applied to the contents of this run.

This element shall not be present with either the emboss (§2.3.2.11) or imprint (§2.3.2.16) properties on the same run, since they are mutually exclusive in terms of appearance.

[*Example*: Consider a run of text which shall have the outline property explicitly turned off for the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:outline w:val="false"/>
</w:rPr>
```

This run explicitly declares that the outline property is `false`, so the contents of this run will not appear as if they have an exterior outline around them. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.2.22 position (Vertically Raised or Lowered Text)

This element specifies the amount by which text shall be raised or lowered for this run in relation to the default baseline of the surrounding non-positioned text. This allows the text to be repositioned without altering the font size of the contents.

If the val attribute is positive, then the parent run shall be raised above the baseline of the surrounding text by the specified number of half-points. If the val attribute is negative, then the parent run shall be lowered below the baseline of the surrounding text by the specified number of half-points.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the text shall not be raised or lowered relative to the default baseline location for the contents of this run.

[*Example*: Consider a run which shall be positioned 12 points above the default baseline location when displaying its contents. This requirement would be specified using the following WordprocessingML:

```
<w:rPr>
  <w:position w:val="24" />
</w:rPr>
```

The resulting run is positioned 24 half-points above the default baseline location because the contents of the val attribute are positive. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description

val (Signed Half-Point Measurement)	Specifies a positive or negative measurement in half-points (1/144 of an inch).
	
	The contents of this attribute value are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:rPr>`
	`  <w:position w:val="-12" />`
	`</w:rPr>`
	
	In this case, the value in the val attribute is amount by which the specified run shall be raised or lowered compared to the baseline of the surrounding text.
	
	In all cases, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_SignedHpsMeasure simple type (§2.18.87).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SignedHpsMeasure">
   <attribute name="val" type="ST_SignedHpsMeasure" use="required"/>
</complexType>
```

2.3.2.23 r (Text Run)

This element specifies a run of content in the parent field, hyperlink, custom XML element, structured document tag, smart tag, or paragraph.

The contents of a run in a WordprocessingML document shall consist of any combination of run content.

[*Example*: Consider a basic WordprocessingML paragraph with a pair of runs. This run would be expressed as follows:

```
<w:document>
  <w:body>
    <w:p>
      <w:r>
        <w:t>Text</w:t>
      </w:r>
      <w:fldSimple w:instr="AUTHOR">
        <w:r>
          <w:t>Author Name</w:t>
        </w:r>
      </w:fldSimple>
    </w:p>
  </w:body>
</w:document>
```

The r element is the container for all of the content in the run, which in this example includes both a run in the paragraph and a run within a simple field. *end example*]

Parent Elements
customXml (§2.5.1.5); del (§2.13.5.12); fldSimple (§2.16.21); hyperlink (§2.16.24); ins (§2.13.5.20); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.35); smartTag (§2.5.1.9)

Child Elements	Subclause
annotationRef (Comment Information Block)	§2.13.4.1
br (Break)	§2.3.3.1
commentReference (Comment Content Reference Mark)	§2.13.4.5
continuationSeparator (Continuation Separator Mark)	§2.11.1
cr (Carriage Return)	§2.3.3.4
dayLong (Date Block - Long Day Format)	§2.3.3.5
dayShort (Date Block - Short Day Format)	§2.3.3.6
delInstrText (Deleted Field Code)	§2.16.13

Child Elements	Subclause
delText (Deleted Text)	§2.3.3.7
drawing (DrawingML Object)	§2.3.3.9
endnoteRef (Endnote Reference Mark)	§2.11.6
endnoteReference (Endnote Reference)	§2.11.7
fldChar (Complex Field Character)	§2.16.18
footnoteRef (Footnote Reference Mark)	§2.11.13
footnoteReference (Footnote Reference)	§2.11.14
instrText (Field Code)	§2.16.25
lastRenderedPageBreak (Position of Last Calculated Page Break)	§2.3.3.13
monthLong (Date Block - Long Month Format)	§2.3.3.15
monthShort (Date Block - Short Month Format)	§2.3.3.16
noBreakHyphen (Non Breaking Hyphen Character)	§2.3.3.18
object (Inline Embedded Object)	§2.3.3.19
pgNum (Page Number Block)	§2.3.3.20
pict (VML Object)	§2.3.3.21
ptab (Absolute Position Tab Character)	§2.3.3.22
rPr (Run Properties)	§2.3.2.25
ruby (Phonetic Guide)	§2.3.3.24
separator (Footnote/Endnote Separator Mark)	§2.11.23
softHyphen (Optional Hyphen Character)	§2.3.3.28
sym (Symbol Character)	§2.3.3.29
t (Text)	§2.3.3.30
tab (Tab Character)	§2.3.3.31
yearLong (Date Block - Long Year Format)	§2.3.3.32
yearShort (Date Block - Short Year Format)	§2.3.3.33

Attributes	Description
rsidDel (Revision Identifier for Run Deletion)	Specifies a unique identifier used to track the editing session when the run was deleted from the main document. All rsid* attributes throughout this document with the same value, if present, must indicate that those regions were modified during the same editing session (time between subsequent save actions). A producer may choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications

Attributes	Description
	in this document. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).
rsidR (Revision Identifier for Run)	Specifies a unique identifier used to track the editing session when the run was added to the main document. All rsid* attributes throughout this document with the same value, if present, must indicate that those regions were modified during the same editing session (time between subsequent save actions). A producer may choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications in this document. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).
rsidRPr (Revision Identifier for Run Properties)	Specifies a unique identifier used to track the editing session when the run properties were last modified in the main document. All rsid* attributes throughout this document with the same value, if present, must indicate that those regions were modified during the same editing session (time between subsequent save actions). A producer may choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications in this document. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_R">
   <sequence>
      <group ref="EG_RPr" minOccurs="0"/>
      <group ref="EG_RunInnerContent" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="rsidRPr" type="ST_LongHexNumber"/>
   <attribute name="rsidDel" type="ST_LongHexNumber"/>
   <attribute name="rsidR" type="ST_LongHexNumber"/>
</complexType>
```

2.3.2.24 rFonts (Run Fonts)

This element specifies the fonts which shall be used to display the text contents of this run. Within a single run, there may be up to four types of content present which shall each be allowed to use a unique font:

- ASCII
- High ANSI
- Complex Script
- East Asian

The use of each of these fonts shall be determined by the Unicode character values of the run content, unless manually overridden via use of the cs element (§2.3.2.6).

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the text shall be displayed in any default font which supports each type of content.

[*Example*: Consider a single text run with both Arabic and English text, as follows:

English العربية

This content may be expressed in a single WordprocessingML run:

```
<w:r>
  <w:t>English العربية</w:t>
</w:r>
```

Although it is in the same run, the contents are in different font faces by specifying a different font for ASCII and CS characters in the run:

```
<w:r>
  <w:rPr>
    <w:rFonts w:ascii="Courier New" w:cs="Times New Roman" />
  </w:rPr>
  <w:t>English العربية</w:t>
</w:r>
```

This text run shall therefore use the Courier New font for all characters in the ASCII range, and shall use the Times New Roman font for all characters in the Complex Script range. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
ascii (ASCII Font)	Specifies a font which shall be used to format all characters in the ASCII range (0 - 127) within the parent run. If the asciiTheme attribute is also specified, then this attribute shall be ignored and that

Attributes	Description

	value shall be used instead.
	
	If this attribute is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this attribute is never applied in the style hierarchy, then the text shall be displayed in any default font which supports ASCII content.
	
	[*Example*: Consider a run of ASCII text which shall be displayed using the `Courier New` font. This requirement would be specified as follows in the resulting WordprocessingML:
	
	`<w:rPr>`
	`  <w:rFonts w:ascii=”Courier New” />`
	`</w:rPr>`
	
	The ascii attribute specifies that the run shall use the `Courier New` font for all text in the ASCII range. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

asciiTheme (ASCII Theme Font)	Specifies a theme font which shall be used to format all characters in the ASCII range (0 - 127) within the parent run. This theme font is a reference to one of the predefined theme fonts, located in the document's Theme part,which allows for font information to be set centrally in the document.
	
	If the ascii attribute is also specified, then that attribute shall be ignored and this value shall be used instead.
	
	If this attribute is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this attribute is never applied in the style hierarchy, then the text shall be displayed in the font specified by the ascii attribute.
	
	[*Example*: Consider a run of ASCII text which shall be displayed using the `majorASCII` theme font. This requirement would be specified as follows in the resulting WordprocessingML:
	
	`<w:rPr>`
	`  <w:rFonts w:asciiTheme=”majorAscii” />`
	`</w:rPr>`
	
	The ascii attribute specifies that the run shall use the `majorAscii` theme font as defined in the document's themes part for all text in the ASCII range. *end example*]
	
	The possible values for this attribute are defined by the ST_Theme simple type (§2.18.103).

cs (Complex Script Font)	Specifies a font which shall be used to format all characters in a complex script Unicode range within the parent run.

Attributes	Description

	If the csTheme attribute is also specified, then this attribute shall be ignored and that value shall be used instead.
	
	If this attribute is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this attribute is never applied in the style hierarchy, then the text shall be displayed in any default font which supports complex script content.
	
	[*Example*: Consider a run of Arabic text which shall be displayed using the `Arial Unicode MS` font. This requirement would be specified as follows in the resulting WordprocessingML:
	
	`<w:rPr>`
	`  <w:rFonts w:cs="Arial Unicode MS" />`
	`</w:rPr>`
	
	The cs attribute specifies that the run shall use the `Arial Unicode MS` font for all text in a complex script range. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

cstheme (Complex Script Theme Font)	Specifies a theme font which shall be used to format all characters in a complex script Unicode range within the parent run. This theme font is a reference to one of the predefined theme fonts, located in the document's Theme part,which allows for font information to be set centrally in the document.
	
	If the cs attribute is also specified, then that attribute shall be ignored and this value shall be used instead.
	
	If this attribute is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this attribute is never applied in the style hierarchy, then the text shall be displayed in the font specified by the cs attribute.
	
	[*Example*: Consider a run of Arabic text which shall be displayed using the `majorBidi` theme font. This requirement would be specified as follows in the resulting WordprocessingML:
	
	`<w:rPr>`
	`  <w:rFonts w:csTheme="majorBidi" />`
	`</w:rPr>`
	
	The csTheme attribute specifies that the run shall use the `majorBidi` theme font as defined in the document's themes part for all text in a complex script range. *end example*]
	
	The possible values for this attribute are defined by the ST_Theme simple type (§2.18.103).

Attributes	Description

eastAsia (East Asian Font)	Specifies a font which shall be used to format all characters in an East Asian Unicode range within the parent run.
	
	If the eastAsiaTheme attribute is also specified, then this attribute shall be ignored and that value shall be used instead.
	
	If this attribute is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this attribute is never applied in the style hierarchy, then the text shall be displayed in any default font which supports East Asian content.
	
	[*Example*: Consider a run of Japanese text which shall be displayed using the `MS Mincho` font. This requirement would be specified as follows in the resulting WordprocessingML:
	
	`<w:rPr>`
	`  <w:rFonts w:eastAsia=”MS Mincho” />`
	`</w:rPr>`
	
	The eastAsia attribute specifies that the run shall use the `MS Mincho` font for all text in an East Asian range. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

eastAsiaTheme (East Asian Theme Font)	Specifies a theme font which shall be used to format all characters in an East Asian Unicode range within the parent run. This theme font is a reference to one of the predefined theme fonts, located in the document's Theme part,which allows for font information to be set centrally in the document.
	
	If the eastAsia attribute is also specified, then that attribute shall be ignored and this value shall be used instead.
	
	If this attribute is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this attribute is never applied in the style hierarchy, then the text shall be displayed in the font specified by the eastAsia attribute.
	
	[*Example*: Consider a run of Japanese text which shall be displayed using the `minorEastAsia` theme font. This requirement would be specified as follows in the resulting WordprocessingML:
	
	`<w:rPr>`
	`  <w:rFonts w:eastAsiaTheme=”minorEastAsia” />`
	`</w:rPr>`
	
	The eastAsiaTheme attribute specifies that the run shall use the `minorEastAsia` theme font as defined in the document's themes part for all text in an East Asian range. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_Theme simple type (§2.18.103).

hAnsi (High ANSI Font)	Specifies a font which shall be used to format all characters in a Unicode range within the parent run which does not fall into one of the three categories defined above, which is called the *high ANSI* range in WordprocessingML.
	
	If the hAnsiTheme attribute is also specified, then this attribute shall be ignored and that value shall be used instead.
	
	If this attribute is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this attribute is never applied in the style hierarchy, then the text shall be displayed in any default font which supports high ANSI content.
	
	[*Example*: Consider a run of text which falls into a high ANSI range, and shall be displayed using the `Bauhaus 93` font. This requirement would be specified as follows in the resulting WordprocessingML:
	
	`<w:rPr>`
	`  <w:rFonts w:hAnsi=”Bauhaus 93” />`
	`</w:rPr>`
	
	The hAnsi attribute specifies that the run shall use the `Bauhaus 93` font for all text in a high ANSI range. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

hAnsiTheme (High ANSI Theme Font)	Specifies a theme font which shall be used to format all characters in a Unicode range within the parent run which does not fall into one of the three categories defined above, which is called the *high ANSI* range in WordprocessingML. This theme font is a reference to one of the predefined theme fonts, located in the document's Theme part,which allows for font information to be set centrally in the document.
	
	If the hAnsi attribute is also specified, then that attribute shall be ignored and this value shall be used instead.
	
	If this attribute is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this attribute is never applied in the style hierarchy, then the text shall be displayed in the font specified by the hAnsi attribute.
	
	[*Example*: Consider a run of text which falls into a high ANSI range, and shall be displayed using the `minorHAnsi` theme font. This requirement would be specified as follows in the resulting WordprocessingML:
	
	`<w:rPr>`
	`  <w:rFonts w:hAnsiTheme="minorHAnsi" />`
	`</w:rPr>`

Attributes	Description
	The hAnsiTheme attribute specifies that the run shall use the `minorHAnsi` theme font as defined in the document's themes part for all text in a high ANSI range. *end example*] The possible values for this attribute are defined by the ST_Theme simple type (§2.18.103).

hint (Font Content Type)	Specifies the font type which shall be used to format any ambiguous characters in the current run.
	
	There are certain characters which are not explicitly stored in the document, and may be mapped into multiple categories of the four mentioned above. This attribute shall be used to arbitrate that conflict, and determine how ambiguities in this run shall be handled. [*Note*: This is primarily used to handle the formatting on the paragraph mark glyph, and other characters that are not stored as text in the WordprocessingML document. *end note*]
	
	If this attribute is omitted, then this ambiguity may be resolved by any means available.
	
	[*Example*: Consider the run representing the paragraph mark glyph, which is not stored as a physical character. Since this could therefore be formatted with any of the fonts specified for the run, this ambiguity is resolved using the following WordprocessingML:
	
	`<w:pPr>`
	`  <w:rPr>`
	`  <w:rFonts w:hint="eastAsia" />`
	`  </w:rPr>`
	`</w:pPr>`
	
	The hint attribute specifies that the run shall use the `eastAsia` font (theme or not, whichever is in use for East Asian text) as defined for this range. *end example*]
	
	The possible values for this attribute are defined by the ST_Hint simple type (§2.18.47).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Fonts">
   <attribute name="hint" type="ST_Hint"/>
   <attribute name="ascii" type="ST_String"/>
   <attribute name="hAnsi" type="ST_String"/>
   <attribute name="eastAsia" type="ST_String"/>
   <attribute name="cs" type="ST_String"/>
   <attribute name="asciiTheme" type="ST_Theme"/>
   <attribute name="hAnsiTheme" type="ST_Theme"/>
   <attribute name="eastAsiaTheme" type="ST_Theme"/>
   <attribute name="cstheme" type="ST_Theme"/>
</complexType>
```

2.3.2.25 rPr (Run Properties)

This element specifies a set of run properties which shall be applied to the contents of the parent run after all style formatting has been applied to the text. These properties are defined as *direct formatting*, since they are directly applied to the run and supersede any formatting from styles.

This formatting is applied at the following location in the *style hierarchy*:

- Document defaults
- Table styles
- Numbering styles
- Paragraph styles
- Character styles
- Direct formatting (this element)

[*Example*: Consider a run which should have a set of run formatting properties. This set of properties is specified in the run properties as follows:

```
<w:r>
  <w:rPr>
    <w:b />
    <w:imprint />
    <w:lang w:val="en-ca" />
  </w:rPr>
</w:r>
```

The rPr element specifies the properties which are applied to the current run - in this case, bold formatting on the run contents using the b element (§2.3.2.1), an imprinted (engraved) text effect using the imprint element (§2.3.2.16), and that this text should be interpreted as English (Canada) when spell or grammar checking the run text using the lang element (§2.3.2.18). *end example*]

Parent Elements
ctrlPr (§7.1.2.23); r (§7.1.2.87); r (§2.3.2.23)

Child Elements	Subclause
b (Bold)	§2.3.2.1
bCs (Complex Script Bold)	§2.3.2.2
bdr (Text Border)	§2.3.2.3
caps (Display All Characters As Capital Letters)	§2.3.2.4
color (Run Content Color)	§2.3.2.5
cs (Use Complex Script Formatting on Run)	§2.3.2.6
dstrike (Double Strikethrough)	§2.3.2.7

Child Elements	Subclause
eastAsianLayout (East Asian Typography Settings)	§2.3.2.8
effect (Animated Text Effect)	§2.3.2.9
em (Emphasis Mark)	§2.3.2.10
emboss (Embossing)	§2.3.2.11
fitText (Manual Run Width)	§2.3.2.12
highlight (Text Highlighting)	§2.3.2.13
i (Italics)	§2.3.2.14
iCs (Complex Script Italics)	§2.3.2.15
imprint (Imprinting)	§2.3.2.16
kern (Font Kerning)	§2.3.2.17
lang (Languages for Run Content)	§2.3.2.18
noProof (Do Not Check Spelling or Grammar)	§2.3.2.19
oMath (Office Open XML Math)	§2.3.2.20
outline (Display Character Outline)	§2.3.2.21
position (Vertically Raised or Lowered Text)	§2.3.2.22
rFonts (Run Fonts)	§2.3.2.24
rPrChange (Revision Information for Run Properties)	§2.13.5.32
rStyle (Referenced Character Style)	§2.3.2.27
rtl (Right To Left Text)	§2.3.2.28
shadow (Shadow)	§2.3.2.29
shd (Run Shading)	§2.3.2.30
smallCaps (Small Caps)	§2.3.2.31
snapToGrid (Use Document Grid Settings For Inter-Character Spacing)	§2.3.2.32
spacing (Character Spacing Adjustment)	§2.3.2.33
specVanish (Paragraph Mark Is Always Hidden)	§2.3.2.34
strike (Single Strikethrough)	§2.3.2.35
sz (Font Size)	§2.3.2.36
szCs (Complex Script Font Size)	§2.3.2.37
u (Underline)	§2.3.2.38
vanish (Hidden Text)	§2.3.2.39
vertAlign (Subscript/Superscript Text)	§2.3.2.40
w (Expanded/Compressed Text)	§2.3.2.41
webHidden (Web Hidden Text)	§2.3.2.42

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RPr">
   <sequence>
      <group ref="EG_RPrContent" minOccurs="0"/>
   </sequence>
</complexType>
```

2.3.2.26 rPr (Previous Run Properties)

This element specifies a set of run properties which shall be attributed to a revision by a particular author and at a particular time. This element contains the set of properties which have been tracked as a specific set of revisions by one author.

[*Example*: Consider a run which has a set of run formatting properties that were added with revision tracking turned on. This set of revised properties is specified in the run properties as follows:

```
<w:r>
  <w:rPr>
    <w:b />
    <w:imprint />
    <w:lang w:val="en-ca" />
    <w:rPrChange w:author="user1">
      <w:rPr>
        <w:i />
        <w:dstrike w:val="false" />
      </w:rPr>
    </w:rPrChange>
  </w:rPr>
</w:r>
```

The rPr element under rPrChange specifies the properties which were applied to the current run before revision tracking was turned on - in this case, italics using the i element (§2.3.2.14), and that any double strikethrough which was applied based on the style hierarchy shall be turned off using the dstrike element (§2.3.2.7). *end example*]

Parent Elements
del (§2.13.5.15); ins (§2.13.5.17); rPrChange (§2.13.5.32)

Child Elements	Subclause
b (Bold)	§2.3.2.1
bCs (Complex Script Bold)	§2.3.2.2
bdr (Text Border)	§2.3.2.3
caps (Display All Characters As Capital Letters)	§2.3.2.4

Child Elements	Subclause
color (Run Content Color)	§2.3.2.5
cs (Use Complex Script Formatting on Run)	§2.3.2.6
dstrike (Double Strikethrough)	§2.3.2.7
eastAsianLayout (East Asian Typography Settings)	§2.3.2.8
effect (Animated Text Effect)	§2.3.2.9
em (Emphasis Mark)	§2.3.2.10
emboss (Embossing)	§2.3.2.11
fitText (Manual Run Width)	§2.3.2.12
highlight (Text Highlighting)	§2.3.2.13
i (Italics)	§2.3.2.14
iCs (Complex Script Italics)	§2.3.2.15
imprint (Imprinting)	§2.3.2.16
kern (Font Kerning)	§2.3.2.17
lang (Languages for Run Content)	§2.3.2.18
noProof (Do Not Check Spelling or Grammar)	§2.3.2.19
oMath (Office Open XML Math)	§2.3.2.20
outline (Display Character Outline)	§2.3.2.21
position (Vertically Raised or Lowered Text)	§2.3.2.22
rFonts (Run Fonts)	§2.3.2.24
rStyle (Referenced Character Style)	§2.3.2.27
rtl (Right To Left Text)	§2.3.2.28
shadow (Shadow)	§2.3.2.29
shd (Run Shading)	§2.3.2.30
smallCaps (Small Caps)	§2.3.2.31
snapToGrid (Use Document Grid Settings For Inter-Character Spacing)	§2.3.2.32
spacing (Character Spacing Adjustment)	§2.3.2.33
specVanish (Paragraph Mark Is Always Hidden)	§2.3.2.34
strike (Single Strikethrough)	§2.3.2.35
sz (Font Size)	§2.3.2.36
szCs (Complex Script Font Size)	§2.3.2.37
u (Underline)	§2.3.2.38
vanish (Hidden Text)	§2.3.2.39
vertAlign (Subscript/Superscript Text)	§2.3.2.40
w (Expanded/Compressed Text)	§2.3.2.41

Child Elements	Subclause
webHidden (Web Hidden Text)	§2.3.2.42

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RPrOriginal">
   <sequence>
      <group ref="EG_RPrBase" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

2.3.2.27 rStyle (Referenced Character Style)

This element specifies the style ID of the character style which shall be used to format the contents of this paragraph.

This formatting is applied at the following location in the *style hierarchy*:

- Document defaults
- Table styles
- Numbering styles
- Paragraph styles
- Character styles (this element)
- Direct Formatting

This means that all properties specified in the style element (§2.7.3.17) with a styleId which corresponds to the value in this element's val attribute are applied to the run at the appropriate level in the hierarchy.

If this element is omitted, or it references a style which does not exist, then no character style shall be applied to the current paragraph. As well, this property is ignored if the run properties are part of a character style.

[*Example*: Consider the following WordprocessingML fragment:

```
<w:rPr>
  <w:pStyle w:val=”TestCharacterStyle” />
  <w:b />
  <w:i />
</w:rPr>
```

This run specifies that it will inherit all of the run properties specified by the paragraph style with a styleId of `TestCharacterStyle`, which will then have any bold or italics settings overridden and set to be applied to the run. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.3.2.28 rtl (Right To Left Text)

This element specifies that the alignment and reading order for this run shall be right to left. This setting determines the way in which the run contents are presented in the document when punctuation characters are part of the run's contents. When this property is specified, each part of the run between a punctuation mark shall be laid out right to left on the line.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then right to left alignment shall not be applied to the contents of this run.

[*Example*: Consider the following run of English text: `This is a list: one, two, three.` Typically, each region of text between a punctuation mark is presented in a left to right reading order in the document (as shown above). If the right to left property is set on this text, as follows:

```
<w:rPr>
  <w:rtl v:val="on"/>
</w:rPr>
```

This run shall now have a right to left reading order, as follows:

```
.three ,two ,one :This is a list
```

In a right to left language, this would result in the correct placement of the punctuation with respect to each region of right to left text. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.2.29 shadow (Shadow)

This element specifies that the contents of this run shall be displayed as if each character has a shadow, displayed beneath the text and to its right.

This formatting property is a *toggle property*, which specifies that its behavior differs between its use within a style definition and its use as direct formatting. When used as part of a style definition, setting this property shall toggle the current state of that property as specified up to this point in the hierarchy (i.e. applied to not applied, and vice versa). Setting it to false (or an equivalent) shall result in the current setting remaining

unchanged. However, when used as direct formatting, setting this property to `true` or `false` shall set the absolute state of the resulting property.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then shadowing shall not be applied to the contents of this run.

This element shall not be present with either the emboss (§2.3.2.11) or imprint (§2.3.2.16) properties on the same run, since they are mutually exclusive in terms of appearance.

[*Example*: Consider a run of text which shall have the shadow property explicitly turned on for the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:shadow w:val="true"/>
</w:rPr>
```

This run explicitly declares that the shadow property is `true`, so the contents of this run will appear with a shadow. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.2.30 shd (Run Shading)

Like paragraph shading, this element specifies the shading applied to the contents of the run.

This shading consists of three components:

- Background Color
- (optional) Pattern
- (optional) Pattern Color

The resulting shading is applied by setting the background color behind the paragraph, then applying the pattern color using the mask supplied by the pattern over that background.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then run shading shall not be applied to the contents of this run.

[*Example*: Consider a run which shall have a background consisting of a theme color `accent6` with a theme color `text2` overlaid using a 20% fill pattern. This requirement is specified using the following WordprocessingML:

```
<w:pPr>
  <w:shd w:val="pct20" w:themeColor="text2" w:themeFill="accent6" />
</w:pPr>
```

The resulting run will use the background color `accent6` under the foreground pattern color `text2` as specified by the `pct20` pattern mask. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
color (Shading Pattern Color)	Specifies the color used for any foreground pattern specified for this shading using the val attribute. This color may either be presented as a hex value (in RRGGBB format), or `auto` to allow a consumer to automatically determine the foreground shading color as appropriate. If the shading style (the val attribute) specifies the use of no shading format or is omitted, then this property has no effect. Also, if the shading specifies the use of a theme color via the themeColor attribute, then this value is superseded by the theme color value. If this attribute is omitted, then its value shall be assumed to be `auto`.

Attributes	Description
	[*Example*: Consider a shading of type `pct20` with a foreground color value of `auto`, as follows: `<w:shd w:val="pct20"... w:color="auto"/>` The foreground color for this shading pattern therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the shading color can be distinguished against the page's background color. *end example*] The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
fill (Shading Background Color)	Specifies the color used for the background for this shading. This color may either be presented as a hex value (in RRGGBB format), or `auto` to allow a consumer to automatically determine the background shading color as appropriate. If this attribute is omitted, then its value shall be assumed to be `auto`. [*Example*: Consider a shading using a background color of hex value C3D69B, using the following WordprocessingML: `<w:shd w:fill="C3D69B" />` The background color for this shading therefore will be a color with a hex value of `C3D69B`. *end example*] If the shading specifies the use of a theme color via the themeFill attribute, then this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
themeColor (Shading Pattern Theme Color)	Specifies a theme color which should be applied to any foreground pattern specified for this shading using the val attribute. The specified theme color is a reference to one of the predefined theme colors, located in the document's themes part, which allows for color information to be set centrally in the document. If this element is omitted, then no theme color is applied, and the color attribute shall be used to determine the shading pattern color. [*Example*: Consider a paragraph which shall have a background consisting of a theme color `accent3` with a theme color `accent6` overlaid using a 20% fill pattern. This requirement is specified using the following WordprocessingML:

Attributes	Description
	<w:pPr> <w:shd w:val="pct20" w:themeColor="accent6" w:themeFill="accent3" /> </w:pPr> The resulting paragraph will use the foreground pattern color accent6 in the region specified by the pct20 pattern mask. *end example*] The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).
themeFill (Shading Background Theme Color)	Specifies a theme color which should be applied to the background for this shading. The specified theme color is a reference to one of the predefined theme colors, located in the document's themes part, which allows for color information to be set centrally in the document. If this element is omitted, then no theme color is applied, and the color attribute shall be used to determine the shading background color. [*Example*: Consider a paragraph which shall have a background consisting of a theme color accent3 with a theme color accent6 overlaid using a 20% fill pattern. This requirement is specified using the following WordprocessingML: <w:shd w:val="pct20" w:themeColor="accent6" w:themeFill="accent3" /> The resulting shading will use the background color specified by the accent3 theme color. *end example*] The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).
themeFillShade (Shading Background Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this shading color. If the themeFillShade is supplied, then it is applied to the RGB value of the themeFill color (from the theme part) to determine the final color applied to this border. The themeFillShade value is stored as a hex encoding of the shade value (from 0 to 255) applied to the current border. [*Example*: Consider a shade of 40% applied to a background shading color in a document. This shade is calculated as follows: $S_{xml} = 0.4 * 255$ $= 102$ $= 66(hex)$

Attributes	Description
	The resulting themeFillShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$L' = L * Shade_{percentage}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L' = 0.53 * 0.75$$ $$= 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting shading's fill attribute: `<w:shd w:fill="943634" w:themeFill="accent2" w:themeFillShade="BF" />` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeFillTint (Shading Background Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this shading instance. If the themeFillTint is supplied, then it is applied to the RGB value of the themeFill color (from the theme part) to determine the final color applied to this border. The themeFillTint value is stored as a hex encoding of the tint value (from 0 to 255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows:

Attributes	Description
	$$T_{xml} = 0.6 * 255$$ $$= 153$$ $$= 99(hex)$$ The resulting themeFillTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6)$$ $$= 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$ and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting shading's fill attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:fill="95B3D7" w:themeFillColor="accent2" w:themeFillTint="99" />` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeShade (Shading Pattern Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this shading color. If the themeShade is supplied, then it is applied to the RGB value of the themeColor color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0 to 255) applied to the current border.

Attributes	Description
	[*Example*: Consider a shade of 40% applied to a background shading color in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255$$ $$= 102$$ $$= 66(hex)$$ Te resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}^{'} = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L^{'} = 0.53 * 0.75$$ $$= 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:shd w:color="943634" w:themeColor="accent2" w:themeShade="BF" />` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Shading Pattern Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this shading instance. If the themeTint is supplied, then it is applied to the RGB value of the themeColor color (from the theme part) to determine the final color applied to this border.

Attributes	Description
	The themeTint value is stored as a hex encoding of the tint value (from 0 to 255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255 \\ = 153 \\ = 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6) \\ = 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$ and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting shading's color attribute: `<w:shd w:color="95B3D7" w:themeColor="accent2" w:themeTint="99" />` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Shading Pattern)	Specifies the pattern which shall be used to lay the pattern color over the background color for this paragraph shading.

Attributes	Description
	This pattern consists of a mask which is applied over the background shading color to get the locations where the pattern color should be shown. Each of these possible masks are shown in the simple type values referenced below. [*Example*: Consider a shaded paragraph which uses a 10 percent foreground fill, resulting in the following WordprocessingML: `<w:shd w:val="pct10" .../>` This shading val is `pct10`, indicating that the border style is a 10 percent foreground fill mask. *end example*] The possible values for this attribute are defined by the ST_Shd simple type (§2.18.85).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Shd">
   <attribute name="val" type="ST_Shd" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="fill" type="ST_HexColor" use="optional"/>
   <attribute name="themeFill" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeFillTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeFillShade" type="ST_UcharHexNumber" use="optional"/>
</complexType>
```

2.3.2.31 smallCaps (Small Caps)

This element specifies that all small letter characters in this text run shall be formatted for display only as their capital letter character equivalents in a font size two points smaller than the actual font size specified for this text. This property does not affect any non-alphabetic character in this run, and does not change the Unicode character for lowercase text, only the method in which it is displayed. If this font cannot be made two point smaller than the current size, then it shall be displayed as the smallest possible font size in capital letters.

This formatting property is a *toggle property*, which specifies that its behavior differs between its use within a style definition and its use as direct formatting. When used as part of a style definition, setting this property shall toggle the current state of that property as specified up to this point in the hierarchy (i.e. applied to not applied, and vice versa). Setting it to `false` (or an equivalent) shall result in the current setting remaining unchanged. However, when used as direct formatting, setting this property to `true` or `false` shall set the absolute state of the resulting property.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the characters are not formatted as capital letters.

This element shall not be present with the caps (§2.3.2.4) property on the same run, since they are mutually exclusive in terms of appearance.

[*Example*: Consider the words `Hello World`, which shall be displayed in small capital letters in a document. This constraint is specified as follows in the WordprocessingML:

```
<w:r>
  <w:rPr>
    <w:sz w:val="24" />
    <w:smallCaps w:val="true" />
  </w:rPr>
  <w:t>Hello World</w:t>
</w:r>
```

This run will display using a 12 point capital letter for the capital letter H and W, and a 10 point capital letter for the lowercase letters in the run, even though the lowercase characters are used in actual run contents. If this property is removed, the original character forms will be displayed (they are not lost). *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.2.32 snapToGrid (Use Document Grid Settings For Inter-Character Spacing)

This element specifies whether the current run should use the document grid characters per line settings defined in the docGrid element (§2.6.5) when laying out the contents in this run. This setting determines whether the additional character pitch specified in the document grid shall be added to each character in this run as specified by the document grid.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the run shall use the document grid setting to lay out text when a document grid is defined for the parent section.

[*Example*: Consider two runs in a section with a document grid set to allow 20 characters per line. This document grid would effectively specifies that an additional character pitch shall be added to each line in order to ensure that the resulting line contains only 20 East Asian characters.

If this property is set on the first run, but turned off on the second run, as follows:

```
<w:r>
  <w:t>Run One</w:t>
</w:r>
<w:r>
  <w:rPr>
    <w:snapToGrid w:val="off" />
  </w:rPr>
  <w:t>Run Two</w:t>
</w:r>
```

The resulting document shall have the required additional character pitch added to each character in run one, but zero additional character pitch added to each character in run two, since the snapToGrid property is turned off. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off.

Attributes	Description
	[*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.2.33 spacing (Character Spacing Adjustment)

This element specifies the amount of character pitch which shall be added or removed after each character in this run before the following character is rendered in the document. This property has an effect equivalent to the additional character pitched added by a document grid applied to the contents of a run.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the run shall not have any additional character pitch applied to any character in its contents.

[*Example*: Consider a run of text which shall have ten points of additional character spacing explicitly added to each character within the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:spacing w:val="200"/>
</w:rPr>
```

This run explicitly declares that the spacing value is 200, so the contents of this run will appear as if they have 10 additional points of spacing added between them. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
val (Positive or Negative Value in Twentieths of a Point)	Specifies a value whose contents shall contain a positive whole number, whose contents consist of a positive or negative measurement in twentieths of a point (equivalent to 1/1440th of an inch). The contents of this measurement shall be interpreted based on the context of the

Attributes	Description
	parent XML element. [*Example*: Consider an attribute value of `-720` whose type is ST_SignedTwipsMeasure. This attribute value specifies a value of negative one-half of an inch or -36 points (`-720` twentieths of a point = -36 points = -0.5 inches). *end example*] The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§2.18.88).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SignedTwipsMeasure">
    <attribute name="val" type="ST_SignedTwipsMeasure" use="required"/>
</complexType>
```

2.3.2.34 specVanish (Paragraph Mark Is Always Hidden)

This element specifies that the given run shall always behave as if it is hidden, even when hidden text is being displayed in the current document.

This property shall only be used to specify that a paragraph mark shall never be used to break the end of a paragraph for display, even if it is being shown on the document, as would be the case if a regularly hidden paragraph was not being displayed in the document. [*Note*: This property was typically used to ensure that a paragraph style can be applied to a part of a paragraph, and still appear as in the Table of Contents (which in previous word processors would ignore the use of the style if it were being used as a character style. *end note*] If this element is applied to any other run, it may be ignored.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the run properties for the paragraph mark shall not always be treated as if hidden.

[*Example*: Consider a paragraph mark which never be used to break the end of the paragraph in the document. This constraint is specified using the following WordprocessingML:

```
<w:pPr>
  <w:rPr>
    <w:specVanish />
  </w:rPr>
</w:pPr>
```

The presence of the specVanish element means that this paragraph mark shall always be treated as hidden (shall never be used to end the paragraph for display), but may be used to mark the end of use of a paragraph style. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30);

Parent Elements
rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.2.35 strike (Single Strikethrough)

This element specifies that the contents of this run shall be displayed with a single horizontal line through the center of the line.

This formatting property is a *toggle property*, which specifies that its behavior differs between its use within a style definition and its use as direct formatting. When used as part of a style definition, setting this property shall toggle the current state of that property as specified up to this point in the hierarchy (i.e. applied to not applied, and vice versa). Setting it to `false` (or an equivalent) shall result in the current setting remaining unchanged. However, when used as direct formatting, setting this property to `true` or `false` shall set the absolute state of the resulting property.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then strikethrough shall not be applied to the contents of this run.

This element shall not be present with the dstrike (§2.3.2.7) property on the same run, since they are mutually exclusive in terms of appearance.

[*Example*: Consider a run of text which shall have the strike property explicitly turned on for the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:strike w:val="true"/>
</w:rPr>
```

This run explicitly declares that the strike property is `true`, so the contents of this run will have a single horizontal strikethrough line. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.2.36 sz (Font Size)

This element specifies the font size which shall be applied to all non complex script characters in the contents of this run when displayed.

If this element is not present, the default value is to leave the value applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then any appropriate font size may be used for non complex script characters.

[*Example*: Consider a run of text which shall have an explicit font size of 13.5 points for the non complex script contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:sz w:val="27"/>
</w:rPr>
```

This run explicitly declares that the sz property is 27 half-point for the non-complex script contents of this run, so the text will be displayed in 13.5 point font size. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description

val (Half Point Measurement)	Specifies a positive measurement specified in half-points (1/144 of an inch).
	
	The contents of this attribute value are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:rPr>`
	`  <w:sz w:val="28" />`
	`</w:rPr>`
	
	The value of the val attribute is the font size of the run's contents.
	
	However, consider the following fragment:
	
	`<w:rPr>`
	`  <w:kern w:val="30" />`
	`</w:rPr>`
	
	In this case, the value in the val attribute is the minimum size for which font characters shall be automatically kerned.
	
	In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_HpsMeasure simple type (§2.18.48).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HpsMeasure">
   <attribute name="val" type="ST_HpsMeasure" use="required"/>
</complexType>
```

2.3.2.37 szCs (Complex Script Font Size)

This element specifies the font size which shall be applied to all complex script characters in the contents of this run when displayed.

If this element is not present, the default value is to leave the value applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then any appropriate font size may be used for complex script characters.

[*Example*: Consider a run of text which shall have an explicit font size of 10 points for the complex script contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:szCs w:val="20"/>
</w:rPr>
```

This run explicitly declares that the sz property is 20 half-point for the non-complex script contents of this run, so the text will be displayed in 10 point font size. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description

val (Half Point Measurement)	Specifies a positive measurement specified in half-points (1/144 of an inch).
	
	The contents of this attribute value are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:rPr>`
	`  <w:sz w:val="28" />`
	`</w:rPr>`
	
	The value of the val attribute is the font size of the run's contents.
	
	However, consider the following fragment:
	
	`<w:rPr>`
	`  <w:kern w:val="30" />`
	`</w:rPr>`
	
	In this case, the value in the val attribute is the minimum size for which font characters shall be automatically kerned.

Attributes	Description
	In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_HpsMeasure simple type (§2.18.48).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HpsMeasure">
   <attribute name="val" type="ST_HpsMeasure" use="required"/>
</complexType>
```

2.3.2.38 u (Underline)

This element specifies that the contents of this run should be displayed along with an underline appearing directly below the character height (less all spacing above and below the characters on the line).

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then an underline shall not be applied to the contents of this run.

[*Example*: Consider a run of text which shall have a double underline explicitly turned on for the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:u w:val="double"/>
</w:rPr>
```

This run explicitly declares an underline using the u property. The val of that underline is `double`, so the style of the underline on this run shall be a double line. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
color (Underline Color)	Specifies the color for the underlining on this run. This color may either be presented as a hex value (in RRGGBB format), or `auto` to allow a consumer to automatically determine the underline color as appropriate. If the underline specifies the use of a theme color via the themeColor attribute, then this value is superseded by the theme color value. [*Example*: Consider a run color with value `auto`, as follows:

Attributes	Description
	<w:rPr> <w:un … w:color="auto" /> </w:rPr> This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the underline can be distinguished against the page's background color. *end example*] The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
themeColor (Underline Theme Color)	Specifies a theme color which should be applied to the current underline. The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows for color information to be set centrally in the document. If the themeColor attribute is specified, then the color attribute is ignored for this underline. [*Example*: Consider an underlined run of text whose underline should be displayed using the accent3 theme color from the document's Theme part. This requirement would be specified as follows in the resulting WordprocessingML: <w:rPr> <w:u … w:themeColor="accent3" /> </w:rPr> The themeColor attribute specifies that the underline shall use the accent3 theme color. *end example*] The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).
themeShade (Underline Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this underline. If the themeShade is supplied, then it is applied to the RGB value of the theme color to determine the final color applied to this underline. The themeShade value is stored as a hex encoding of the shade value (from 0 to 255) applied to the current border. [*Example*: Consider a shade of 40% applied to a underline in a document. This shade is calculated as follows: $S_{xml} = 0.4 * 255$ $= 102$ $= 66(hex)$

Attributes	Description
	The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$L' = L * Shade_{percentage}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L' = 0.53 * 0.75$$ $$= 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting underline's color attribute: `<w:u w:color="943634" w:themeColor="accent2" w:themeShade="BF" />` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Underline Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this underline's contents. If the themeTint is supplied, then it is applied to the RGB value of the theme color to determine the final color applied to this run. The themeTint value is stored as a hex encoding of the tint value (from 0 to 255) applied to the current border. [*Example*: Consider a tint of 60% applied to an underline in a document. This tint is calculated as follows:

Attributes	Description
	$$T_{xml} = 0.6 * 255$$ $$= 153$$ $$= 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6)$$ $$= 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$ and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting underline formatting's WordprocessingML color attribute: `<w:u … w:color="95B3D7" w:themeColor="accent2" w:themeTint="99" />` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Underline Style)	Specifies the pattern which shall be used to create the underline applied beneath the text in this run. Each of these possible patterns are shown in the simple type referenced below. [*Example*: Consider a run of text which shall have a double underline explicitly turned on for the contents of the run. This constraint is specified using the following WordprocessingML:

Attributes	Description
	<w:rPr> <w:u w:val="double"/> </w:rPr> The val of the underline on this run is double, so the style of the underline on this run shall be a double line. *end example*] The possible values for this attribute are defined by the ST_Underline simple type (§2.18.107).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Underline">
   <attribute name="val" type="ST_Underline" use="optional"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
</complexType>
```

2.3.2.39 vanish (Hidden Text)

This element specifies whether the contents of this run shall be hidden from display at display time in a document. [*Note*: The setting should affect the normal display of text, but an application may have settings to force hidden text to be displayed. *end note*]

This formatting property is a *toggle property*, which specifies that its behavior differs between its use within a style definition and its use as direct formatting. When used as part of a style definition, setting this property shall toggle the current state of that property as specified up to this point in the hierarchy (i.e. applied to not applied, and vice versa). Setting it to `false` (or an equivalent) shall result in the current setting remaining unchanged. However, when used as direct formatting, setting this property to `true` or `false` shall set the absolute state of the resulting property.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy* .If this element is never applied in the style hierarchy, then this text shall not be hidden when displayed in a document.

[*Example*: Consider a run of text which shall have the hidden text property turned on for the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:vanish />
</w:rPr>
```

This run declares that the vanish property is set for the contents of this run, so the contents of this run will be hidden when the document contents are displayed. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.2.40 vertAlign (Subscript/Superscript Text)

This element specifies the alignment which shall be applied to the contents of this run in relation to the default appearance of the run's text. This allows the text to be repositioned as subscript or superscript without altering the font size of the run properties.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the text shall not be subscript or superscript relative to the default baseline location for the contents of this run.

[*Example*: Consider a run which shall be positioning as superscript when displaying its contents. This requirement would be specified using the following WordprocessingML:

```
<w:rPr>
  <w:vertAlign w:val="superscript" />
</w:rPr>
```

The resulting run is positioned as superscript, therefore it is rendered in a smaller size above the default baseline location for the contents of the run. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description

val (Subscript/Superscript Value)	Specifies the type of vertical alignment applied to the contents of the current run.
	
	[*Example*: Consider a run which shall be positioning as superscript when displaying its contents. This requirement would be specified using the following WordprocessingML:
	
	`<w:rPr>`
	`  <w:vertAlign w:val="superscript" />`
	`</w:rPr>`
	
	The value of the val attribute is `superscript`, therefore the run's contents are rendered in a smaller size above the default baseline location for the contents of the run. *end example*]
	
	The possible values for this attribute are defined by the ST_VerticalAlignRun simple type (§2.18.110).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_VerticalAlignRun">
   <attribute name="val" type="ST_VerticalAlignRun" use="required"/>
</complexType>
```

2.3.2.41 w (Expanded/Compressed Text)

This element specifies the amount by which each character shall be expanded or when the character is rendered in the document. This property has an of stretching or compressing each character in the run, as opposed to the spacing element (§2.3.2.33) which expands/compresses the text by adding additional character pitch but not changing the width of the actual characters displayed on the line.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the run shall be displayed at 100% of its normal width.

[*Example*: Consider a run of text which shall be expanded to 200% of its normal width when displaying each character within the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:w w:val="200"/>
</w:rPr>
```

This run explicitly declares that the w value is 200, so the contents of this run will appear at 200% of their normal character width by stretching the width of each character. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description

val (Text Expansion/Compression Value)	Specifies that the percentage by which the contents of this run shall be expanded or compressed with respect to its normal (100%) character width.
	
	If this attribute is omitted, then the contents of this run shall be displayed at 100% of its normal size.
	
	[*Example*: Consider a run of text which shall be compressed to 200% when displaying each character within the contents of the run. This constraint is specified using the following WordprocessingML:
	
	`<w:rPr>`
	`  <w:w w:val="50"/>`
	`</w:rPr>`
	
	This run explicitly declares that the w value is 50, so the contents of this run will appear at 50% of their normal character width by compressing the width of each character. *end example*]
	
	The possible values for this attribute are defined by the ST_TextScale simple type (§2.18.102).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextScale">
   <attribute name="val" type="ST_TextScale"/>
</complexType>
```

2.3.2.42 webHidden (Web Hidden Text)

This element specifies whether the contents of this run shall be hidden from display at display time in a document when the document is being displayed in a web page view. [*Note*: The setting should affect the normal display of text in a web page view, but an application may have settings to force hidden text to be displayed. *end note*] As well, this setting should not affect a normal paginated view of the document.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy* .If this element is never applied in the style hierarchy, then this text shall not be hidden when displayed in a document in a web page view.

[*Example*: Consider a run of text which shall have the hidden text property turned on for the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:webHidden />
</w:rPr>
```

This run declares that the webHidden property is set for the contents of this run, so the contents of this run will be hidden when the document contents are displayed in a web page view. *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.3.1.29); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.3.2.26); rPr (§2.7.4.4); rPr (§2.3.1.30); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.3 Run Content

The final level of the document hierarchy is *run content*, which is defined as the set of elements which can be contained as the contents of a particular run in a document.

[*Note*: Types of run content in WordprocessingML include:

- Text
- Field Codes
- DrawingML objects

- VML objects
- Fields

end note]

[*Example*: Consider the following run within a WordprocessingML document:

```
<w:r>
  <w:rPr>
    <w:b/>
    <w:i/>
  </w:rPr>
  <w:t>quick</w:t>
</w:r>
```

The run content consists of a single string of run text inside the t element, which reads `quick`. *end example*]

2.3.3.1 br (Break)

This element specifies that a break shall be placed at the current location in the run content. A *break* is a special character which is used to override the normal line breaking that would be performed based on the normal layout of the document's contents. [*Example*: Normal breaking for English would occur only after a breaking space or optional hyphen character. *end example*]

The behavior of this break character (the location where text shall be restarted after this break) shall be determined by its type and clear attribute values, described below.

[*Example*: Consider the following sentence in a WordprocessingML document:

```
This is a simple sentence.
```

Normally, just as shown above, this sentence would be displayed on a single line as it is not long enough to require line breaking (given the width of the current page). However, if a text wrapping break character (a typical line break) were inserted after the word `is`, as follows:

```
<w:r>
  <w:t>This is</w:t>
  <w:br/>
  <w:t xml:space="preserve"> a simple sentence.</w:t>
</w:r>
```

This would imply that this break shall be treated as a simple line break, and break the line after that word:

```
This is
 a simple sentence.
```

The break character forced the following text to be restarted on the next available line in the document. *end example*]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

Attributes	Description
clear (Restart Location For Text Wrapping Break)	Specifies the location which shall be used as the next available line when the break's type attribute has a value of `textWrapping`. This property only affects the restart location when the current run is being displayed on a line which does not span the full text extents due to the presence of a floating object (see possible values for details). If this break is not of type `textWrapping`, then this attribute shall be ignored. If this attribute is omitted, then its value shall be assumed to be none if needed. [*Example*: Consider a text wrapping break character which should force the restart location to the next line which spans the full width of the text extents of the page (there are no floating objects which interrupt the line). This line break is of type `textWrapping`, since it shall only advance to the next line, but the clear value shall specify that this restart location shall ignore all lines which are not of the full line width by specifying a value of `all`, as follows: `<w:br w:type="textWrapping" w:clear="all" />` This break shall therefore not use the next available line, but rather the next available line ignoring all lines which do not span the full text width. *end example*] The possible values for this attribute are defined by the ST_BrClear simple type (§2.18.5).
type (Break Type)	Specifies the type of the current break. The break type determines the next location where text shall be placed after this manual break is applied to the text contents (see possible values for details). If this attribute is omitted, then it shall be assumed to be of type `textWrapping`. [*Example*: Consider a manual break which shall advance the text to the next text column in the document, rather than just the next available line. This break would therefore be specified as follows: `<w:br w:type=”column”/>` The type attribute specifies a value of `column`, which means that the break shall force the next character in the document to be restarted on the next line in a new text column in the document. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_BrType simple type (§2.18.6).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Br">
   <attribute name="type" type="ST_BrType" use="optional"/>
   <attribute name="clear" type="ST_BrClear" use="optional"/>
</complexType>
```

2.3.3.2 control (Floating Embedded Control)

This element specifies that the parent VML object is a representation of an embedded control at the current location in the document. This element shall be used to associate the VML data with the appropriate embedded control settings and properties when the document is displayed.

If the embedded control is not present, cannot be loaded due to application settings, or is not supported, then the VML data shall be used to provide an image representation of the control at the appropriate location in the document.

[*Example*: Consider a run which consists of an embedded control. That run would be specified using the following WordprocessingML:

```
<w:r>
  <w:pict>
    …
    <w:control r:id=”rId99” w:shapeid=”shape01” … />
  </w:pict>
</w:r>
```

The control element indicates that the parent VML object contains the positioning and last known image representation of an embedded control, whose settings and properties are stored on this element. *end example*]

Parent Elements
pict (§2.3.3.21); pict (§2.9.23)

Attributes	Description
id (Embedded Control Properties Relationship Reference) Namespace: …/officeDocument/2006/relationshi	Specifies the relationship ID for the relationship which contains the properties for this embedded control. This property bag is contained in a separate part within the Word Open XML package. The relationship explicitly targeted by this attribute shall be of type http://schemas.openxmlformats.org/officeDocument/2006/relationships/control or the document shall be conisdered non-conformant.

Attributes	Description
ps	If this attribute is omitted, then the embedded control shall be given no property bag when instantiated. [*Example*: Consider the following WordprocessingML markup for an embedded control in a document: `<w:control r:id="rId5" w:id="CheckBox1" w:name="CheckBox1" w:shapeid="_x0000_s1027" w:class="shape" w:w="145" w:h="28" w:align="left" />` The id attribute in the relationship reference namespace specifies that the relationship with relationship ID `rId5` shall contain the property data for this embedded control. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
name (Unique Name for Embedded Control)	Specifies a unique name for this embedded control. This name shall be unique across all controls in this document. [*Example*: Consider the following WordprocessingML markup for an embedded control in a document: `<w:control r:id="rId5" w:id="CheckBox1" w:name="CheckBox1" w:shapeid="_x0000_s1027" w:class="shape" w:w="145" w:h="28" w:align="left" />` The name attribute specifies that the unique name for this control shall be `CheckBox1`. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).
shapeid (Associated VML Data Reference)	Specifies the shape ID for a shape which shall be used to define the presentation and location of this embedded control within the document if the control is floating using the VML syntax. [*Note*: This positioning data is sufficient to display the control in any case where: • The embedded control is not on the current machine • Embedded controls are disabled • Embedded controls of this type are not supported *end note*] This shape ID reference is resolved by looking for a VML shape element (§6.1.2.19) whose id attribute matches the value specified within this attribute. If no such shape exists, then the control shall be rendered inline in the document content at the current run content location.

Attributes	Description
	If this attribute is omitted, then this embedded control shall be displayed inline in the current location in the parent run. [*Example*: Consider the following WordprocessingML markup for an embedded control in a document: `<w:control r:id="rId5" w:id="CheckBox1" w:name="CheckBox1" w:shapeid="_x0000_s1027" w:class="shape" w:w="145" w:h="28" w:align="left" />` The shapeid attribute specifies that the VML shape element with a shape id attribute value of `_x0000_s1027` shall contain the VML positioning data for this embedded control. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Control">
   <attribute name="name" type="ST_String" use="optional"/>
   <attribute name="shapeid" type="ST_String" use="optional"/>
   <attribute ref="r:id" use="optional"/>
</complexType>
```

2.3.3.3 control (Inline Embedded Control)

This element specifies that the parent embedded object is a representation of an embedded control at the current location in the document. This element shall be used to associate the appropriate embedded control settings and properties when the document is displayed.

If the embedded control is not present, cannot be loaded due to application settings, or is not supported, then a suitable placeholder image shall be used to provide a representation of the presence of an embedded control at the appropriate location in the document.

[*Example*: Consider a run which consists of an embedded control. That run would be specified using the following WordprocessingML:

```
<w:r>
  <w:object>
    …
    <w:control r:id=”rId99” w:shapeid=”shape01” … />
  </w:object>
</w:r>
```

The control element indicates that the parent embedded object is an embedded control, whose settings and properties are stored on this element and the (optional) target of the relationship specified using the id attribute. *end example*]

Parent Elements
object (§2.3.3.19)

Attributes	Description
id (Embedded Control Properties Relationship Reference) Namespace: .../officeDocument/2006/relationships	Specifies the relationship ID for the relationship which contains the properties for this embedded control. This property bag is contained in a separate part within the Word Open XML package. The relationship explicitly targeted by this attribute shall be of type http://schemas.openxmlformats.org/officeDocument/2006/relationships/control or the document shall be conisdered non-conformant. If this attribute is omitted, then the embedded control shall be given no property bag when instantiated. [*Example*: Consider the following WordprocessingML markup for an embedded control in a document: `<w:control r:id="rId5" w:id="CheckBox1" w:name="CheckBox1" w:shapeid="_x0000_s1027" w:class="shape" w:w="145" w:h="28" w:align="left" />` The id attribute in the relationship reference namespace specifies that the relationship with relationship ID `rId5` shall contain the property data for this embedded control. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
name (Unique Name for Embedded Control)	Specifies a unique name for this embedded control. This name shall be unique across all controls in this document. [*Example*: Consider the following WordprocessingML markup for an embedded control in a document: `<w:control r:id="rId5" w:id="CheckBox1" w:name="CheckBox1" w:shapeid="_x0000_s1027" w:class="shape" w:w="145" w:h="28" w:align="left" />` The name attribute specifies that the unique name for this control shall be `CheckBox1`. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).
shapeid (Associated VML Data Reference)	Specifies the shape ID for a shape which shall be used to define the presentation and location of this embedded control within the document if the control is floating using the VML syntax.

Attributes	Description
	[*Note*: This positioning data is sufficient to display the control in any case where: • The embedded control is not on the current machine • Embedded controls are disabled • Embedded controls of this type are not supported *end note*] This shape ID reference is resolved by looking for a VML shape element (§6.1.2.19) whose id attribute matches the value specified within this attribute. If no such shape exists, then the control shall be rendered inline in the document content at the current run content location. If this attribute is omitted, then this embedded control shall be displayed inline in the current location in the parent run. [*Example*: Consider the following WordprocessingML markup for an embedded control in a document: `<w:control r:id="rId5" w:id="CheckBox1" w:name="CheckBox1" w:shapeid="_x0000_s1027" w:class="shape" w:w="145" w:h="28" w:align="left" />` The shapeid attribute specifies that the VML shape element with a shape id attribute value of `_x0000_s1027` shall contain the VML positioning data for this embedded control. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Control">
   <attribute name="name" type="ST_String" use="optional"/>
   <attribute name="shapeid" type="ST_String" use="optional"/>
   <attribute ref="r:id" use="optional"/>
</complexType>
```

2.3.3.4 cr (Carriage Return)

This element specifies that a carriage return shall be placed at the current location in the run content. A *carriage return* is the equivalent of Unicode character 000D, and is used to end the current line of text in WordprocessingML.

The behavior of a carriage return in run content shall be identical to a break character with null type and clear attributes, which shall end the current line and find the next available line on which to continue.

[*Example*: Consider the following sentence in a WordprocessingML document:

```
This is another simple sentence.
```

Normally, just as shown above, this sentence would be displayed on a single line as it is not long enough to require line breaking (given the width of the current page). However, if a carriage return were inserted after the word another, as follows:

```
<w:r>
  <w:t>This is another</w:t>
  <w:cr/>
  <w:t xml:space="preserve"> simple sentence.</w:t>
</w:r>
```

This would imply that this carriage return character shall force a line break, and break the line after that word:

```
This is another
 simple sentence.
```

The carriage return character forced the following text to be restarted on the next available line in the document. *end example*]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

2.3.3.5 dayLong (Date Block - Long Day Format)

This element specifies the presence of a date block at the current location in the run content. A *date block* is a non-editable region of text which shall display the current date filtered through the specified date picture (see following paragraphs) . [*Note*: The date block is a legacy construct used for compatibility with older word processors, and should not be produced unless it was consumed while reading a document – it is recommended that the DATE field is used in its place. *end note*]

A date block shall be displayed using the primary editing language of the host application, regardless of the languages specified in the parent run's lang property (§2.3.2.18).

The long day format date block shall use a date picture of DDDD, retrieving the long day format for the primary editing language.

[*Example*: Consider a WordprocessingML run with the following run content:

```
<w:r>
  <w:t xml:space="preserve">This is a long date: </w:t>
  <w:dayLong />
</w:r>
```

This run specifies that a long day format date block shall be placed after the text string literal `This is a long date:` in the document. Assuming that the host application's primary editing language is French (Canada) and today's date is 04/12/2006 (a Wednesday), this run would be displayed as follows:

```
This is a long date: mercredi
```

end example]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

2.3.3.6 dayShort (Date Block - Short Day Format)

This element specifies the presence of a date block at the current location in the run content. A *date block* is a non-editable region of text which shall display the current date filtered through the specified date picture (see following paragraphs) . [*Note*: The date block is a legacy construct used for compatibility with older word processors, and should not be produced unless it was consumed while reading a document – it is recommended that the DATE field is used in its place. *end note*]

A date block shall be displayed using the primary editing language of the host application, regardless of the languages specified in the parent run's lang property (§2.3.2.18).

The short day format date block shall use a date picture of DD, retrieving the short day format for the primary editing language.

[*Example*: Consider a WordprocessingML run with the following run content:

```
<w:r>
  <w:t xml:space="preserve">This is a short date: </w:t>
  <w:dayShort />
</w:r>
```

This run specifies that a short day format date block shall be placed after the text string literal `This is a short date:` in the document. Assuming that the host application's primary editing language is English (Canada) and today's date is 04/12/2006 (a Wednesday), this run would be displayed as follows:

```
This is a short date: 12
```

end example]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

2.3.3.7 delText (Deleted Text)

This element specifies that this run contains literal text which shall be displayed in the document. The `delText` element shall be used for all text runs which are part of a region of text that is contained in a deleted region using the del element (§2.13.5.12).

[*Example*: Consider a paragraph of WordprocessingML content which reads `This is deleted text`, where the words `deleted text` are part of a deleted region of the document. This paragraph would therefore be represented as follows:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">This is </w:t>
  </w:r>
  <w:del>
    <w:r>
      <w:delText>deleted text</w:delText>
    </w:r>
  </w:del>
</w:p>
```

The deleted text is contained in a delText node, while the regular text is contained in a t node. *end example*]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

Attributes	Description

space (Content Contains Significant Whitespace)
 	
Namespace: http://www.w3.org/XML/1998/namespace
	Specifies how white space should be handled for the contents of this element using the W3C space preservation rules.
	
	[*Example*: Consider the following run contained within a WordprocessingML document:
	
	`<w:r>`
	`  <w:t>significant whitespace  </w:t>`
	`</w:r>`
	
	Although there are three spaces on each side of the text content in the run, that whitespace has not been specifically marked as significant, therefore it is removed when this run is added to the document. *end example*]
	
	The possible values for this attribute are defined by the type in the namespace.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Text">
   <simpleContent>
      <extension base="ST_String">
         <attribute ref="xml:space" use="optional"/>
      </extension>
   </simpleContent>
</complexType>
```

2.3.3.8 dirty (Invalidated Field Cache)

This element specifies that the field has been changed and the results shall be updated on open in a conforming consumer.

Parent Elements
rubyPr (§2.3.3.27)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.3.3.9 drawing (DrawingML Object)

This element specifies that a DrawingML object is located at this position in the run's contents. The layout properties of this DrawingML object are specified using the WordprocessingML Drawing syntax (§5.5).

[*Example*: Consider a run which consists of a picture which is in line with the text in that paragraph (i.e. on the line and affects the line height). That run would be specified using the following WordprocessingML:

```
<w:r>
  <w:drawing>
    <wp:inline>
      …
    </wp:inline>
  </w:drawing>
</w:r>
```

The drawing element indicates that a DrawingML object and its WordprocessingML Drawing positioning data are located at the current position in the run (e.g. a picture or a chart). *end example*]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

Child Elements	Subclause
anchor (Anchor for Floating DrawingML Object)	§5.5.2.3
inline (Inline DrawingML Object)	§5.5.2.8

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Drawing">
   <choice minOccurs="1" maxOccurs="unbounded">
      <element ref="wp:anchor" minOccurs="0"/>
      <element ref="wp:inline" minOccurs="0"/>
   </choice>
</complexType>
```

2.3.3.10 hps (Phonetic Guide Text Font Size)

This element specifies the font size which shall be applied to the phonetic guide text in the contents of this run when displayed.

If this element disagrees with the run properties on the phonetic guide text rt element (§2.3.3.23), then those properties shall be ignored and this element shall determine the size of the phonetic guide text.

[*Example*: Consider a run of phonetic guide text which shall have an explicit font size of 13.5 points. This constraint is specified using the following WordprocessingML:

```
<w:rubyPr>
  …
  <w:hps w:val="27"/>
  …
</w:rubyPr>
```

The hps property is 27 half-points for the ruby text in this run, so the phonetic guide text will be displayed in 13.5 point font size. *end example*]

Parent Elements
rubyPr (§2.3.3.27)

Attributes	Description

val (Half Point Measurement)	Specifies a positive measurement specified in half-points (1/144 of an inch).
	
	The contents of this attribute value are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:rPr>`
	`  <w:sz w:val="28" />`
	`</w:rPr>`
	
	The value of the val attribute is the font size of the run's contents.
	
	However, consider the following fragment:
	
	`<w:rPr>`
	`  <w:kern w:val="30" />`
	`</w:rPr>`
	
	In this case, the value in the val attribute is the minimum size for which font characters shall be automatically kerned.
	
	In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_HpsMeasure simple type (§2.18.48).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HpsMeasure">
   <attribute name="val" type="ST_HpsMeasure" use="required"/>
</complexType>
```

2.3.3.11 hpsBaseText (Phonetic Guide Base Text Font Size)

This element specifies the font size which shall be applied to the base text of this phonetic guide text when displayed. If this element disagrees with the run properties on the phonetic guide base text rubyBase element (§2.3.3.26), then this property shall be ignored and the sz element (§2.3.2.36) in that run shall determine the size of the phonetic guide base text.

[*Example*: Consider a run of phonetic guide base text which shall have an explicit font size of 30 points. This constraint is specified using the following WordprocessingML:

```
<w:rubyPr>
  …
  <w:hpsBaseText w:val="60"/>
  …
</w:rubyPr>
```

The hpsBaseText property is 60 half-points for the base text in this phonetic guide, so the phonetic guide base text will be displayed in 30 point font size. *end example*]

Parent Elements
rubyPr (§2.3.3.27)

Attributes	Description

val (Half Point Measurement)	Specifies a positive measurement specified in half-points (1/144 of an inch).
	
	The contents of this attribute value are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:rPr>`
	`  <w:sz w:val="28" />`
	`</w:rPr>`
	
	The value of the val attribute is the font size of the run's contents.
	
	However, consider the following fragment:
	
	`<w:rPr>`
	`  <w:kern w:val="30" />`
	`</w:rPr>`
	
	In this case, the value in the val attribute is the minimum size for which font characters shall be automatically kerned.
	
	In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_HpsMeasure simple type (§2.18.48).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HpsMeasure">
    <attribute name="val" type="ST_HpsMeasure" use="required"/>
</complexType>
```

2.3.3.12 hpsRaise (Distance Between Phonetic Guide Text and Phonetic Guide Base Text)

This element specifies the distance which shall be left between the phonetic guide base text and the phonetic guide text when this phonetic guide text is displayed.

[*Example*: Consider a run of phonetic guide text which shall have 10 points between the phonetic guide base text and the phonetic guide text. This constraint is specified using the following WordprocessingML:

```
<w:rubyPr>
  …
  <w:hpsRaise w:val="20"/>
  …
</w:rubyPr>
```

The hpsRaise property is 20 half-points for the phonetic guide, so the phonetic guide text will be displayed 10 points above the phonetic guide base text. *end example*]

Parent Elements
rubyPr (§2.3.3.27)

Attributes	Description

val (Half Point Measurement)	Specifies a positive measurement specified in half-points (1/144 of an inch).
	
	The contents of this attribute value are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:rPr>`
	`  <w:sz w:val="28" />`
	`</w:rPr>`
	
	The value of the val attribute is the font size of the run's contents.
	
	However, consider the following fragment:
	
	`<w:rPr>`
	`  <w:kern w:val="30" />`
	`</w:rPr>`
	
	In this case, the value in the val attribute is the minimum size for which font characters shall be automatically kerned.
	
	In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_HpsMeasure simple type

Attributes	Description
	(§2.18.48).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HpsMeasure">
   <attribute name="val" type="ST_HpsMeasure" use="required"/>
</complexType>
```

2.3.3.13 lastRenderedPageBreak (Position of Last Calculated Page Break)

This element specifies that this position delimited the end of a page when this document was last saved by an application which paginates its content.

[*Guidance*: This element shall be used by applications to specify the locations of page breaks within a document when it is saved as WordprocessingML, in order to allow other applications (e.g. assistive software) to utilize this information when reading the document. *end guidance*]

[*Example*: Consider a run which consists of the text `This is the end of the page`, where the word `end` was the last word on a page. If the application saving this file had paginated this content, that information may be saved with the file as follows:

```
<w:r>
  <w:t>This is the end</w:t>
  <w:lastRenderedPageBreak/>
  <w:t xml:space="preserve"> of the page</w:t>
</w:r>
```

The lastRenderedPageBreak element indicates that there was a page break resulting from pagination of this content, which occurred between the word `end` and the word `of`. *end example*]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

2.3.3.14 lid (Language ID for Phonetic Guide)

This element specifies the language which shall be for this phonetic guide.

[*Example*: Consider a run of phonetic guide text which is using Japanese as it language. This constraint is specified using the following WordprocessingML:

```
<w:rubyPr>
  …
  <w:lid w:val="ja-JP"/>
```

```
    …
</w:rubyPr>
```

The lid property is `ja-JP` for the phonetic guide, so the phonetic guide is specified to be Japanese. *end example*]

Parent Elements
rubyPr (§2.3.3.27)

Attributes	Description
val (Language Code)	Specifies an ISO 639-1 letter code or 4 digit hexadecimal code for a specific language. This code is interpreted in the context of the parent XML element. [*Example*: Consider an object which shall specify the English(Canada) language. That object would use the ISO 639-1 letter code of `en-CA` to specify this language. *end example*] The possible values for this attribute are defined by the ST_Lang simple type (§2.18.51).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Lang">
   <attribute name="val" type="ST_Lang" use="required"/>
</complexType>
```

2.3.3.15 monthLong (Date Block - Long Month Format)

This element specifies the presence of a date block at the current location in the run content. A *date block* is a non-editable region of text which shall display the current date filtered through the specified date picture (see following paragraphs) . [*Note*: The date block is a legacy construct used for compatibility with older word processors, and should not be produced unless it was consumed while reading a document – it is recommended that the DATE field is used in its place. *end note*]

A date block shall be displayed using the primary editing language of the host application, regardless of the languages specified in the parent run's lang property (§2.3.2.18).

The long month format date block shall use a date picture of `MMMM`, retrieving the long month format for the primary editing language.

[*Example*: Consider a WordprocessingML run with the following run content:

```
<w:r>
  <w:t xml:space="preserve">This is a long date: </w:t>
  <w:monthLong />
</w:r>
```

This run specifies that a long month format date block shall be placed after the text string literal `This is a long date:` in the document. Assuming that the host application's primary editing language is French (Canada) and today's date is 04/12/2006 (a Wednesday), this run would be displayed as follows:

```
This is a long date: avril
```

end example]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

2.3.3.16 monthShort (Date Block - Short Month Format)

This element specifies the presence of a date block at the current location in the run content. A *date block* is a non-editable region of text which shall display the current date filtered through the specified date picture (see following paragraphs). [*Note*: The date block is a legacy construct used for compatibility with older word processors, and should not be produced unless it was consumed while reading a document – it is recommended that the DATE field is used in its place. *end note*]

A date block shall be displayed using the primary editing language of the host application, regardless of the languages specified in the parent run's lang property (§2.3.2.18).

The short month format date block shall use a date picture of `MM`, retrieving the short month format for the primary editing language.

[*Example*: Consider a WordprocessingML run with the following run content:

```
<w:r>
  <w:t xml:space="preserve">This is a short date: </w:t>
  <w:monthShort />
</w:r>
```

This run specifies that a short month format date block shall be placed after the text string literal `This is a short date:` in the document. Assuming that the host application's primary editing language is English (Canada) and today's date is 04/12/2006 (a Wednesday), this run would be displayed as follows:

```
This is a short date: 04
```

end example]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

2.3.3.17 movie (Embedded Video)

This element specifies a location within a document where the specified parent image shall be treated as a static placeholder for an embedded movie. The specified movie file's contents should be displayed when requested at this location in the document. The location of the embedded movie to be displayed when supported shall be specified by the relationship whose Id attribute matches the id attribute on this element.

If the relationship type of the relationship specified by this element is not http://schemas.openxmlformats.org/officeDocument/2006/movie, or is not present, then the document shall be considered non-conformant. If an application cannot process external content of the content type specified by the targeted part, then it may be ignored.

[*Example*: Consider a WordprocessingML document which contains a VML shape holding the static image for a movie:

```
<v:shape id="_x0000_s1026" … >
  <w:movie r:id="rIdMovie" />
  <v:imagedata r:id="rId5" r:pict="rId6" o:title="" o:movie="7168" />
</v:shape>
```

The movie element specifies that the part targeted by the relationship with an ID of `rIdMovie` shall be imported at the beginning of the document. Examining the contents of the corresponding relationship part item, we can see the targets for that relationship:

```
<Relationships … >
  …
  <Relationship Id="rIdMovie" TargetMode="Internal"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/movie"
Target="movie.mov" />
  …
</Relationships>
```

The corresponding relationship part item shows that the movie file is located next to the main document and is named `movie.mov`. *end example*]

Parent Elements
pict (§2.3.3.21); pict (§2.9.23)

Attributes	Description
id (Relationship to Part)	Specifies the relationship ID to a specified part.

Attributes	Description
Namespace: .../officeDocument/2006/relationships	The specified relationship shall match the type required by the parent element: • http://schemas.openxmlformats.org/officeDocument/2006/relationships/footer for the footerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/header for the headerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://schemas.openxmlformats.org/officeDocument/2006/relationships/printerSettings for the printerSettings element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Rel">
   <attribute ref="r:id" use="required"/>
</complexType>
```

2.3.3.18 noBreakHyphen (Non Breaking Hyphen Character)

This element specifies that a non breaking hyphen character shall be placed at the current location in the run content. A *non breaking hyphen* is the equivalent of Unicode character 002D (the hyphen-minus), however it shall not be used as a valid line breaking character for the current line of text when displaying this WordprocessingML content.

The behavior of a non breaking hyphen in run content shall be to display using the same glyph as the hyphen-minus character, however without being a valid line breaking position (unlike the hyphen-minus character).

[*Example*: Consider the following sentence in a WordprocessingML document:

`This makes a very very very wordy and deliberately overcomplicated sentence.`

Normally, just as shown above, this sentence not would be displayed on a single line as it is long enough to require line breaking (given the width of the current page). However, if a hyphen minus were inserted after the letter s in `sentence`, as follows:

```
<w:r>
  <w:t>This makes a very very very wordy and deliberately overcomplicated s-
entence.</w:t>
</w:r>
```

This would allow a break at that position, and break the word after that character:

```
This makes a very very very wordy and deliberately overcomplicated s-entence.
```

If this was not desired, the non breaking hyphen character could be specified as follows:

```
<w:r>
  <w:t>This makes a very very very wordy and deliberately overcomplicated
s</w:t>
  <w:nonBreakHyphen/>
  <w:t>entence.</w:t>
</w:r>
```

This would display a hyphen character, but would not allow the text to break at that location:

```
This makes a very very very wordy and deliberately overcomplicated s-entence.
```

end example]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

2.3.3.19 object (Inline Embedded Object)

This element specifies that an embedded object is located at this position in the run's contents. The layout properties of this embedded object are specified using the VML syntax (§6.1).

[*Example*: Consider a run which consists of an embedded object which is in line with the text in that paragraph (i.e. on the line and affects the line height). That run would be specified using the following WordprocessingML:

```
<w:r>
  <w:object>
    …
  </w:object>
</w:r>
```

The object element indicates that an embedded object and its VML positioning data are located at the current position in the run (e.g. an embedded object). *end example*]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

Child Elements	Subclause
Any element from the urn:schemas-microsoft-com:vml namespace	§6.1
Any element from the urn:schemas-microsoft-com:office:office namespace	§6.2
control (Inline Embedded Control)	§2.3.3.3

Attributes	Description

dxaOrig (Original Image Width)	Specifies the original (natural) width of the image representation of the current control within the document. Some vector image formats do not store a native size within their format, and this attribute shall only be used in those cases to store this information, so that the image may be appropriately restored as needed.
	
	If this element is excluded, then the natural size of the image as stored in its format shall be used.
	
	[*Example*: Consider the following WordprocessingML for an embedded object:
	
	`<w:object w:dxaOrig="3360" w:dyaOrig="2520">`
	`  …`
	`  <v:shape … style="width:168pt;height:4in">`
	`  …`
	`  </v:shape>`
	`  <o:OLEObject … />`
	`</w:object>`
	
	The dxaOrig attribute has a value of 3360, which specifies that the image used for the embedded object doesn't store its native width, but that width should be 3360 twentieths of a point. *end example*]
	
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

dyaOrig (Original Image Height)	Specifies the original (natural) height of the image representation of the current control within the document. Some vector image formats do not store a native size within their format, and this attribute shall only be used in those cases to store this information, so that the image may be appropriately restored as needed. If this element is excluded, then the natural size of the image as stored in its format shall be used. [*Example*: Consider the following WordprocessingML for an embedded object:

Attributes	Description
	<w:object w:dxaOrig="3360" w:dyaOrig="2520"> … <v:shape … style="width:168pt;height:4in"> … </v:shape> <o:OLEObject … /> </w:object> The dyaOrig attribute has a value of 2520, which specifies that the image used for the embedded object doesn't store its native height, but that height should be 2520 twentieths of a point. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Object">
   <complexContent>
      <extension base="CT_PictureBase">
         <sequence>
            <element name="control" type="CT_Control" minOccurs="0"/>
         </sequence>
         <attribute name="dxaOrig" type="ST_TwipsMeasure" use="optional"/>
         <attribute name="dyaOrig" type="ST_TwipsMeasure" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

2.3.3.20 pgNum (Page Number Block)

This element specifies the presence of a page number block at the current location in the run content. A *page number block* is a non-editable region of text which shall display the current page using ascending decimal numbers. [*Note*: The page number block is a legacy construct used for compatibility with older word processors, and should not be produced unless it was consumed while reading a document – it is recommended that the PAGENUM field is used in its place. *end note*]

A page number block shall be displayed using ascending decimal numbers, regardless of the languages specified in the parent run's lang property (§2.3.2.18).

[*Example*: Consider a WordprocessingML run with the following run content:

```
<w:r>
  <w:t xml:space="preserve">This is the current page: </w:t>
  <w:pgNum />
</w:r>
```

This run specifies that a page number block shall be placed after the text string literal `This is the curent page:` in the document. Assuming that this content is on the first page, this run would be displayed as follows:

```
This is the current page: 1
```

end example]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

2.3.3.21 pict (VML Object)

This element specifies that an object is located at this position in the run's contents. The layout properties of this object are specified using the VML syntax (§6.1).

[*Example*: Consider a run which consists of an object specified using VML. That run would be specified using the following WordprocessingML:

```
<w:r>
  <w:pict>
    …
  </w:pict>
</w:r>
```

The pict element indicates that an object specified in VML is located at the current position in the run (e.g. a floating embedded control). *end example*]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

Child Elements	Subclause
Any element from the urn:schemas-microsoft-com:vml namespace	§6.1
Any element from the urn:schemas-microsoft-com:office:office namespace	§6.2
control (Floating Embedded Control)	§2.3.3.2
movie (Embedded Video)	§2.3.3.17

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Picture">
   <complexContent>
      <extension base="CT_PictureBase">
         <sequence maxOccurs="1">
            <element name="movie" type="CT_Rel" minOccurs="0"/>
            <element name="control" type="CT_Control" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.3.3.22 ptab (Absolute Position Tab Character)

This element specifies that an absolute position tab character shall be placed at the current location in the run content. An *absolute position tab* is a character which is used to advance the position on the current line of text when displaying this WordprocessingML content, using the following logic:

Regardless of any number of custom tab stops defined using the tabs element (§2.3.1.38) , the absolute position tab character shall advance to the position specified by its alignment and relativeTo attributes. The resulting end position of the tab character shall not be affected by the addition of any custom tab stops or changes to the value of the defaultTabStop element (§2.15.1.24).

If the alignment location specified by the positional tab cannot be found on the current line, because the starting location is past that point, then the tab character shall advance to that location on the next available line in the document.

[*Example*: Consider a paragraph which contains two custom tab stops at 1.5" and 3.5", respectively. These two tab stops would be contained within a tabs element defining the set of tab stops of the paragraph as follows:

```
<w:pPr>
  <w:tabs>
    <w:tab w:val="left" w:pos="2160" />
    <w:tab w:val="left" w:pos="5040" />
  </w:tabs>
</w:pPr>
```

If a positional tab character was added to a run in this paragraph starting at 1” inside the margin and was defined as follows:

```
<w:ptab w:alignment="center" w:relativeTo="margin" />
```

This positional tab would then ignore the next custom tab stop and the indents on the current paragraph defined using the ind element (§2.3.1.12) and would advance to the center of the line with respect to the text margins, moving to a new line if needed. *end example*]

Parent Elements

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

Attributes	Description
alignment (Positional Tab Stop Alignment)	Specifies the location of the positional tab stop on the line, as well as the alignment which shall be applied to text subsequent to the current positional tab stop. [*Example*: Consider a positional tab stop in a WordprocessingML document who shall move to the left edge of the text margins and whose subsequent text should be left aligned. This positional tab stop would be defined as follows: `<w:ptab w:alignment="left" w:relativeTo=”margin” … />` The alignment attribute specifies that this absolute position tab stop shall align on the left edge of the line relative to the margin. *end example*] The possible values for this attribute are defined by the ST_PTabAlignment simple type (§2.18.78).
leader (Tab Leader Character)	Specifies the character which shall be used to fill in the space created by a positional tab. This character shall be repeated as required to completely fill the tab spacing generated by the positional tab character. [*Example*: Consider a positional tab stop which should be preceded by a sequence of underscore characters, as follows: _____Text at the positional tab stop This tab stop would have a leader attribute value of `underscore`, indicating that the tab stop shall be preceded by underscore characters as needed to fill the tab spacing. *end example*] The possible values for this attribute are defined by the ST_PTabLeader simple type (§2.18.79).
relativeTo (Positional Tab Base)	Specifies the extents which shall be used to calculate the absolute positioning of this positional tab character. [*Example*: Consider a positional tab stop in a WordprocessingML document that should have a resulting position that is centered on the text margins, ignoring both any custom tab stops and any text indents on the paragraph. This positional tab stop would be defined as follows: `<w:ptab w:relativeTo="margin" … />` The relativeTo attribute specifies that this absolute position tab stop shall be relative to the margin. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_PTabRelativeTo simple type (§2.18.80).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PTab">
   <attribute name="alignment" type="ST_PTabAlignment" use="required"/>
   <attribute name="relativeTo" type="ST_PTabRelativeTo" use="required"/>
   <attribute name="leader" type="ST_PTabLeader" use="required"/>
</complexType>
```

2.3.3.23 rt (Phonetic Guide Text)

This element specifies the presence of the guide text within a phonetic guide at the current location in the document.

The contents of the guide text run are specified in the child r element (§2.3.2.23).

[*Example*: Consider the following two runs, each containing a phonetic guide:

tōkyō
東京

The guide text run would be specified using the following WordprocessingML:

```
<w:rt>
  …
  <w:r>
    <w:t>tō</w:t>
  </w:r>
</w:rt>
```

The guide text is contained in a run within the rt element. *end example*]

Parent Elements
ruby (§2.3.3.24)

Child Elements	Subclause
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4

Child Elements	Subclause
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
del (Deleted Run Content)	§2.13.5.12
ins (Inserted Run Content)	§2.13.5.20
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
r (Text Run)	§2.3.2.23

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RubyContent">
    <group ref="EG_RubyContent" minOccurs="0" maxOccurs="unbounded"/>
</complexType>
```

2.3.3.24 ruby (Phonetic Guide)

This element specifies the presence of a phonetic guide at the current location in the document. A *phonetic guide* (often called ruby text) is a run of content with base text which appears at the normal baseline location for text in this run, with phonetic guide text displayed above it in the document. The resulting construct is called a phonetic guide as it is typically used to map words in one language to another phonetically.

The base text is stored in the rubyBase element (§2.3.3.26) and the guide text is stored in the rt element (§2.3.3.23).

[*Example*: Consider the following two runs, each containing a phonetic guide:

tōkyō
東京

The first run would be specified using the following WordprocessingML:

```
<w:r>
  <w:ruby>
    <w:rubyPr>
       …
    </w:rubyPr>
    <w:rt>
      …
      <w:r>
        <w:t>tō</w:t>
      </w:r>
    </w:rt>
    <w:rubyBase>
      …
      <w:r>
        <w:t>東/w:t>
      </w:r>
    </w:rubyBase>
  </w:ruby>
</w:r>
```

The base text is contained in a run within the rubyBase element, and the guide text is contained in a run within the rt element. end example]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

Child Elements	Subclause
rt (Phonetic Guide Text)	§2.3.3.23
rubyBase (Phonetic Guide Base Text)	§2.3.3.26
rubyPr (Phonetic Guide Properties)	§2.3.3.27

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Ruby">
   <sequence>
      <element name="rubyPr" type="CT_RubyPr"/>
      <element name="rt" type="CT_RubyContent"/>
      <element name="rubyBase" type="CT_RubyContent"/>
   </sequence>
</complexType>
```

2.3.3.25 rubyAlign (Phonetic Guide Text Alignment)

This element specifies the alignment setting which shall be used to determine the placement of phonetic guide text with respect to the base text when this phonetic guide is displayed.

[*Example*: Consider a run of phonetic guide text which shall have the ruby text positioned to the far left of the base text. This constraint is specified using the following WordprocessingML:

```
<w:rubyPr>
  …
  <w:rubyAlign w:val="left"/>
  …
</w:rubyPr>
```

The rubyAlign property is `left` for the phonetic guide, so the ruby text will be displayed on the left side of the base text. *end example*]

Parent Elements
rubyPr (§2.3.3.27)

Attributes	Description

val (Phonetic Guide Text Alignment Value)	Specifies the type of alignment to be applied to the phonetic guide text.
	
	[*Example*: Consider a run of phonetic guide text which shall have the ruby text positioned to the far right of the base text. This constraint is specified using the following WordprocessingML:
	
	`<w:rubyPr>`
	`  …`
	`  <w:rubyAlign w:val="left"/>`
	`  …`
	`</w:rubyPr>`
	
	The value of the val attribute is `right` for the phonetic guide, so the ruby text will be displayed on the right side of the base text. *end example*]
	
	The possible values for this attribute are defined by the ST_RubyAlign simple type (§2.18.82).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RubyAlign">
   <attribute name="val" type="ST_RubyAlign" use="required"/>
</complexType>
```

2.3.3.26 rubyBase (Phonetic Guide Base Text)

This element specifies the presence of the base text within a phonetic guide at the current location in the document.

The contents of the base text run are specified in the child r element (§2.3.2.23).

[*Example*: Consider the following two runs, each containing a phonetic guide:

tōkyō
東京

The base text run would be specified using the following WordprocessingML:

```
<w:rubyBase>
  …
  <w:r>
    <w:t>東</w:t>
  </w:r>
</w:rubyBase>
```

The base text is contained in a run within the rubyBase element. *end example*]

Parent Elements
ruby (§2.3.3.24)

Child Elements	Subclause
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
del (Deleted Run Content)	§2.13.5.12
ins (Inserted Run Content)	§2.13.5.20

Child Elements	Subclause
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
r (Text Run)	§2.3.2.23

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RubyContent">
   <group ref="EG_RubyContent" minOccurs="0" maxOccurs="unbounded"/>
</complexType>
```

2.3.3.27 rubyPr (Phonetic Guide Properties)

This element specifies a set of properties which determine the behavior and appearance of a phonetic guide within the document.

[*Example*: Consider the following two runs, each containing a phonetic guide:

tōkyō
東京

The properties for both of these phonetic guides are as follows:

```
<w:r>
  <w:ruby>
    <w:rubyPr>
      <w:rubyAlign w:val="distributeSpace" />
      <w:hps w:val="16" />
      <w:hpsRaise w:val="20" />
      <w:hpsBaseText w:val="22" />
      <w:lid w:val="ja-JP" />
    </w:rubyPr>
    …
  </w:ruby>
</w:r>
```

The phonetic guide properties specify that the guide text shall be:

- Distributed across the top (using the rubyAlign element)
- 8 point font face (using the hps element)
- 10 points above the base text (using the hpaRaise element)
- Japanese (using the lid element)

As well, the phonetic guide properties specify that the base text shall be:

- 11 point font face (using the hpsBaseText element)

end example]

Parent Elements
ruby (§2.3.3.24)

Child Elements	Subclause
dirty (Invalidated Field Cache)	§2.3.3.8
hps (Phonetic Guide Text Font Size)	§2.3.3.10
hpsBaseText (Phonetic Guide Base Text Font Size)	§2.3.3.11
hpsRaise (Distance Between Phonetic Guide Text and Phonetic Guide Base Text)	§2.3.3.12
lid (Language ID for Phonetic Guide)	§2.3.3.14
rubyAlign (Phonetic Guide Text Alignment)	§2.3.3.25

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RubyPr">
   <sequence>
      <element name="rubyAlign" type="CT_RubyAlign"/>
      <element name="hps" type="CT_HpsMeasure"/>
      <element name="hpsRaise" type="CT_HpsMeasure"/>
      <element name="hpsBaseText" type="CT_HpsMeasure"/>
      <element name="lid" type="CT_Lang"/>
      <element name="dirty" type="CT_OnOff" minOccurs="0"/>
   </sequence>
</complexType>
```

2.3.3.28 softHyphen (Optional Hyphen Character)

This element specifies that an optional hyphen character shall be placed at the current location in the run content. An *optional hyphen* is a character which may be used as a valid line breaking character for the current line of text when displaying this WordprocessingML content, using the following logic:

- When this character is not the character which is used to break the line, then it shall not change the normal display of text (it shall have zero width)

- When this character is the character used to break the line, it shall display using the hyphen-minus character within the display of text

[*Note*: This character is typically used to mark locations where a word may optionally be hyphenated without causing the hyphen character to be displayed unnecessarily. *end note*]

[*Example*: Consider the following sentence in a WordprocessingML document:

```
This sentence needs to be long enough to cause some kind of line breaking.
```

Normally, just as shown above, this sentence not would be displayed on a single line as it is long enough to require line breaking (given the width of the current page). However, if an optional hyphen were inserted after the letter r in `breaking`, as follows:

```
<w:r>
  <w:t>This sentence needs to be long enough to cause some kind of line br</w:t>
  <w:softHyphen/>
  <w:t>eaking.</w:t>
</w:r>
```

This would allow a break at that position, and when that location is the point of the line break, would insert a hyphen-minus in the word after that character:

```
This sentence needs to be long enough to cause some kind of line breaking.
```

If this was not the point of the line break, then no character would be displayed at that location:

```
This sentence should not be long enough to cause line breaking.
```

The sentence now does not break at that location, so no hyphen appears in the word `breaking`. *end example*]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

2.3.3.29 sym (Symbol Character)

This element specifies the presence of a symbol character at the current location in the run's content. A *symbol character* is a special character within a run's content which does not use any of the run fonts specified in the rFonts element (§2.3.2.24) (or by the style hierarchy).

Instead, this character shall be determined by pulling the character with the hexadecimal value specified in the char attribute from the font specified in the font attribute.

[*Example*: Consider a run containing the following run content:

```
This is a symbol character:💻
```

The last character in that run is a symbol character from the Wingdings font, and the run is specified as follows:

```
<w:r>
  <w:rPr>
    <w:rFonts w:ascii="Courier New" w:hAnsi="Courier New" />
  </w:rPr>
  <w:t>This is a symbol character:</w:t>
  <w:sym w:font="Wingdings" w:code="F03A" />
</w:r>
```

The resulting symbol is the specified using the sym element, and consists of character code `003A` formatted as `Wingdings`, even though the run properties specify the `Courier New` font. *end example*]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

Attributes	Description

char (Symbol Character Code)	Specifies the hexadecimal code for the Unicode character value of the symbol.
	
	When this value is stored in the char attribute, it may be stored in either of the following two formats:
	1. Directly in its Unicode character value from the font glyph
	2. In a Unicode character value created by adding F000 to the actual character value, shifting the character value of this character into the Unicode private use area.
	
	[*Note*: The use of the latter syntax allows for interoperability with legacy word processing formats, as they used this technique to store the fact that a particular character or set of characters came from a font which was not Unicode compliant, and therefore any font matching performed on this range (if the specified font was not present) would be undesirable, as the resulting glyphs and their appearance could not be predicted. *end note*]
	
	[*Example*: Consider a run with a single symbol character defined as follows:
	
	`<w:r>`
	`  <w:rPr>`
	`  <w:rFonts w:ascii="Arial Black" w:hAnsi="Arial Black" />`
	`  </w:rPr>`
	`  <w:sym w:font="Wingdings" w:char="F045" />`
	`</w:r>`
	
	The symbol character shall use the font defined in its font attribute and hence use the

Attributes	Description
	Wingdings font. The character value for the character to be used from this font is obtained by removing the F000 value from the value in the char attribute, and therefore is the character at hexadecimal position 0045 in that font. *end example*] The possible values for this attribute are defined by the ST_ShortHexNumber simple type (§2.18.86).

font (Symbol Character Font)	Specifies a font which shall be used to format this symbol character.
	
	[*Example*: Consider a run with a single symbol character defined as follows:
	
	`<w:r>`
	`  <w:rPr>`
	`  <w:rFonts w:ascii="Arial Black" w:hAnsi="Arial Black" />`
	`  </w:rPr>`
	`  <w:sym w:font="Wingdings" w:char="F045" />`
	`</w:r>`
	
	Although the run specifies that its contents shall use the Arial Black font, the symbol character shall use the font defined in its font attribute and hence use the Wingdings font. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Sym">
   <attribute name="font" type="ST_String"/>
   <attribute name="char" type="ST_ShortHexNumber"/>
</complexType>
```

2.3.3.30 t (Text)

This element specifies that this run contains literal text which shall be displayed in the document. The t element shall be used for all text runs which is not:

- Part of a region of text that is contained in a deleted region using the del element (§2.13.5.12)
- Part of a region of text that is contained within a field code

[*Example*: Consider a paragraph of WordprocessingML content which reads This is text. This paragraph would therefore be represented as follows:

```
<w:p>
  <w:r>
    <w:t>This is text</w:t>
  </w:r>
</w:p>
```

The text is contained in a t node. *end example*]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

Attributes	Description

space (Content Contains Significant Whitespace)
 	
Namespace: http://www.w3.org/XML/1998/namespace
	Specifies how white space should be handled for the contents of this element using the W3C space preservation rules.
	
	[*Example*: Consider the following run contained within a WordprocessingML document:
	
	`<w:r>`
	`  <w:t>significant whitespace  </w:t>`
	`</w:r>`
	
	Although there are three spaces on each side of the text content in the run, that whitespace has not been specifically marked as significant, therefore it is removed when this run is added to the document. *end example*]
	
	The possible values for this attribute are defined by the type in the namespace.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Text">
   <simpleContent>
      <extension base="ST_String">
         <attribute ref="xml:space" use="optional"/>
      </extension>
   </simpleContent>
</complexType>
```

2.3.3.31 tab (Tab Character)

This element specifies that a tab character shall be placed at the current location in the run content. An *tab* is a character which is used to advance the position on the current line of text when displaying this WordprocessingML content, using the following logic:

- When there are one or more custom tab stops defined using the tabs element (§2.3.1.38) , then the tab character shall advance to the next custom tab stop location which is further along than the starting location of the tab
- When there are no custom tab stops which are further than the current position in the line, the tab character shall advance to the nearest multiple of the defaultTabStop element (§2.15.1.24) width value.

[*Example*: Consider a paragraph which contains two custom tab stops at 1.5" and 3.5", respectively. These two tab stops would be contained within a tabs element defining the set of tab stops of the paragraph as follows:

```
<w:pPr>
```

```
    <w:tabs>
      <w:tab w:val="left" w:pos="2160" />
      <w:tab w:val="left" w:pos="5040" />
    </w:tabs>
  </w:pPr>
```

If a tab character was added to a run in this paragraph and appeared 1.4” along the line after all preceding content was laid out, then this tab would move the position to 1.5”. If the tab character appeared 1.6” along the line after all preceding content was laid out, then this tab would move the position to 3.5”. In both cases, the tab advanced to the next custom tab stop. *end example*]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

2.3.3.32 yearLong (Date Block - Long Year Format)

This element specifies the presence of a date block at the current location in the run content. A *date block* is a non-editable region of text which shall display the current date filtered through the specified date picture (see following paragraphs). [*Note*: The date block is a legacy construct used for compatibility with older word processors, and should not be produced unless it was consumed while reading a document – it is recommended that the DATE field is used in its place. *end note*]

A date block shall be displayed using the primary editing language of the host application, regardless of the languages specified in the parent run’s lang property (§2.3.2.18).

The long year format date block shall use a date picture of YYYY, retrieving the long year format for the primary editing language.

[*Example*: Consider a WordprocessingML run with the following run content:

```
<w:r>
  <w:t xml:space="preserve">This is a long date: </w:t>
  <w:yearLong />
</w:r>
```

This run specifies that a long year format date block shall be placed after the text string literal `This is a long date:` in the document. Assuming that the host application’s primary editing language is English (Canada) and today’s date is 04/12/2006 (a Wednesday), this run would be displayed as follows:

```
This is a long date: 2006
```

end example]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

2.3.3.33 yearShort (Date Block - Short Year Format)

This element specifies the presence of a date block at the current location in the run content. A *date block* is a non-editable region of text which shall display the current date filtered through the specified date picture (see following paragraphs). [*Note*: The date block is a legacy construct used for compatibility with older word processors, and should not be produced unless it was consumed while reading a document – it is recommended that the DATE field is used in its place. *end note*]

A date block shall be displayed using the primary editing language of the host application, regardless of the languages specified in the parent run's lang property (§2.3.2.18).

The short year format date block shall use a date picture of YY, retrieving the short year format for the primary editing language.

[*Example*: Consider a WordprocessingML run with the following run content:

```
<w:r>
  <w:t xml:space="preserve">This is a short date: </w:t>
  <w:yearShort />
</w:r>
```

This run specifies that a short year format date block shall be placed after the text string literal `This is a short date:` in the document. Assuming that the host application's primary editing language is French (Canada) and today's date is 04/12/2006 (a Wednesday), this run would be displayed as follows:

```
This is a short date: 06
```

end example]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

2.4 Tables

Another type of block-level content in WordprocessingML, a *table* is a set of paragraphs (and other block-level content) arranged in *rows* and *columns*. Tables in WordprocessingML are defined via the tbl element, which is

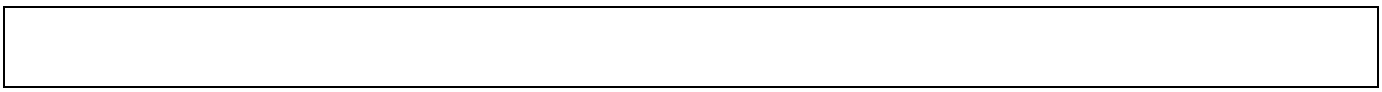
analogous to the HTML <table> tag. The table element specifies the location of a table present in the document.

A tbl element (§2.4.36) has two elements that define its properties:

- tblPr (§2.4.55), which defines the set of table-wide properties (such as style and width)
- tblGrid (§2.4.44), which defines the grid layout of the table.

A tbl element can also contain an arbitrary non-zero number of rows, where each row is specified with a tr element (§2.4.75). Each tr element can contain an arbitrary non-zero number of cells, where each cell is specified with a tc element (§2.4.62).

[*Example*: Consider an empty one-cell table (i.e.; a table with one row, one column) and 1 point borders on all sides as follows:



This table is represented by the following WordprocessingML:

```
<w:tbl>
  <w:tblPr>
    <w:tblW w:w="5000" w:type="pct"/>
    <w:tblBorders>
      <w:top w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:left w:val="single" w:sz="4 w:space="0" w:color="auto"/>
      <w:bottom w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:right w:val="single" w:sz="4" w:space="0" w:color="auto"/>
    </w:tblBorders>
  </w:tblPr>
  <w:tblGrid>
    <w:gridCol w:w="10296"/>
  </w:tblGrid>
  <w:tr>
    <w:tc>
      <w:tcPr>
        <w:tcW w:w="0" w:type="auto"/>
      </w:tcPr>
      <w:p/>
    </w:tc>
  </w:tr>
</w:tbl>
```

This table specifies table-wide properties of 100% of page width using the tblW element and the set of table borders using the tblBorders element, the table grid which defines a set of shared vertical edges within the table using the tblGrid element, and a single row using the tr element. *end example*]

2.4.1 bidiVisual (Visually Right to Left Table)

This element specifies that the cells with this table shall be visually represented in a right to left direction. This element also affects the application of all table-level properties.

When this property is specified, then the ordering of all cells (and table-level properties) in this table shall be applied to the table assuming that the table is a normal left to right table, but the table cells shall be displayed in a right to left direction. [*Example*: A left border on the first table cell shall be displayed on the right side of that cell (which would be the rightmost cell) in a visually right to left table. *end example*]

If this element is omitted, then the table shall not be presented right to left.

[*Example*: Consider the following table which has the logical right to left property set:

	Three	Two	One

This property would be specified in the WordprocessingML as follows:

```
<w:tblPr>
  <w:bidiVisual/>
</w:tblPr>
```

Since the bidiVisual element specifies this is a visually right to left table, the actual table data would be stored in its logical order as follows:

```
<w:tr>
  <w:tc>
    <w:p>
      <w:r>
        <w:t>One</w:t>
      </w:r>
    </w:p>
  </w:tc>
```

```
    <w:tc>
      <w:p>
        <w:r>
          <w:t>Two</w:t>
        </w:r>
      </w:p>
    </w:tc>
    <w:tc>
      <w:p>
        <w:r>
          <w:t>Three</w:t>
        </w:r>
      </w:p>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
  </w:tr>
```

The first logical cell with text `One` is stored first in the file format, and displayed on the rightmost in this table visually. *end example*]

Parent Elements
tblPr (§2.7.5.3); tblPr (§2.7.5.4); tblPr (§2.4.55); tblPr (§2.4.56)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.4.2 bottom (Table Cell Bottom Margin Exception)

This element specifies the amount of space which shall be left between the bottom extent of the cell contents and the border of a specific table cell within a table. This setting shall override the table cell bottom margin definition specified by the bottom element contained within the table properties (§2.4.5).

This value is specified in the units applied via its type attribute. Any width value of type pct or auto for this element shall be ignored.

If omitted, then this table cell shall use the bottom cell margins defined in the bottom element contained within the table properties (§2.4.5).

[*Example*: Consider a table with two cells in which the first table cell's bottom margin is specified via an exception to be ten times larger (0.2 inches vs. 0.02 inches) than the other table cell margins:

This text fills the extents of the cell.
So does this

The first cell in the table would be specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    <w:tcMar>
      <w:bottom w:w="288" w:type="dxa" />
    </w:tcMar>
  </w:tcPr>
</w:tc>
```

The first cell in this table has an exception applied to the table cell bottom cell margin setting it to 288 twentieths of a point (0.2 inches). *end example*]

Parent Elements
tcMar (§2.4.65)

Attributes	Description
type (Table Width Type)	Specifies the units of the width property being defined by the parent element's w attribute. This property is used to define various properties of a table, including: cell spacing, preferred width, and table margins.

Attributes	Description
	If this attribute is omitted, then its value shall be assumed to be dxa (twentieths of a point). [*Example*: Consider a table with a table cell bottom cell spacing with a type of dxa, as follows: `<w:bottom ... w:type="dxa" />` This type shall therefore be used to interpret the width specified in the w attribute as a value in twentieths of a point. *end example*] The possible values for this attribute are defined by the ST_TblWidth simple type (§2.18.97).
w (Table Width Value)	Specifies the value of the width property being defined by the parent element. This property is used to define various properties of a table, including: cell spacing, preferred widths, and table margins. If this attribute is omitted, then its value shall be assumed to be 0. [*Example*: Consider a table with a bottom margin with a width of 302, as follows: `<w:bottom w:w="302" w:type="dxa" />` The value in the w attribute shall therefore be used to determine the width being specified in the context of the units specified in the type attribute. In this case, the type is twentieths of a point (dxa), so the width is 302 twentieths of a point (.2097 inches). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblWidth">
   <attribute name="w" type="ST_DecimalNumber"/>
   <attribute name="type" type="ST_TblWidth"/>
</complexType>
```

2.4.3 bottom (Table Cell Bottom Border)

This element specifies the border which shall be displayed at the bottom of the current table cell. The appearance of this table cell border in the document shall be determined by the following settings:

- If the net tblCellSpacing element value (§2.4.41;§2.4.42;§2.4.43) applied to the cell is non-zero, then the cell border shall always be displayed

- Otherwise, the display of the border is subject to the conflict resolution algorithm defined by the tcBorders element (§2.4.63) and the tblBorders element (§2.4.37;§2.4.38)

If this element is omitted, then the bottom of this table cell shall not have a cell border, and its border may use the table's border settings as appropriate.

[*Example*: Consider a table in which the first cell in the first row specifies a bottom cell border

R1C1	R1C2
R2C1	R2C2

This bottom cell border is specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    …
    <w:tcBorders>
      <w:bottom w:val="thinThickThinSmallGap" w:sz="24" w:space="0"
w:color="FF0000"/>
    </w:tcBorders>
  </w:tcPr>
  <w:p/>
</w:tc>
```

The bottom element specifies a three point border of type `thinThinThickSmallGap`. *end example*]

Parent Elements
tcBorders (§2.4.63)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or `auto` to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value `auto`, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*]

Attributes	Description
	If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is `true`, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object. When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points. When a document has a page border that is relative to the text extents (using a value of

Attributes	Description

	text in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.
	
	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24/>`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*]
	
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).

sz (Border Width)	Specifies the width of the current border.
	
	If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value.
	
	If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value.
	
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`
	
	The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	
	The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).

themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border. The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the

Attributes	Description

	document.
	
	[*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup:
	
	`<w:top … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />`
	`<w:bottom … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />`
	`<w:left … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />`
	`<w:right … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />`
	
	The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*]
	
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).

themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255$$ $$= 102$$ $$= 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB

Attributes	Description
	[*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `C0504D`. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $L' = 0.53 * 0.75$ $= 0.39698$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$ and converting back to RGB, we get `943634`. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $T_{xml} = 0.6 * 255$ $= 153$ $= 99(hex)$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows:

Attributes	Description
	$$L^{'} = L * Tint_{pct} + (1 - Tint_{pct})$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `4F81BD`. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L^{'} = 0.53 * 0.6 + (1 - .6)$$ $$= 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get `95B3D7`. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <attribute name="val" type="ST_Border" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
   <attribute name="space" type="ST_PointMeasure" use="optional"/>
   <attribute name="shadow" type="ST_OnOff" use="optional"/>
   <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.4.4 bottom (Table Bottom Border)

This element specifies the border which shall be displayed at the bottom of the current table. The appearance of this table border in the document shall be determined by the following settings:

- The display of the border is subject to the conflict resolution algorithm defined by the tcBorders element (§2.4.63) and the tblBorders element (§2.4.37;§2.4.38)

If this element is omitted, then the bottom of this table shall have the border specified by the associated table style. If no bottom border is specified in the style hierarchy, then this table shall not have a bottom border.

[*Example*: Consider a table in which the table properties specifies a bottom table border, as follows:

R1C1	R1C2
R2C1	R2C2

This bottom table border is specified using the following WordprocessingML:

```
<w:tbl>
  <w:tblPr>
    <w:tblBorders>
      <w:bottom w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
w:color="D0D0D0" w:themeColor="accent3" w:themeTint="99"/>
    </w:tblBorders>
  </w:tblPr>
  …
</w:tbl>
```

The bottom element specifies a three point bottom table border of type thinThinThickMediumGap. *end example*]

Parent Elements

Parent Elements
tblBorders (§2.4.37); tblBorders (§2.4.38)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or `auto` to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value `auto`, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is `true`, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect.

Attributes	Description
	[*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object.
	
	When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points.
	
	When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.
	
	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24/>`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*]
	
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).

sz (Border Width)	Specifies the width of the current border. If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value. If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value.

Attributes	Description

	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`
	
	The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	
	The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).

themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border.
	
	The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document.
	
	[*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup:
	
	`<w:top … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:bottom … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:left … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:right … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	
	The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*]
	
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).

themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border.

Attributes	Description
	[*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255 \\ = 102 \\ = 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}^{'} = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L^{'} = 0.53 * 0.75 \\ = 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from

Attributes	Description
	the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255 = 153 = 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6) = 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).

Attributes	Description
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <attribute name="val" type="ST_Border" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
   <attribute name="space" type="ST_PointMeasure" use="optional"/>
   <attribute name="shadow" type="ST_OnOff" use="optional"/>
   <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.4.5 bottom (Table Cell Bottom Margin Default)

This element specifies the amount of space which shall be left between the bottom extent of the cell contents and the border of all table cells within the parent table (or table row). This setting may be overridden by the table cell bottom margin definition specified by the bottom element contained within the table cell's properties (§2.4.2).

This value is specified in the units applied via its type attribute. Any width value of type `pct` or `auto` for this element shall be ignored.

If this element is omitted, then it shall inherit the table cell margin from the associated table style. If a bottom margin is never specified in the style hierarchy, then this table shall have no bottom cell padding by default (excepting individual cell overrides).

[*Example*: Consider a two by two table in which the default table cell bottom margin is specified to be exactly 0.25 inches, as follows (marked with an arrow in the first table cell below):

R1C1 ↕	R2C1

R2C1	R2C2

This table property is specified using the following WordprocessingML markup:

```
<w:tbl>
  <w:tblPr>
    <w:tblCellMar>
      <w:bottom w:w="360" w:type="dxa"/>
    </w:tblCellMar>
  </w:tblPr>
  …
</w:tbl>
```

Every cell in the table has a default cell margin setting it to 360 twentieths of a point. *end example*]

Parent Elements
tblCellMar (§2.4.39); tblCellMar (§2.4.40)

Attributes	Description
type (Table Width Type)	Specifies the units of the width property being defined by the parent element's w attribute. This property is used to define various properties of a table, including: cell spacing, preferred width, and table margins. If this attribute is omitted, then its value shall be assumed to be dxa (twentieths of a point). [*Example*: Consider a table with a table cell bottom cell spacing with a type of dxa, as follows: `<w:bottom ... w:type="dxa" />` This type shall therefore be used to interpret the width specified in the w attribute as a value in twentieths of a point. *end example*] The possible values for this attribute are defined by the ST_TblWidth simple type (§2.18.97).
w (Table Width Value)	Specifies the value of the width property being defined by the parent element. This property is used to define various properties of a table, including: cell spacing, preferred widths, and table margins. If this attribute is omitted, then its value shall be assumed to be 0. [*Example*: Consider a table with a bottom margin with a width of 302, as follows:

Attributes	Description
	<w:bottom w:w="302" w:type="dxa" /> The value in the w attribute shall therefore be used to determine the width being specified in the context of the units specified in the type attribute. In this case, the type is twentieths of a point (dxa), so the width is 302 twentieths of a point (.2097 inches). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

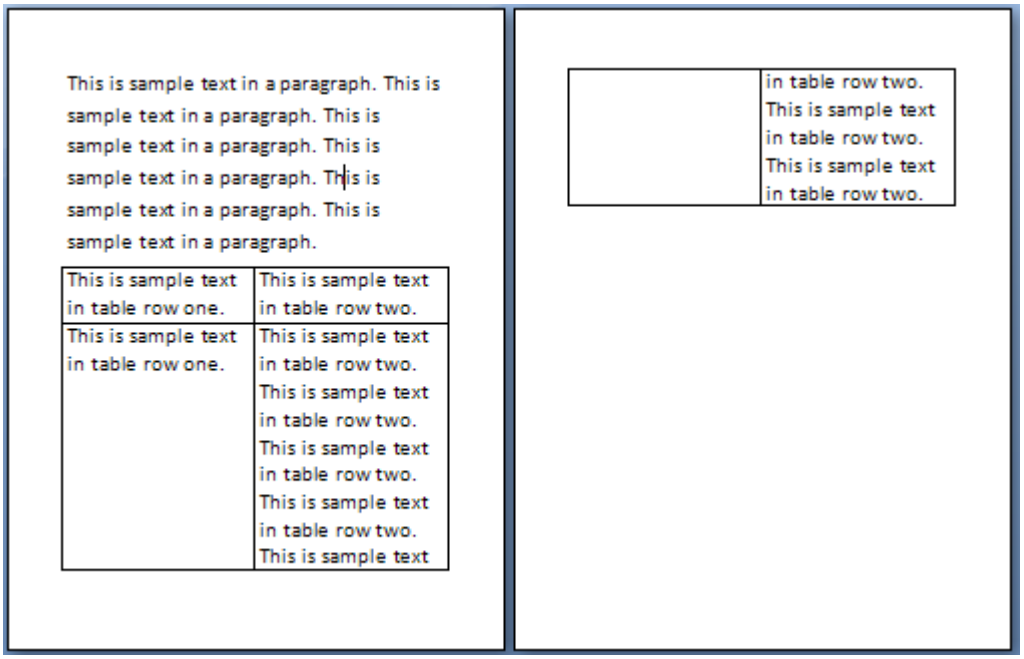
```
<complexType name="CT_TblWidth">
   <attribute name="w" type="ST_DecimalNumber"/>
   <attribute name="type" type="ST_TblWidth"/>
</complexType>
```

2.4.6 cantSplit (Table Row Cannot Break Across Pages)

This element specifies whether the contents within the current cell shall be rendered on a single page. When displaying the contents of a table cell (such as the table cells in this specification), it is possible that a page break would fall within the contents of a table cell, causing the contents of that cell to be displayed across two different pages. If this property is set, then all contents of a table row shall be rendered on the same page by moving the start of the current row to the start of a new page if necessary. If the contents of this table row cannot fit on a single page, then this row shall start on a new page and flow onto multiple pages as necessary.

If this element is not present, the default behavior is dictated by the setting in the associated table style. If this property is not specified in the style hierarchy, then this table row shall be allowed to split across multiple pages.

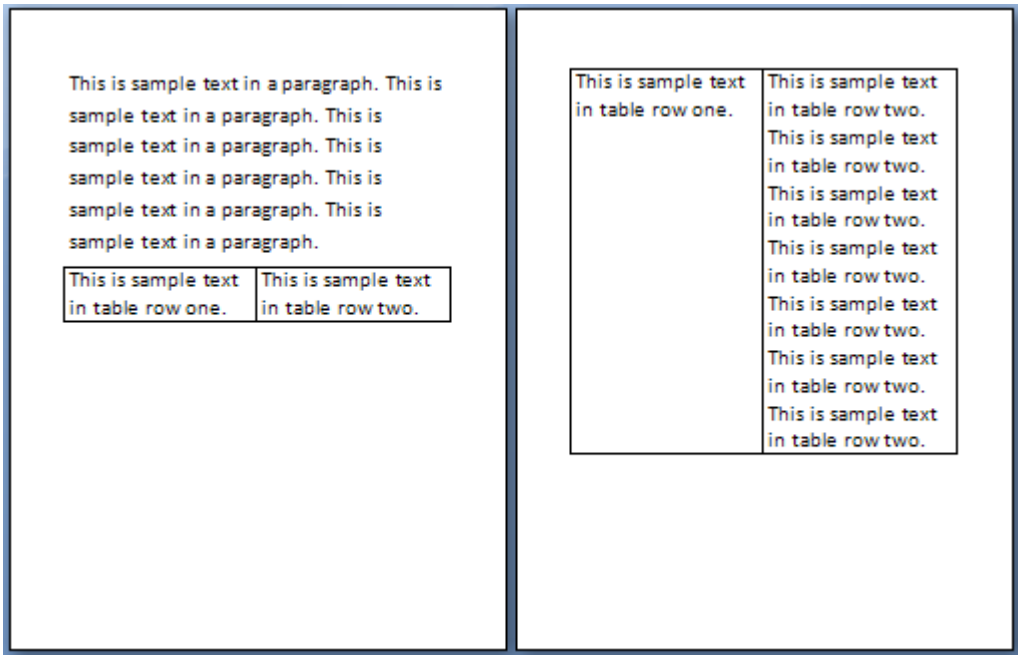
[*Example*: Consider the following content displayed on two different pages of a WordprocessingML document:



When this content is displayed, the contents of the 2nd table row in this document are displayed across two different pages. If the contents of this row are to be displayed on one page, then this requirement would be specified as follows:

```
<w:tr>
  <w:trPr>
    …
    <w:cantSplit />
  </w:trPr>
  …
</w:tr>
```

The presence of the cantSplit element specifies that the table row shall not be broken across multiple pages, therefore the second table row starts on a new page:



This setting therefore ensures that the content will always be displayed on a single page (if it fits on one page). *end example*]

Parent Elements
trPr (§2.7.5.10); trPr (§2.7.5.11); trPr (§2.4.78); trPr (§2.4.79)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.4.7 cnfStyle (Table Cell Conditional Formatting)

This element specifies the set of conditional table style formatting properties which have been applied to this table cell. [*Note*: This property is an optimization which is used by consumers to determine if a given property on a table cell is the result of the table style conditional formatting properties vs. direct formatting on the table cell itself. It specifies the components of the conditional formatting in the table style applied to this cell, so that the table's conditional formatting can be applied after the document is displayed without having the table style properties override the style hierarchy. *end note*]

If this element is omitted, then its value shall be assumed to be zero for all entries in the bit mask.

[*Example*: Consider a table cell in the top right corner of a table with a table style applied. This table cell would need to specify the following WordprocessingML to express that fact:

```
<w:tc>
  <w:tcPr>
    <w:cnfStyle w:val="101000000100" />
    …
  </w:tcPr>
  …
</w:tc>
```

This table cell specifies that it has the conditional properties from the table style for the first column, first row, and the top left corner of the parent table by setting the appropriate bits in the val attribute. *end example*]

Parent Elements
tcPr (§2.7.5.8); tcPr (§2.4.66); tcPr (§2.7.5.9); tcPr (§2.4.67)

Attributes	Description
val (Conditional Formatting Bit Mask)	Specifies the set of conditional formatting properties that have been applied to this object. These properties are expressed using a string serialization of a binary bitmask for each of the following properties (reading from the first character position right): • First Row - Is this the first row of the table? • Last Row - Is this the last row of the table? • First Column - Does this belong to the first column of the table? • Last Column - Does this belong to the last column of the table? • Band 1 Vertical - Does this belong to a column which should receive band 1

Attributes	Description

	formatting? This property specifies whether the cell should receive the formatting specified for odd-numbered columns (e.g. 1,3,5,...)
	• Band 2 Vertical - Does this belong to a column which should receive band 2 formatting? This property specifies whether the cell should receive the formatting specified for even-numbered columns (e.g. 2,4,6...)
	• Band 1 Horizontal - Does this receive band 1 formatting? This property specifies whether the cell should receive the formatting specified for odd-numbered rows (e.g. 1,3,5,...)
	• Band 2 Horizontal - Does this receive band 2 formatting? This property specifies whether the cell should receive the formatting specified for even-numbered rows (e.g. 2,4,6...)
	• NE Cell - Is this part of the top-right corner of the table?
	• NW Cell - Is this part of the top-left corner of the table?
	• SE Cell - Is this part of the bottom-right corner of the table?
	• SW Cell - Is this part of the bottom-left corner of the table?
	
	For each of these properties, a value of 1 in the specified character position in the string means that the value is true, a value of 0 means false. All values must be specified.
	
	[*Example*: Consider a paragraph in the top right corner of a table with a table style applied. This paragraph would need to specify the following WordprocessingML:
	
	`<w:p>`
	`  <w:pPr>`
	`  <w:cnfStyle w:val="101000000100" />`
	`  ...`
	`  </w:pPr>`
	`  ...`
	`</w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the first column, first row, and the NW corner of the parent table by setting the appropriate bits in the val attribute. *end example*]
	
	The possible values for this attribute are defined by the ST_Cnf simple type (§2.18.11).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Cnf">
    <attribute name="val" type="ST_Cnf" use="required"/>
</complexType>
```

2.4.8 cnfStyle (Table Row Conditional Formatting)

This element specifies the set of conditional table style formatting properties which have been applied to this table row. [*Note*: This property is an optimization which is used by consumers to determine if a given property on a table row is the result of the table style conditional formatting properties vs. direct formatting on the table

cell itself. It specifies the components of the conditional formatting in the table style applied to this cell, so that the table's conditional formatting can be applied after the document is displayed without having the table style properties override the style hierarchy. *end note*]

If this element is omitted, then its value shall be assumed to be zero for all entries in the bit mask.

[*Example*: Consider a table row in the top of a table with a table style applied. This table cell would need to specify the following WordprocessingML to express that fact:

```
<w:tr>
  <w:trPr>
    <w:cnfStyle w:val="100000000000" />
    …
  </w:trPr>
  …
</w:tr>
```

This table row specifies that it has the conditional properties from the table style for the first row of the parent table by setting the appropriate bits in the val attribute. *end example*]

Parent Elements
trPr (§2.7.5.10); trPr (§2.7.5.11); trPr (§2.4.78); trPr (§2.4.79)

Attributes	Description
val (Conditional Formatting Bit Mask)	Specifies the set of conditional formatting properties that have been applied to this object. These properties are expressed using a string serialization of a binary bitmask for each of the following properties (reading from the first character position right): • First Row - Is this the first row of the table? • Last Row - Is this the last row of the table? • First Column - Does this belong to the first column of the table? • Last Column - Does this belong to the last column of the table? • Band 1 Vertical - Does this belong to a column which should receive band 1 formatting? This property specifies whether the cell should receive the formatting specified for odd-numbered columns (e.g. 1,3,5,...) • Band 2 Vertical - Does this belong to a column which should receive band 2 formatting? This property specifies whether the cell should receive the formatting specified for even-numbered columns (e.g. 2,4,6...) • Band 1 Horizontal - Does this receive band 1 formatting? This property specifies whether the cell should receive the formatting specified for odd-numbered rows (e.g. 1,3,5,...) • Band 2 Horizontal - Does this receive band 2 formatting? This property specifies

Attributes	Description

	whether the cell should receive the formatting specified for even-numbered rows (e.g. 2,4,6...)
	• NE Cell - Is this part of the top-right corner of the table?
	• NW Cell - Is this part of the top-left corner of the table?
	• SE Cell - Is this part of the bottom-right corner of the table?
	• SW Cell - Is this part of the bottom-left corner of the table?
	
	For each of these properties, a value of 1 in the specified character position in the string means that the value is true, a value of 0 means false. All values must be specified.
	
	[*Example*: Consider a paragraph in the top right corner of a table with a table style applied. This paragraph would need to specify the following WordprocessingML:
	
	`<w:p>`
	`  <w:pPr>`
	`  <w:cnfStyle w:val="101000000100" />`
	`  ...`
	`  </w:pPr>`
	`  ...`
	`</w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the first column, first row, and the NW corner of the parent table by setting the appropriate bits in the val attribute. *end example*]
	
	The possible values for this attribute are defined by the ST_Cnf simple type (§2.18.11).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Cnf">
    <attribute name="val" type="ST_Cnf" use="required"/>
</complexType>
```

2.4.9 divId (Associated HTML div ID)

This element specifies the HTML div information which is associated with the current table row. This information, stored in the Web Settings part, is used to associate one or more table rows with a particular HTML `div` element. [*Note*: This property is used when saving an HTML document into the WordprocessingML format in order to prevent a loss of all HTML `div` information, so that the document can later be saved back into HTML format and have the stored information replaced, since the HTML `div` can store formatting properties on arbitrary regions. *end note*]

In order to determine the associated HTML `div` properties, the value of the val attribute on this element is used to look up an associated div element (§2.15.2.8) whose id attribute matches this value.

If this table row does not have a divId element present, then this table row shall not have any associated HTML div information. If this element is present, but the val attribute specifies an id value which does not have an associated div element, then this element is ignored.

[*Example*: Consider an HTML document defined as follows:

```
<html>
  <body>
    <div style="…">
      <table>
        <tr>
          <td>R1C1</td>
           …
        </tr>
      </table>
      <p>
        …
      </p>
    </div>
    …
  </body>
</html>
```

This HTML document specifies a div spanning the table and the first paragraph. If this document is saved into WordprocessingML, then both the rows of the table and the paragraph shall have a divId which points at the same div information in the web settings part:

```
<w:trPr>
  …
  <w:divId w:val="1102603671"/>
</w:trPr>
```

The val attribute then points at a div element which stores the associated div properties:

```
<w:divs>
  <w:div w:id="1102603671">
    …
  </w:div>
</w:divs>
```

This specifies that this table's rows are part of a single HTML div. *end example*]

Parent Elements
trPr (§2.7.5.10); trPr (§2.7.5.11); trPr (§2.4.78); trPr (§2.4.79)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.4.10 gridAfter (Grid Columns After Last Cell)

This element specifies the number of grid columns in the parent table's table grid (§2.4.44; §2.4.45) which shall be left after the last cell in the table row.

If this element conflicts with the remaining size of the document grid after all table cells in this row have been added to the grid, then it shall be ignored. If this element is not specified, then its value shall be assumed to be zero grid units.

[*Example*: Consider a table whose second row ends before the first row by one grid unit:

In this table, the second row leaves one grid unit after its cell contents, which is represented using the following WordprocessingML:

```
<w:tr>
  <w:trPr>
    <w:gridAfter w:val="1" />
    …
  </w:trPr>
  …
</w:tr>
```

The gridAfter element specifies that 1 grid unit remains after the row's contents. *end example*]

Parent Elements
trPr (§2.7.5.10); trPr (§2.7.5.11); trPr (§2.4.78); trPr (§2.4.79)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
    <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.4.11 gridBefore (Grid Columns Before First Cell)

This element specifies the number of grid columns in the parent table's table grid (§2.4.44; §2.4.45) which shall be skipped before the contents of this table row (its table cells) are added to the parent table. [*Note*: This property is used to specify tables whose leading edge (left for left-to-right tables, right for right-to-left tables) does not start at the first grid column (the same shared edge). *end note*]

If this element is omitted, then its value shall be assumed to be zero grid units. If this element's value is larger than the size of the table grid, then the value shall be ignored and the first cell in the row may span the full table grid (i.e. the second cell, if one exists, should start at the last shared edge in the table).

[*Example*: Consider a table whose second row starts after the first grid unit:

In this table, the second row skips one grid unit at the beginning which is represented by the following WordprocessingML:

```
<w:tr>
  <w:trPr>
    <w:gridBefore w:val="1" />
    …
  </w:trPr>
  …
</w:tr>
```

end example]

Parent Elements
trPr (§2.7.5.10); trPr (§2.7.5.11); trPr (§2.4.78); trPr (§2.4.79)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.4.12 gridCol (Grid Column Definition)

This element specifies the presence and details about a single grid column within a table grid. A *grid column* is a logical column in a table used to specify the presence of a shared vertical edge in the table. When table cells are then added to this table, these shared edges (or grid columns, looking at the column between those shared edges) determine how table cells are placed into the table grid.

[*Example*: If a table row specifies that it is preceded by two grid columns, then it would start on the third vertical edge in the table including edges which are not shared by all columns. *end example*]

If the table grid does not match the requirements of one or more rows in the table (i.e. it does not define enough grid columns), then the grid may be redefined as needed when the table is processed.

[*Example*: Consider the following, more complex table that has two rows and two columns; as shown, the columns are not aligned:

This table is represented by laying out the cells on a table grid consisting of three table grid columns as follows, each grid column representing a logical vertical column in the table:

The dashed lines represent the virtual vertical continuations of each table grid column, and thus resulting table grid is represented as the following in WordprocessingML:

```
<w:tblGrid>
  <w:gridCol w:w="5051" />
  <w:gridCol w:w="3008" />
  <w:gridCol w:w="1531" />
</w:tblGrid>
```

end example]

Parent Elements
tblGrid (§2.4.44); tblGrid (§2.4.45)

Attributes	Description
w (Grid Column Width)	Specifies the width of this grid column. [*Note*: This value does not solely determine the actual width of the resulting grid column in the document. When the table is displayed in a document, these widths determine the initial width of each grid column, which may then be overridden by: • The table layout algorithm (§2.4.49;§2.4.50) applied to the current table row(s) • The preferred widths of specific cells which are part of that grid column as the table is displayed (which is an input to the algorithm above) *end note*] This value is specified in twentieths of a point.

Attributes	Description

	If this attribute is omitted, then the last saved width of the grid column is assumed to be zero.
	
	[*Example*: Consider the following table grid definition:
	
	`<w:tblGrid>`
	`  <w:gridCol w:w="6888"/>`
	`  <w:gridCol w:w="248"/>`
	`  <w:gridCol w:w="886"/>`
	`  <w:gridCol w:w="1554"/>`
	`</w:tblGrid>`
	
	This table grid specifies four grid columns, each of which has an initial size of 6888 twentieths of a point, 248 twentieths of a point, 886 twentieths of a point, and 1554 twentieths of a point respectively. *end example*]
	
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblGridCol">
    <attribute name="w" type="ST_TwipsMeasure"/>
</complexType>
```

2.4.13 gridSpan (Grid Columns Spanned by Current Table Cell)

This element specifies the number of grid columns in the parent table's table grid which shall be spanned by the current cell. This property allows cells to have the appearance of being merged, as they span vertical boundaries of other cells in the table.

If this element is omitted, then the number of grid units spanned by this cell shall be assumed to be one. If the number of grid units specified by the val attribute exceeds the size of the table grid, then the table grid shall be augmented as needed to create the number of grid columns required.

[*Example*: Consider the following table that has two rows and two columns where the columns are not aligned:

This table is represented by laying out the cells on a table grid consisting of three table grid columns, each grid column representing a logical vertical column in the table:

The first table cell in the first row spans two grid column units. The second cell in the second row also consumes two grid column units (see the grid lines represented using dotted lines in the example above). This table is represented using the following WordprocessingML:

```
<w:tbl>
  …
  <w:tr>
    <w:tc>
      <w:tcPr>
        …
        <w:gridSpan w:val="2" />
      </w:tcPr>
      …
    </w:tc>
    …
  </w:tr>
  <w:tr>
    <w:tc>
      …
    </w:tc>
    <w:tc>
      <w:tcPr>
        …
        <w:gridSpan w:val="2" />
      </w:tcPr>
      …
    </w:tc>
  </w:tr>
</w:tbl>
```

The gridSpan element indicates the number of columns spanned by each cell with respect to the table grid (in the case of R1C1 and TR2C2, two. *end example*]

Parent Elements
tcPr (§2.7.5.8); tcPr (§2.4.66); tcPr (§2.7.5.9); tcPr (§2.4.67)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element.

Attributes	Description
	[*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.4.14 hidden (Hidden Table Row Marker)

This element specifies that the glyph representing the end character of current table row shall not be displayed in the current document.

[*Note*: This setting is used to hide the end of row glyph in order to ensure that the entire table row is hidden and not displayed in the document, as if any part of the row is visible, the row is displayed. *end note*]

[*Note*: Applications may have settings which allow hidden content to be displayed, in which case this content may be visible - this property is not meant to supersede that setting. *end note*]

If this element is omitted, then this table row shall not be hidden in the document.

[*Example*: Consider a table with a table row in which the row is specified to be hidden. That requirement is specified using the following WordprocessingML:

```
<w:tbl>
  …
  <w:tr>
    <w:trPr>
      <w:hidden />
      …
    </w:trPr>
    …
  </w:tr>
</w:tbl>
```

In this example this row will not be displayed nor printed, as the hidden element is specified on in table row's properties. *end example*]

Parent Elements
trPr (§2.7.5.10); trPr (§2.7.5.11); trPr (§2.4.78); trPr (§2.4.79)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.4.15 hideMark (Ignore End Of Cell Marker In Row Height Calculation)

This element specifies whether the end of cell glyph shall influence the height of the given table row in the table. If it is specified, then only printing characters in this cell shall be used to determine the row height.

[*Rationale*: Typically, the height of a table row is determined by the height of all glyphs in all cells in that row, including the non-printing end of cell glyph characters. However, if these characters are not formatted, they are always created with the document default style properties. This means that the height of a table row cannot ever be reduced below the size of the end of cell marker glyph without manually formatting each paragraph in that run.

In a typical document, this behavior is desirable as it prevents table rows from 'disappearing' if they have no content. However, if a table row is being used as a border (for example, by shading its cells or putting an image in them), then this behavior makes it impossible to have a virtual border that is reasonably small without formatting each cell's content directly. This setting specifies that the end of cell glyph shall be ignored for this cell, allowing it to collapse to the height of its contents without formatting each cell's end of cell marker, which would have the side effect of formatting any text ever entered into that cell. *end rationale*]

If this element is omitted, then the end of cell marker shall be included in the determination of the height of this row.

[*Example*: Consider the following WordprocessingML table:

Here is some small text.			

Notice that the only printing content in this table row is displayed using 5 point font, yet the row height is influenced by the end of cell markers in the empty cells.

If each cell in the second row in this table was set to exclude the table cell from this calculation, using the following WordprocessingML:

```
<w:tcPr>
  <w:hideMark/>
</w:tcPr>
```

The resulting table shall exclude the cell markers from the row height calculation:

Here is some text.			

The hideMark element specified that each cell marker was excluded, resulting in the row height being defined by the actual run contents. *end example*]

Parent Elements
tcPr (§2.7.5.8); tcPr (§2.4.66); tcPr (§2.7.5.9); tcPr (§2.4.67)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.4.16 hMerge (Horizontally Merged Cell)

This element specifies that this cell is part of a horizontally merged set of cells in a table. The val attribute on this element determines how this cell is defined with respect to the previous cell in the table (i.e. does this cell continue the horizontal merge or start a new merged group of cells).

[*Note*: This property is maintained for compatibility with legacy word processing documents which defined tables in this manner. Whenever possible, this form or horizontal merges should not be produced, and should be translated to the appropriate gridSpan (§2.4.13) settings on the table cells instead. *end note*]

If this element is omitted, then this cell shall not be part of any horizontally merged grouping of cells, and any horizontal merge group in the preceding cells shall be closed.

[*Example*: Consider a table with one row and three columns with the last two columns horizontally merged:

The second cell in the first row starts a merge that is completed in the right adjacent cell, resulting in the following WordprocessingML:

```
<w:tbl>
   …
  <w:tr>
     <w:tc>
       …
    </w:tc>
     <w:tc>
       <w:tcPr>
         <w:hmerge w:val="restart"/>
      </w:tcPr>
      …
    </w:tc>
    <w:tc>
      <w:tcPr>
        <w:hmerge/>
      </w:tcPr>
      …
    </w:tc>
  </w:tr>
</w:tbl>
```

The hmerge element defines the cells which are horizontally merged, and how each group is merged together. *end example*]

Parent Elements
tcPr (§2.7.5.8); tcPr (§2.4.66); tcPr (§2.7.5.9); tcPr (§2.4.67)

Attributes	Description

val (Horizontal Merge Type)	Specifies how the table cell is part of a horizontally merged region. This determine whether the cell should join onto an existing grouping of merged cells if any exist, or start a new group of merged cells. Refer to the simple type definition for a full description of each type.
	
	If this attribute is omitted, its value shall be assumed to be `continue`.
	
	[*Example*: Consider a table cell where a horizontal cell merge begins represented as the following WordprocessingML:
	
	`<w:tcPr>`
	`  <w:hmerge w:val="restart"/>`
	`</w:tcPr>`
	
	The attribute value of `restart` specifies that this element shall start a new horizontally merged region in this table. *end example*]
	
	The possible values for this attribute are defined by the ST_Merge simple type (§2.18.64).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HMerge">
   <attribute name="val" type="ST_Merge"/>
</complexType>
```

2.4.17 insideH (Table Inside Horizontal Edges Border)

This element specifies the border which shall be displayed on all horizontal table cell borders which are not on an outmost edge of the parent table (all horizontal borders which are not the topmost or bottommost border). The appearance of this table cell border in the document shall be determined by the following settings:

- The display of the border on interior edges is subject to the conflict resolution algorithm defined by the tcBorders element (§2.4.63) and the tblBorders element (§2.4.37;§2.4.38)

If this element is omitted, then the inside horizontal borders of this table shall have the border specified by the associated table style. If no inside horizontal edge border is specified in the style hierarchy, then this table shall not have an inside horizontal edge border.

[*Example*: Consider a table in which the table specifies a border on all interior horizontal and vertical edges, as follows:

R1C1	R1C2
R2C1	R2C2

This interior horizontal cell border is specified using the following WordprocessingML:

```
<w:tblPr>
  <w:tblBorders>
    <w:insideH w:val="doubleWave" w:sz="6" w:space="0" w:color="92D050"/>
    <w:insideV w:val="doubleWave" w:sz="6" w:space="0" w:color="92D050"/>
  </w:tblBorders>
  …
</w:tblPr>
```

The insideH element specifies a ¾ point border of type doubleWave. *end example*]

Parent Elements
tblBorders (§2.4.37); tblBorders (§2.4.38)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or auto to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value auto, as follows: <w:bottom … w:color="auto"/> This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest

Attributes	Description
	from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is `true`, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object. When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points. When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points. [*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows: `<w:pgBorders w:offsetFrom="page">`

Attributes	Description
	<w:bottom … w:space="24/> </w:pgBorders> The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*] The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).
sz (Border Width)	Specifies the width of the current border. If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value. If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value. [*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup: <w:top w:val="dashed" w:sz="24" …/> <w:left w:val="dashed" w:sz="24" …/> <w:bottom w:val="dashed" w:sz="24" …/> <w:right w:val="dashed" w:sz="24" …/> The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*] The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).
themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border. The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document. [*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup: <w:top … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:bottom … w:color="FFA8A0" w:themeColor="accent2"

Attributes	Description
	w:themeTint="99" /> <w:left … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:right … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*] The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).
themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255$$ $$= 102$$ $$= 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get:

Attributes	Description
	$$L^{'} = 0.53 * 0.75$$ $$= 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255$$ $$= 153$$ $$= 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}^{'} = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD.

Attributes	Description
	The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $L' = 0.53 * 0.6 + (1 - .6)$ $= 0.71$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is single, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <attribute name="val" type="ST_Border" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
   <attribute name="space" type="ST_PointMeasure" use="optional"/>
   <attribute name="shadow" type="ST_OnOff" use="optional"/>
   <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.4.18 insideH (Table Cell Inside Horizontal Edges Border)

This element specifies the border which shall be displayed on all interior horizontal edges of the current group of table cells. [*Note*: Although individual table cells have no concept of an internal edge, which would render this property useless in most cases, it is used to determine the cell borders to apply to a specific group of cells as part of table conditional formatting in a table style, for example, the inside horizontal edges on the set of cells in the first column. *end note*]

The appearance of this table cell border in the document shall be determined by the following settings:

- If the net tblCellSpacing element value (§2.4.41;§2.4.42;§2.4.43) applied to the cell is non-zero, then the cell border shall always be displayed
- Otherwise, the display of the border is subject to the conflict resolution algorithm defined by the tcBorders element (§2.4.63) and the tblBorders element (§2.4.37;§2.4.38)

If this element is omitted, then the specified conditional formatting on the table shall not change the current set of internal edge borders on its set of table cells (i.e. their current setting shall remain unchanged).

[*Example*: Consider a table in which the conditional formatting on the first column specified in the associated table style specifies a double line red cell border for all internal horizontal lines as follows:

R1C1	R1C2
R2C1	R2C2

This inner horizontal edge cell border is specified using the following WordprocessingML:

```
<w:tblStylePr w:type="firstColumn">
  <w:tcPr>
    <w:tcBorders>
      <w:insideH w:val="double" w:sz="4" w:space="0" w:color="FF0000"/>
    </w:tcBorders>
  </w:tcPr>
</w:tblStylePr>
```

The insideH element specifies a ¼ point border of type double. *end example*]

Parent Elements
tcBorders (§2.4.63)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or auto to allow a

Attributes	Description
	consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value auto, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is `true`, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border.

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object.
	
	When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points.
	
	When a document has a page border that is relative to the text extents (using a value of text in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.
	
	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24/>`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*]
	
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).

sz (Border Width)	Specifies the width of the current border.
	
	If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value.
	
	If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value.
	
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`

Attributes	Description
	The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*] The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).
themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border. The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document. [*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup: `<w:top … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:bottom … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:left … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:right … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />` The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*] The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).
themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255 = 102 = 66(hex)$$

Attributes	Description
	The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$L' = L * Shade_{percentage}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L' = 0.53 * 0.75$$ $$= 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows:

Attributes	Description
	$$T_{xml} = 0.6 * 255$$ $$= 153$$ $$= 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6)$$ $$= 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$ and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML:

Attributes	Description
	<w:left w:val="single" …/> This border's val is single, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <attribute name="val" type="ST_Border" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
   <attribute name="space" type="ST_PointMeasure" use="optional"/>
   <attribute name="shadow" type="ST_OnOff" use="optional"/>
   <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.4.19 insideV (Table Cell Inside Vertical Edges Border)

This element specifies the border which shall be displayed on all interior vertical edges of the current group of table cells. [*Note*: Although individual table cells have no concept of an internal edge, which would render this property useless in most cases, it is used to determine the cell borders to apply to a specific group of cells as part of table conditional formatting in a table style, for example, the inside vertical edges on the set of cells in the header row. *end note*]

The appearance of this table cell border in the document shall be determined by the following settings:

- If the net tblCellSpacing element value (§2.4.41;§2.4.42;§2.4.43) applied to the cell is non-zero, then the cell border shall always be displayed
- Otherwise, the display of the border is subject to the conflict resolution algorithm defined by the tcBorders element (§2.4.63) and the tblBorders element (§2.4.37;§2.4.38)

If this element is omitted, then the specified conditional formatting on the table shall not change the current set of internal edge borders on its set of table cells (i.e. their current setting shall remain unchanged).

[*Example*: Consider a table in which the conditional formatting on the header row in the associated table style specifies a double line red cell border for all internal vertical lines as follows:

R1C1	R1C2
R2C1	R2C2

This inner vertical edge cell border is specified using the following WordprocessingML:

```
<w:tblStylePr w:type="firstRow">
  <w:tcPr>
    <w:tcBorders>
      <w:insideV w:val="double" w:sz="4" w:space="0" w:color="FF0000"/>
    </w:tcBorders>
  </w:tcPr>
</w:tblStylePr>
```

The insideV element specifies a ¼ point border of type double. *end example*]

Parent Elements
tcBorders (§2.4.63)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or auto to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value auto, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is true, indicating that the border frame effect shall be applied. *end*

Attributes	Description
	example] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object.
	
	When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points.
	
	When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.
	
	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24/>`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*]
	
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).

sz (Border Width)	Specifies the width of the current border.

Attributes	Description

	If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value.
	
	If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value.
	
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`
	
	The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	
	The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).

themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border.
	
	The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document.
	
	[*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup:
	
	`<w:top … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:bottom … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:left … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:right … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	
	The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).
themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255 = 102 = 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L' = 0.53 * 0.75 = 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$ and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2"`

Attributes	Description
	`w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255$$ $$= 153$$ $$= 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `4F81BD`. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6)$$ $$= 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get `95B3D7`.

Attributes	Description
	This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
    <attribute name="val" type="ST_Border" use="required"/>
    <attribute name="color" type="ST_HexColor" use="optional"/>
    <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
    <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
    <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
    <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
    <attribute name="space" type="ST_PointMeasure" use="optional"/>
    <attribute name="shadow" type="ST_OnOff" use="optional"/>
    <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.4.20 insideV (Table Inside Vertical Edges Border)

This element specifies the border which shall be displayed on all vertical table cell borders which are not on an outmost edge of the parent table (all horizontal borders which are not the leftmost or rightmost border). The appearance of this table cell border in the document shall be determined by the following settings:

- The display of the border on interior edges is subject to the conflict resolution algorithm defined by the tcBorders element (§2.4.63) and the tblBorders element (§2.4.37;§2.4.38)

If this element is omitted, then the inside vertical borders of this table shall have the border specified by the associated table style. If no inside vertical edge border is specified in the style hierarchy, then those cells in this table shall not have an inside vertical edge border.

[*Example*: Consider a table in which the table specifies a border on all interior horizontal and vertical edges, as follows:

R1C1	R1C2
R2C1	R2C2

This interior horizontal cell border is specified using the following WordprocessingML:

```
<w:tblPr>
  <w:tblBorders>
    <w:insideH w:val="doubleWave" w:sz="6" w:space="0" w:color="92D050"/>
    <w:insideV w:val="doubleWave" w:sz="6" w:space="0" w:color="92D050"/>
  </w:tblBorders>
  …
</w:tblPr>
```

The insideV element specifies a ¾ point border of type doubleWave on all interior vertical edges. *end example*]

Parent Elements
tblBorders (§2.4.37); tblBorders (§2.4.38)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or auto to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value auto, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value.

Attributes	Description
	The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is `true`, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object. When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points. When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.

Attributes	Description

	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24/>`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*]
	
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).

sz (Border Width)	Specifies the width of the current border.
	
	If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value.
	
	If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value.
	
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`
	
	The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	
	The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).

themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border. The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document. [*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup:

Attributes	Description
	<w:top … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:bottom … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:left … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:right … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*] The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).
themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255 = 102 = 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D.

Attributes	Description
	The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $L^{'} = 0.53 * 0.75$ $= 0.39698$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $T_{xml} = 0.6 * 255$ $= 153$ $= 99(hex)$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $\mathrm{L}^{'} = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$ • Convert the resultant HSL color to RGB

Attributes	Description
	[*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `4F81BD`. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $L' = 0.53 * 0.6 + (1 - .6)$ $= 0.71$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get `95B3D7`. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
    <attribute name="val" type="ST_Border" use="required"/>
    <attribute name="color" type="ST_HexColor" use="optional"/>
    <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
    <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
    <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
    <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
    <attribute name="space" type="ST_PointMeasure" use="optional"/>
    <attribute name="shadow" type="ST_OnOff" use="optional"/>
    <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.4.21 jc (Table Alignment Exception)

This element specifies the alignment of the set of rows which are part of the current table properties exception list with respect to the text margins in the current section. When a table is placed in a WordprocessingML document that does not have the same width as the margins, this property is used to determine how the table is positioned with respect to those margins. The interpretation of property is reversed if the parent table is right to left using the bidiVisual element (§2.4.1).

If this property is omitted on a table, then the justification shall be determined by the default set of table properties on the parent table.

[*Example*: Consider the following WordprocessingML table, centered on the text margins with a subset of its rows justified to the left margin by a table property exception:

That exception would be specified using the following WordprocessingML:

```
<w:tblPrEx>
  <w:jc w:val="left"/>
</w:tblPrEx>
```

The jc element specifies that the rows which are part of the table properties exception table shall be `left` aligned with respect to the text margins. *end example*]

Parent Elements
tblPrEx (§2.4.57); tblPrEx (§2.4.58)

Attributes	Description

val (Alignment Type)	Specifies the justification which should be applied to the parent object within a document.
	
	The possible values (see below) for this attribute are always specified relative to the page, and do not change semantic from right-to-left and left-to-right documents.
	
	[*Example*: Consider the following WordprocessingML fragment for a paragraph in a document:
	
	`<w:pPr>`
	`  <w:jc w:val="right" />`
	`</w:pPr>`
	
	This paragraph is now right justified on the page, regardless of the paragraph or section settings. *end example*]
	
	The possible values for this attribute are defined by the ST_Jc simple type (§2.18.50).

The following XML Schema fragment defines the contents of this element:

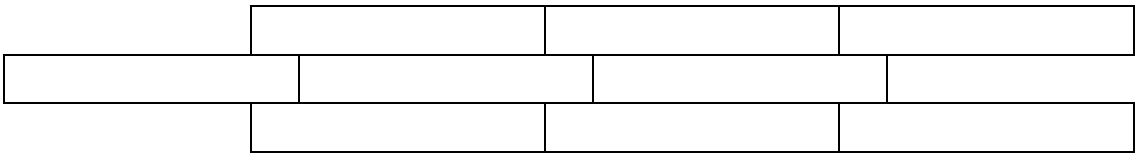
```
<complexType name="CT_Jc">
   <attribute name="val" type="ST_Jc" use="required"/>
</complexType>
```

2.4.22 jc (Table Row Alignment)

This element specifies the alignment of a single row in the parent table with respect to the text margins in the current section. When a table is placed in a WordprocessingML document that does not have the same width as the margins, this property is used to determine how a specific row in that table is positioned with respect to those margins. The interpretation of property is reversed if the parent table is right to left using the bidiVisual element (§2.4.1).

If this property is omitted on a table, then the justification shall be determined by the default set of table properties on the parent table.

[*Example*: Consider the following WordprocessingML table, centered on the text margins with its second rows justified to the left margin by a table row level justification:



That row level setting would be specified using the following WordprocessingML:

```
<w:trPr>
  <w:jc w:val="left"/>
</w:trPr>
```

The jc element specifies that the rows which are part of the table properties exception table shall be `left` aligned with respect to the text margins. *end example*]

Parent Elements
trPr (§2.7.5.10); trPr (§2.7.5.11); trPr (§2.4.78); trPr (§2.4.79)

Attributes	Description

val (Alignment Type)	Specifies the justification which should be applied to the parent object within a document.
	
	The possible values (see below) for this attribute are always specified relative to the page, and do not change semantic from right-to-left and left-to-right documents.
	
	[*Example*: Consider the following WordprocessingML fragment for a paragraph in a document:
	
	`<w:pPr>`
	`  <w:jc w:val="right" />`
	`</w:pPr>`
	
	This paragraph is now right justified on the page, regardless of the paragraph or section settings. *end example*]
	
	The possible values for this attribute are defined by the ST_Jc simple type (§2.18.50).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Jc">
   <attribute name="val" type="ST_Jc" use="required"/>
</complexType>
```

2.4.23 jc (Table Alignment)

This element specifies the alignment of the current table with respect to the text margins in the current section. When a table is placed in a WordprocessingML document that does not have the same width as the margins, this property is used to determine how the table is positioned with respect to those margins. The interpretation of property is reversed if the parent table is right to left using the bidiVisual element (§2.4.1).

If this property is omitted on a table, then the justification shall be determined by the associated table style. If this property is not specified in the style hierarchy, then the table shall be left justified with zero indentation from the leading margin (the left margin in a left-to-right table or the right margin in a right-to-left table).

[*Example*: Consider the following WordprocessingML table, justified to the left margin by default:

R1C1	R1C2	R1C3
R2C1	R2C2	R2C3

This table does not fill the entire width of the text margins. If the table should be right justified to the margin, as follows:

R1C1	R1C2	R1C3
R2C1	R2C2	R2C3

That requirement would be specified using the following WordprocessingML:

```
<w:tblPr>
  <w:jc w:val="right"/>
</w:tblPr>
```

The jc element specifies that the table shall be `right` aligned with respect to the text margins. *end example*]

Parent Elements
tblPr (§2.7.5.3); tblPr (§2.7.5.4); tblPr (§2.4.55); tblPr (§2.4.56)

Attributes	Description

val (Alignment Type)	Specifies the justification which should be applied to the parent object within a document.
	
	The possible values (see below) for this attribute are always specified relative to the page, and do not change semantic from right-to-left and left-to-right documents.
	
	[*Example*: Consider the following WordprocessingML fragment for a paragraph in a document:
	
	`<w:pPr>`
	`  <w:jc w:val="right" />`
	`</w:pPr>`
	
	This paragraph is now right justified on the page, regardless of the paragraph or section settings. *end example*]
	
	The possible values for this attribute are defined by the ST_Jc simple type (§2.18.50).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Jc">
   <attribute name="val" type="ST_Jc" use="required"/>
</complexType>
```

2.4.24 left (Table Cell Left Border)

This element specifies the border which shall be displayed on the left side of the current table cell. The appearance of this table cell border in the document shall be determined by the following settings:

- If the net tblCellSpacing element value (§2.4.41;§2.4.42;§2.4.43) applied to the cell is non-zero, then the cell border shall always be displayed
- Otherwise, the display of the border is subject to the conflict resolution algorithm defined by the tcBorders element (§2.4.63) and the tblBorders element (§2.4.37;§2.4.38)

If this element is omitted, then the left side of this table cell shall not have a cell border, and its border may use the table's border settings as appropriate.

[*Example*: Consider a table in which the second cell in the first row specifies a left cell border

R1C1	R1C2
R2C1	R2C2

This left cell border is specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    …
    <w:tcBorders>
      <w:left w:val="double" w:sz="4" w:space="0" w:color="FF0000" />
    </w:tcBorders>
  </w:tcPr>
  <w:p/>
</w:tc>
```

The left element specifies a ½ point border of type double on the table cell. *end example*]

Parent Elements
tcBorders (§2.4.63)

Attributes	Description
color (Border Color)	Specifies the color for this border.

Attributes	Description
	This color may either be presented as a hex value (in RRGGBB format), or auto to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value auto, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is true, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />`

Attributes	Description
	This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object.
	
	When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points.
	
	When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.
	
	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24/>`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*]
	
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).

sz (Border Width)	Specifies the width of the current border.
	
	If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value.
	
	If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value.
	
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`

Attributes	Description
	The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*] The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).

themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border.
	
	The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document.
	
	[*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup:
	
	`<w:top … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />`
	`<w:bottom … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />`
	`<w:left … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />`
	`<w:right … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />`
	
	The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*]
	
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).

themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255 = 102 = 66(hex)$$

Attributes	Description
	The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$L' = L * Shade_{percentage}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L' = 0.53 * 0.75$$ $$= 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows:

Attributes	Description
	$$T_{xml} = 0.6 * 255$$ $$= 153$$ $$= 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6)$$ $$= 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$ and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML:

Attributes	Description
	<w:left w:val="single" …/> This border's val is single, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
    <attribute name="val" type="ST_Border" use="required"/>
    <attribute name="color" type="ST_HexColor" use="optional"/>
    <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
    <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
    <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
    <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
    <attribute name="space" type="ST_PointMeasure" use="optional"/>
    <attribute name="shadow" type="ST_OnOff" use="optional"/>
    <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.4.25 left (Table Cell Left Margin Exception)

This element specifies the amount of space which shall be left between the left extent of the current cell contents and the left border of a specific individual table cell within a table. This setting shall override the table cell bottom margin definition specified by the left element contained within the table properties (§2.4.26).

This value is specified in the units applied via its type attribute. Any width value of type pct or auto for this element shall be ignored.

If omitted, then this table cell shall use the bottom cell margins defined in the left element contained within the table properties (§2.4.26).

[*Example*: Consider a two row, two column table in which the first table cell in the second row has a left margin which is specified via an exception to be 0.5 inches, causing the text to be position 0.5" inside the cell, as follows:

R1C1	R1C2
R2C1	R2C2

The exception on this cell would be specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    <w:tcMar>
      <w:left w:w="720" w:type="dxa" />
    </w:tcMar>
  </w:tcPr>
</w:tc>
```

The R2C1 cell in this table has an exception applied to the table cell left cell margin setting it to 720 twentieths of a point (0.5 inches). *end example*]

Parent Elements
tcMar (§2.4.65)

Attributes	Description
type (Table Width Type)	Specifies the units of the width property being defined by the parent element's w attribute. This property is used to define various properties of a table, including: cell spacing, preferred width, and table margins. If this attribute is omitted, then its value shall be assumed to be dxa (twentieths of a point). [*Example*: Consider a table with a table cell bottom cell spacing with a type of dxa, as follows: `<w:bottom ... w:type="dxa" />` This type shall therefore be used to interpret the width specified in the w attribute as a value in twentieths of a point. *end example*] The possible values for this attribute are defined by the ST_TblWidth simple type (§2.18.97).
w (Table Width Value)	Specifies the value of the width property being defined by the parent element. This property is used to define various properties of a table, including: cell spacing, preferred widths, and table margins. If this attribute is omitted, then its value shall be assumed to be 0. [*Example*: Consider a table with a bottom margin with a width of 302, as follows: `<w:bottom w:w="302" w:type="dxa" />` The value in the w attribute shall therefore be used to determine the width being specified in the context of the units specified in the type attribute. In this case, the type is twentieths of a point (dxa), so the width is 302 twentieths of a point (.2097 inches).

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblWidth">
   <attribute name="w" type="ST_DecimalNumber"/>
   <attribute name="type" type="ST_TblWidth"/>
</complexType>
```

2.4.26 left (Table Cell Left Margin Default)

This element specifies the amount of space which shall be left between the left extent of the cell contents and the left border of all table cells within the parent table (or table row). This setting may be overridden by the table cell bottom margin definition specified by the left element contained within the table cell's properties (§2.4.26).

This value is specified in the units applied via its type attribute. Any width value of type `pct` or `auto` for this element shall be ignored.

If this element is omitted, then it shall inherit the table cell margin from the associated table style. If a left margin is never specified in the style hierarchy, this table shall have 115 twentieths of a point (0.08 inches) left cell padding by default (excepting individual cell overrides).

[*Example*: Consider a two by two table in which the default table cell left margin is specified to be exactly 0.25 inches, as follows (marked with an arrow in the first table cell below):

↔ R1C1	R2C1
R2C1	R2C2

This table property is specified using the following WordprocessingML markup:

```
<w:tbl>
  <w:tblPr>
    <w:tblCellMar>
      <w:left w:w="360" w:type="dxa"/>
    </w:tblCellMar>
  </w:tblPr>
  …
</w:tbl>
```

Every cell in the table has a default left cell margin setting it to `360` twentieths of a point. *end example*]

Parent Elements
tblCellMar (§2.4.39); tblCellMar (§2.4.40)

Attributes	Description
type (Table Width Type)	Specifies the units of the width property being defined by the parent element's w attribute. This property is used to define various properties of a table, including: cell spacing, preferred width, and table margins. If this attribute is omitted, then its value shall be assumed to be dxa (twentieths of a point). [*Example*: Consider a table with a table cell bottom cell spacing with a type of dxa, as follows: `<w:bottom ... w:type="dxa" />` This type shall therefore be used to interpret the width specified in the w attribute as a value in twentieths of a point. *end example*] The possible values for this attribute are defined by the ST_TblWidth simple type (§2.18.97).
w (Table Width Value)	Specifies the value of the width property being defined by the parent element. This property is used to define various properties of a table, including: cell spacing, preferred widths, and table margins. If this attribute is omitted, then its value shall be assumed to be 0. [*Example*: Consider a table with a bottom margin with a width of 302, as follows: `<w:bottom w:w="302" w:type="dxa" />` The value in the w attribute shall therefore be used to determine the width being specified in the context of the units specified in the type attribute. In this case, the type is twentieths of a point (dxa), so the width is 302 twentieths of a point (.2097 inches). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblWidth">
   <attribute name="w" type="ST_DecimalNumber"/>
   <attribute name="type" type="ST_TblWidth"/>
</complexType>
```

2.4.27 left (Table Left Border)

This element specifies the border which shall be displayed at the left edge of the current table. The appearance of this table border in the document shall be determined by the following settings:

- The display of the border is subject to the conflict resolution algorithm defined by the tcBorders element (§2.4.63) and the tblBorders element (§2.4.37;§2.4.38)

If this element is omitted, then the left edge of this table shall have the border specified by the associated table style. If no left border is specified in the style hierarchy, then this table shall not have a left border.

[*Example*: Consider a table in which the table properties specify a left table border, as follows:

R1C1	R1C2
R2C1	R2C2

This left table border is specified using the following WordprocessingML:

```
<w:tbl>
  <w:tblPr>
    <w:tblBorders>
      <w:left w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
w:color="D0D0D0" w:themeColor="accent3" w:themeTint="99"/>
    </w:tblBorders>
  </w:tblPr>
  …
</w:tbl>
```

The left element specifies a three point left table border of type `thinThinThickMediumGap`. *end example*]

Parent Elements
tblBorders (§2.4.37); tblBorders (§2.4.38)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or `auto` to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value `auto`, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for

Attributes	Description
	example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is `true`, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object. When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance

Attributes	Description

	between the edge of the page and the beginning of this border in points.
	
	When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.
	
	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24/>`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*]
	
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).

sz (Border Width)	Specifies the width of the current border.
	
	If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value.
	
	If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value.
	
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`
	
	The border style is specified using the val attribute, and because that border style is a line border (`dashed`), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	
	The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).

themeColor	Specifies a theme color to be applied to the current border.

Attributes	Description

(Border Theme Color)	The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document.
	
	[*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup:
	
	`<w:top … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />`
	`<w:bottom … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />`
	`<w:left … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />`
	`<w:right … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />`
	
	The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*]
	
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).

themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255$$ $$= 102$$ $$= 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows:

Attributes	Description
	$$L^{'} = L * Shade_{percentage}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `C0504D`. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L^{'} = 0.53 * 0.75$$ $$= 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get `943634`. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255$$ $$= 153$$ $$= 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied

Attributes	Description
	as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$L' = L * Tint_{pct} + (1 - Tint_{pct})$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `4F81BD`. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6)$$ $$= 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get `95B3D7`. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <attribute name="val" type="ST_Border" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
   <attribute name="space" type="ST_PointMeasure" use="optional"/>
   <attribute name="shadow" type="ST_OnOff" use="optional"/>
   <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.4.28 noWrap (Don't Wrap Cell Content)

This element specifies how this table cell shall be laid out when the parent table is displayed in a document. This setting only affects the behavior of the cell when the tblLayout for this row (§2.4.49; §2.4.50) is set to use the auto algorithm.

This setting shall be interpreted in the context of the tcW element (§2.4.68) as follows:

- If the table cell width has a type attribute value of fixed, then this element specifies that that this table cell shall never be smaller than that fixed value when other cells on the line are not at their absolute minimum width.
- If the table cell width has a type attribute value of pct or auto, then this element specifies that when running the auto fit algorithm, the contents of that this table cell shall be treated as though they have no breaking characters (the contents should be treated as a single contiguous non-breaking string)

If this element is omitted, then cell content shall be allowed to wrap (the cell may be shrunk as needed if it is a fixed preferred width value, and the contents shall be treated as having breaking characters if it is a percentage or automatic width value).

[*Example*: Consider the following three row by three column WordprocessingML table:

In this table, each cell has a fixed preferred width of 2.38 inches (3427 twentieths of a point), and the tblLayout for this row (§2.4.49; §2.4.50) is set to use the auto algorithm. If a long non breaking string is added to the middle row, as follows, the two cells are adjusted to override their preferences and accommodate the string:

	sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss	

However, if the first table cell has the noWrap element present as follows:

```
<w:tcPr>
  <w:noWrap/>
</w:tcPr>
```

The noWrap element specifies that because it is a fixed width cell, that cell shall not be collapsed beyond its original size until all other cells are at their minimum size, so in this example the cell maintains its width:

	sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss	

end example]

Parent Elements
tcPr (§2.7.5.8); tcPr (§2.4.66); tcPr (§2.7.5.9); tcPr (§2.4.67)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.4.29 right (Table Cell Right Margin Default)

This element specifies the amount of space which shall be present between the right extent of the cell contents and the right border of all table cells within the parent table (or table row). This setting may be overridden by the table cell bottom margin definition specified by the right element contained within the table cell's properties (§2.4.31).

This value is specified in the units applied via its type attribute. Any width value of type `pct` or `auto` for this element shall be ignored.

If this element is omitted, then it shall inherit the table cell margin from the associated table style. If a right margin is never specified in the style hierarchy, this table shall have 115 twentieths of a point (0.08 inches) left cell padding by default (excepting individual cell overrides).

[*Example*: Consider a two by two table in which the default table cell right margin is specified to be exactly 0.25 inches, as follows (marked with an arrow in the first table cell below):

R1C1 ↔	R2C1
R2C1	R2C2

This table property is specified using the following WordprocessingML markup:

```
<w:tbl>
  <w:tblPr>
    <w:tblCellMar>
      <w:right w:w="360" w:type="dxa"/>
    </w:tblCellMar>
  </w:tblPr>
  …
</w:tbl>
```

Every cell in the table has a default right cell margin setting it to `360` twentieths of a point. *end example*]

Parent Elements
tblCellMar (§2.4.39); tblCellMar (§2.4.40)

Attributes	Description
type (Table Width Type)	Specifies the units of the width property being defined by the parent element's w attribute. This property is used to define various properties of a table, including: cell spacing, preferred width, and table margins. If this attribute is omitted, then its value shall be assumed to be dxa (twentieths of a point).

Attributes	Description
	[*Example*: Consider a table with a table cell bottom cell spacing with a type of dxa, as follows: `<w:bottom ... w:type="dxa" />` This type shall therefore be used to interpret the width specified in the w attribute as a value in twentieths of a point. *end example*] The possible values for this attribute are defined by the ST_TblWidth simple type (§2.18.97).
w (Table Width Value)	Specifies the value of the width property being defined by the parent element. This property is used to define various properties of a table, including: cell spacing, preferred widths, and table margins. If this attribute is omitted, then its value shall be assumed to be 0. [*Example*: Consider a table with a bottom margin with a width of 302, as follows: `<w:bottom w:w="302" w:type="dxa" />` The value in the w attribute shall therefore be used to determine the width being specified in the context of the units specified in the type attribute. In this case, the type is twentieths of a point (dxa), so the width is 302 twentieths of a point (.2097 inches). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblWidth">
   <attribute name="w" type="ST_DecimalNumber"/>
   <attribute name="type" type="ST_TblWidth"/>
</complexType>
```

2.4.30 right (Table Cell Right Border)

This element specifies the border which shall be displayed on the right side of the current table cell. The appearance of this table cell border in the document shall be determined by the following settings:

- If the net tblCellSpacing element value (§2.4.41;§2.4.42;§2.4.43) applied to the cell is non-zero, then the cell border shall always be displayed
- Otherwise, the display of the border is subject to the conflict resolution algorithm defined by the tcBorders element (§2.4.63) and the tblBorders element (§2.4.37;§2.4.38)

If this element is omitted, then the right side of this table cell shall not have a cell border, and its border may use the table's border settings as appropriate.

[*Example*: Consider a table in which the second cell in the first row specifies a right cell border as follows:

R1C1	R1C2
R2C1	R2C2

This right cell border is specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    …
    <w:tcBorders>
      <w:right w:val="double" w:sz="4" w:space="0" w:color="FF0000" />
    </w:tcBorders>
  </w:tcPr>
  <w:p/>
</w:tc>
```

The right element specifies a ½ point border of type `double` on the right side of the table cell. *end example*]

Parent Elements
tcBorders (§2.4.63)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or `auto` to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value `auto`, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value.

Attributes	Description
	The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is `true`, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object. When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points. When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.

Attributes	Description

	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24/>`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*]
	
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).

sz (Border Width)	Specifies the width of the current border.
	
	If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value.
	
	If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value.
	
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`
	
	The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	
	The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).

themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border. The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document. [*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup:

Attributes	Description
	<w:top … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:bottom … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:left … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:right … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*] The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).
themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255 = 102 = 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D.

Attributes	Description
	The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $L^{'} = 0.53 * 0.75$ $= 0.39698$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $T_{xml} = 0.6 * 255$ $= 153$ $= 99(hex)$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $\mathrm{L}^{'} = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$ • Convert the resultant HSL color to RGB

Attributes	Description
	[*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `4F81BD`. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $L' = 0.53 * 0.6 + (1 - .6)$ $= 0.71$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get `95B3D7`. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
    <attribute name="val" type="ST_Border" use="required"/>
    <attribute name="color" type="ST_HexColor" use="optional"/>
    <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
    <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
    <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
    <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
    <attribute name="space" type="ST_PointMeasure" use="optional"/>
    <attribute name="shadow" type="ST_OnOff" use="optional"/>
    <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.4.31 right (Table Cell Right Margin Exception)

This element specifies the amount of space which shall be present between the right extent of the current cell's text contents and the right border of a specific individual table cell within a table. This setting shall override the table cell bottom margin definition specified by the right element contained within the table properties (§2.4.29).

This value is specified in the units applied via its type attribute. Any width value of type `pct` or `auto` for this element shall be ignored.

If omitted, then this table cell shall use the bottom cell margins defined in the right element contained within the table properties (§2.4.29).

[*Example*: Consider a two row, two column table in which the first table cell in the second row has a right margin which is specified via an exception to be 0.5 inches, the region marked with an arrow in the table below:

R1C1	R1C2
R2C1 ⟷	R2C2

The exception on this cell would be specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    <w:tcMar>
      <w:right w:w="720" w:type="dxa" />
    </w:tcMar>
  </w:tcPr>
  …
</w:tc>
```

The R2C1 cell in this table has an exception applied to the table cell right cell margin setting it to `720` twentieths of a point (0.5 inches). *end example*]

Parent Elements
tcMar (§2.4.65)

Attributes	Description
type (Table Width Type)	Specifies the units of the width property being defined by the parent element's w attribute. This property is used to define various properties of a table, including: cell spacing, preferred width, and table margins. If this attribute is omitted, then its value shall be assumed to be dxa (twentieths of a point). [*Example*: Consider a table with a table cell bottom cell spacing with a type of dxa, as follows: `<w:bottom ... w:type="dxa" />` This type shall therefore be used to interpret the width specified in the w attribute as a value in twentieths of a point. *end example*] The possible values for this attribute are defined by the ST_TblWidth simple type (§2.18.97).
w (Table Width Value)	Specifies the value of the width property being defined by the parent element. This property is used to define various properties of a table, including: cell spacing, preferred widths, and table margins. If this attribute is omitted, then its value shall be assumed to be 0. [*Example*: Consider a table with a bottom margin with a width of 302, as follows: `<w:bottom w:w="302" w:type="dxa" />` The value in the w attribute shall therefore be used to determine the width being specified in the context of the units specified in the type attribute. In this case, the type is twentieths of a point (dxa), so the width is 302 twentieths of a point (.2097 inches). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblWidth">
   <attribute name="w" type="ST_DecimalNumber"/>
   <attribute name="type" type="ST_TblWidth"/>
</complexType>
```

2.4.32 right (Table Right Border)

This element specifies the border which shall be displayed at the right edge of the current table. The appearance of this table border in the document shall be determined by the following settings:

- The display of the border is subject to the conflict resolution algorithm defined by the tcBorders element (§2.4.63) and the tblBorders element (§2.4.37;§2.4.38)

If this element is omitted, then the right edge of this table shall have the border specified by the associated table style. If no right border is specified in the style hierarchy, then this table shall not have a right border.

[*Example*: Consider a table in which the table properties specify a right table border, as follows:

R1C1	R1C2
R2C1	R2C2

This right table border is specified using the following WordprocessingML:

```
<w:tbl>
  <w:tblPr>
    <w:tblBorders>
      <w:right w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
w:color="D0D0D0" w:themeColor="accent3" w:themeTint="99"/>
    </w:tblBorders>
  </w:tblPr>
  …
</w:tbl>
```

The right element specifies a three point right table border of type `thinThinThickMediumGap`. *end example*]

Parent Elements
tblBorders (§2.4.37); tblBorders (§2.4.38)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or `auto` to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value `auto`, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for

Attributes	Description
	example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is `true`, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object. When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance

Attributes	Description

	between the edge of the page and the beginning of this border in points.
	
	When a document has a page border that is relative to the text extents (using a value of text in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.
	
	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24/>`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*]
	
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).

sz (Border Width)	Specifies the width of the current border.
	
	If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value.
	
	If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value.
	
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`
	
	The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	
	The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).

themeColor	Specifies a theme color to be applied to the current border.

Attributes	Description

(Border Theme Color)	The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document.
	
	[*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup:
	
	`<w:top … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />`
	`<w:bottom … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />`
	`<w:left … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />`
	`<w:right … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />`
	
	The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*]
	
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).

themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255$$ $$= 102$$ $$= 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows:

Attributes	Description
	$$L^{'} = L * Shade_{percentage}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `C0504D`. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L^{'} = 0.53 * 0.75$$ $$= 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get `943634`. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255$$ $$= 153$$ $$= 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied

Attributes	Description
	as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `4F81BD`. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6)$$ $$= 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get `95B3D7`. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
    <attribute name="val" type="ST_Border" use="required"/>
    <attribute name="color" type="ST_HexColor" use="optional"/>
    <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
    <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
    <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
    <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
    <attribute name="space" type="ST_PointMeasure" use="optional"/>
    <attribute name="shadow" type="ST_OnOff" use="optional"/>
    <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.4.33 shd (Table Cell Shading)

This element specifies the shading which shall be applied to the extents of the current table cell. Similarly to paragraph shading, this shading shall be applied to the contents of the cell up to the cell borders, regardless of the presence of text.

This shading consists of three components:

- Background Color
- (optional) Pattern
- (optional) Pattern Color

The resulting shading is applied by setting the background color behind the paragraph, then applying the pattern color using the mask supplied by the pattern over that background.

If this element is omitted, then the cell shading shall be determined by the table-level or table-level exception cell shading settings (§2.4.34;§2.4.35) for the current table.

[*Example*: Consider a table in which the first cell in the first row has cell-level red shading, as follows:

R1C1	R1C2
R2C1	R2C2

This cell shading would be specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    <w:shd w:val="clear" w:color="auto" w:fill="FF0000" />
  </w:tcPr>
</w:tc>
```

The shd element specifies cell shading with a `clear` pattern using a background color of `FF0000` (red). *end example*]

Parent Elements
tcPr (§2.7.5.8); tcPr (§2.4.66); tcPr (§2.7.5.9); tcPr (§2.4.67)

Attributes	Description
color (Shading Pattern Color)	Specifies the color used for any foreground pattern specified for this shading using the val attribute. This color may either be presented as a hex value (in RRGGBB format), or auto to allow a consumer to automatically determine the foreground shading color as appropriate. If the shading style (the val attribute) specifies the use of no shading format or is omitted, then this property has no effect. Also, if the shading specifies the use of a theme color via the themeColor attribute, then this value is superseded by the theme color value. If this attribute is omitted, then its value shall be assumed to be auto. [*Example*: Consider a shading of type pct20 with a foreground color value of auto, as follows: `<w:shd w:val="pct20"... w:color="auto"/>` The foreground color for this shading pattern therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the shading color can be distinguished against the page's background color. *end example*] The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
fill (Shading Background Color)	Specifies the color used for the background for this shading. This color may either be presented as a hex value (in RRGGBB format), or auto to allow a consumer to automatically determine the background shading color as appropriate. If this attribute is omitted, then its value shall be assumed to be auto. [*Example*: Consider a shading using a background color of hex value C3D69B, using the following WordprocessingML: `<w:shd w:fill="C3D69B" />` The background color for this shading therefore will be a color with a hex value of C3D69B. *end example*] If the shading specifies the use of a theme color via the themeFill attribute, then this value is superseded by the theme color value.

Attributes	Description
	The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).

themeColor (Shading Pattern Theme Color)	Specifies a theme color which should be applied to any foreground pattern specified for this shading using the val attribute.
	
	The specified theme color is a reference to one of the predefined theme colors, located in the document's themes part, which allows for color information to be set centrally in the document.
	
	If this element is omitted, then no theme color is applied, and the color attribute shall be used to determine the shading pattern color.
	
	[*Example*: Consider a paragraph which shall have a background consisting of a theme color `accent3` with a theme color `accent6` overlaid using a 20% fill pattern. This requirement is specified using the following WordprocessingML:
	
	`<w:pPr>`
	`  <w:shd w:val="pct20" w:themeColor="accent6"`
	`w:themeFill="accent3" />`
	`</w:pPr>`
	
	The resulting paragraph will use the foreground pattern color `accent6` in the region specified by the `pct20` pattern mask. *end example*]
	
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).

themeFill (Shading Background Theme Color)	Specifies a theme color which should be applied to the background for this shading.
	
	The specified theme color is a reference to one of the predefined theme colors, located in the document's themes part, which allows for color information to be set centrally in the document.
	
	If this element is omitted, then no theme color is applied, and the color attribute shall be used to determine the shading background color.
	
	[*Example*: Consider a paragraph which shall have a background consisting of a theme color `accent3` with a theme color `accent6` overlaid using a 20% fill pattern. This requirement is specified using the following WordprocessingML:
	
	`<w:shd w:val="pct20" w:themeColor="accent6"`
	`w:themeFill="accent3" />`
	
	The resulting shading will use the background color specified by the `accent3` theme color. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).
themeFillShade (Shading Background Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this shading color. If the themeFillShade is supplied, then it is applied to the RGB value of the themeFill color (from the theme part) to determine the final color applied to this border. The themeFillShade value is stored as a hex encoding of the shade value (from 0 to 255) applied to the current border. [*Example*: Consider a shade of 40% applied to a background shading color in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255 = 102 = 66(hex)$$ The resulting themeFillShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L' = 0.53 * 0.75 = 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$ and converting back to RGB, we get 943634. This transformed value can be seen in the resulting shading's fill attribute: `<w:shd w:fill="943634" w:themeFill="accent2" w:themeFillShade="BF" />`

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeFillTint (Shading Background Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this shading instance. If the themeFillTint is supplied, then it is applied to the RGB value of the themeFill color (from the theme part) to determine the final color applied to this border. The themeFillTint value is stored as a hex encoding of the tint value (from 0 to 255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255$$ $$= 153$$ $$= 99(hex)$$ The resulting themeFillTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6)$$ $$= 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting shading's fill attribute:

Attributes	Description
	<w:top w:val="single" w:sz="4" w:space="24" w:fill="95B3D7" w:themeFillColor="accent2" w:themeFillTint="99" /> *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeShade (Shading Pattern Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this shading color. If the themeShade is supplied, then it is applied to the RGB value of the themeColor color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0 to 255) applied to the current border. [*Example*: Consider a shade of 40% applied to a background shading color in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255 = 102 = 66(hex)$$ Te resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L' = 0.53 * 0.75 = 0.39698$$

Attributes	Description
	Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:shd w:color="943634" w:themeColor="accent2" w:themeShade="BF" />` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Shading Pattern Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this shading instance. If the themeTint is supplied, then it is applied to the RGB value of the themeColor color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0 to 255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255 = 153 = 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$L' = L * \text{Tint}_{pct} + \left(1 - \text{Tint}_{pct}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get:

Attributes	Description
	$$L^{'} = 0.53 * 0.6 + (1 - .6)$$ $$= 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting shading's color attribute: `<w:shd w:color="95B3D7" w:themeColor="accent2" w:themeTint="99" />` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Shading Pattern)	Specifies the pattern which shall be used to lay the pattern color over the background color for this paragraph shading. This pattern consists of a mask which is applied over the background shading color to get the locations where the pattern color should be shown. Each of these possible masks are shown in the simple type values referenced below. [*Example*: Consider a shaded paragraph which uses a 10 percent foreground fill, resulting in the following WordprocessingML: `<w:shd w:val="pct10" .../>` This shading val is pct10, indicating that the border style is a 10 percent foreground fill mask. *end example*] The possible values for this attribute are defined by the ST_Shd simple type (§2.18.85).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Shd">
   <attribute name="val" type="ST_Shd" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="fill" type="ST_HexColor" use="optional"/>
   <attribute name="themeFill" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeFillTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeFillShade" type="ST_UcharHexNumber" use="optional"/>
</complexType>
```

2.4.34 shd (Table Shading Exception)

This element specifies the shading which shall be applied to all cells in the current row as part of a set of table-level property exceptions. Similarly to paragraph shading, this shading shall be applied to the contents of the tab up to the table borders, regardless of the presence of text - unlike cell shading, table shading shall include any cell padding. This property shall be superseded by any cell-level shading on any cell in this row (§2.4.33).

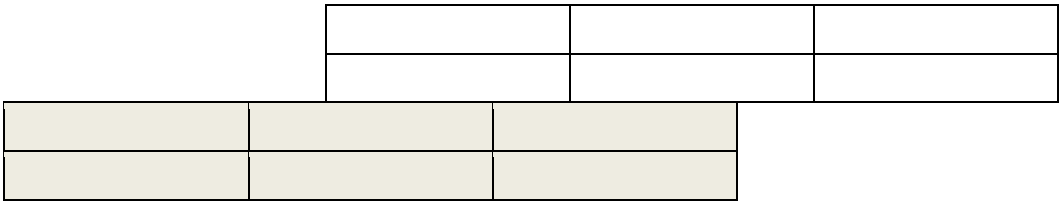
This shading consists of three components:

- Background Color
- (optional) Pattern
- (optional) Pattern Color

The resulting shading is applied by setting the background color behind the paragraph, then applying the pattern color using the mask supplied by the pattern over that background.

If this element is omitted, then the cell shading shall be determined by the table-level cell shading settings (§2.4.35) for the current table.

[*Example*: Consider a table in which the final two rows have a set of table-level property exceptions giving them theme color shading in the `background2` theme color, as follows:



This table-level shading exception would be specified using the following WordprocessingML:

```
<w:tblPrEx>
  <w:jc w:val="left" />
  <w:shd w:val="clear" w:color="auto" w:fill="EEECE1" w:themeFill="background2"
/>
</w:tblPrEx>
```

The shd element specifies cell shading with a `clear` pattern using a background theme color of `background2`. *end example*]

Parent Elements
tblPrEx (§2.4.57); tblPrEx (§2.4.58)

Attributes	Description

Attributes	Description
color (Shading Pattern Color)	Specifies the color used for any foreground pattern specified for this shading using the val attribute. This color may either be presented as a hex value (in RRGGBB format), or auto to allow a consumer to automatically determine the foreground shading color as appropriate. If the shading style (the val attribute) specifies the use of no shading format or is omitted, then this property has no effect. Also, if the shading specifies the use of a theme color via the themeColor attribute, then this value is superseded by the theme color value. If this attribute is omitted, then its value shall be assumed to be auto. [*Example*: Consider a shading of type pct20 with a foreground color value of auto, as follows: `<w:shd w:val="pct20"... w:color="auto"/>` The foreground color for this shading pattern therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the shading color can be distinguished against the page's background color. *end example*] The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
fill (Shading Background Color)	Specifies the color used for the background for this shading. This color may either be presented as a hex value (in RRGGBB format), or auto to allow a consumer to automatically determine the background shading color as appropriate. If this attribute is omitted, then its value shall be assumed to be auto. [*Example*: Consider a shading using a background color of hex value C3D69B, using the following WordprocessingML: `<w:shd w:fill="C3D69B" />` The background color for this shading therefore will be a color with a hex value of C3D69B. *end example*] If the shading specifies the use of a theme color via the themeFill attribute, then this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
themeColor	Specifies a theme color which should be applied to any foreground pattern specified for

Attributes	Description

(Shading Pattern Theme Color)	this shading using the val attribute.
	
	The specified theme color is a reference to one of the predefined theme colors, located in the document's themes part, which allows for color information to be set centrally in the document.
	
	If this element is omitted, then no theme color is applied, and the color attribute shall be used to determine the shading pattern color.
	
	[*Example*: Consider a paragraph which shall have a background consisting of a theme color `accent3` with a theme color `accent6` overlaid using a 20% fill pattern. This requirement is specified using the following WordprocessingML:
	
	`<w:pPr>`
	`  <w:shd w:val="pct20" w:themeColor="accent6"`
	`w:themeFill="accent3" />`
	`</w:pPr>`
	
	The resulting paragraph will use the foreground pattern color `accent6` in the region specified by the `pct20` pattern mask. *end example*]
	
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).

themeFill (Shading Background Theme Color)	Specifies a theme color which should be applied to the background for this shading.
	
	The specified theme color is a reference to one of the predefined theme colors, located in the document's themes part, which allows for color information to be set centrally in the document.
	
	If this element is omitted, then no theme color is applied, and the color attribute shall be used to determine the shading background color.
	
	[*Example*: Consider a paragraph which shall have a background consisting of a theme color `accent3` with a theme color `accent6` overlaid using a 20% fill pattern. This requirement is specified using the following WordprocessingML:
	
	`<w:shd w:val="pct20" w:themeColor="accent6"`
	`w:themeFill="accent3" />`
	
	The resulting shading will use the background color specified by the `accent3` theme color. *end example*]
	
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).

themeFillShade (Shading	Specifies the shade value applied to the supplied theme color (if any) for this shading color.

Attributes	Description
Background Theme Color Shade)	If the themeFillShade is supplied, then it is applied to the RGB value of the themeFill color (from the theme part) to determine the final color applied to this border. The themeFillShade value is stored as a hex encoding of the shade value (from 0 to 255) applied to the current border. [*Example*: Consider a shade of 40% applied to a background shading color in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255$$ $$= 102$$ $$= 66(hex)$$ The resulting themeFillShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L' = 0.53 * 0.75$$ $$= 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting shading's fill attribute: `<w:shd w:fill="943634" w:themeFill="accent2" w:themeFillShade="BF" />` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple

Attributes	Description
	type (§2.18.106).
themeFillTint (Shading Background Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this shading instance. If the themeFillTint is supplied, then it is applied to the RGB value of the themeFill color (from the theme part) to determine the final color applied to this border. The themeFillTint value is stored as a hex encoding of the tint value (from 0 to 255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255$$ $$= 153$$ $$= 99(hex)$$ The resulting themeFillTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6)$$ $$= 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$ and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting shading's fill attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:fill="95B3D7" w:themeFillColor="accent2" w:themeFillTint="99" />`

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeShade (Shading Pattern Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this shading color. If the themeShade is supplied, then it is applied to the RGB value of the themeColor color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0 to 255) applied to the current border. [*Example*: Consider a shade of 40% applied to a background shading color in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255$$ $$= 102$$ $$= 66(hex)$$ Te resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L' = 0.53 * 0.75$$ $$= 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute:

Attributes	Description
	<w:shd w:color="943634" w:themeColor="accent2" w:themeShade="BF" /> *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Shading Pattern Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this shading instance. If the themeTint is supplied, then it is applied to the RGB value of the themeColor color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0 to 255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255$$ $$= 153$$ $$= 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6)$$ $$= 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we

Attributes	Description
	get 95B3D7. This transformed value can be seen in the resulting shading's color attribute: `<w:shd w:color="95B3D7" w:themeColor="accent2" w:themeTint="99" />` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Shading Pattern)	Specifies the pattern which shall be used to lay the pattern color over the background color for this paragraph shading. This pattern consists of a mask which is applied over the background shading color to get the locations where the pattern color should be shown. Each of these possible masks are shown in the simple type values referenced below. [*Example*: Consider a shaded paragraph which uses a 10 percent foreground fill, resulting in the following WordprocessingML: `<w:shd w:val="pct10" .../>` This shading val is `pct10`, indicating that the border style is a 10 percent foreground fill mask. *end example*] The possible values for this attribute are defined by the ST_Shd simple type (§2.18.85).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Shd">
   <attribute name="val" type="ST_Shd" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="fill" type="ST_HexColor" use="optional"/>
   <attribute name="themeFill" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeFillTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeFillShade" type="ST_UcharHexNumber" use="optional"/>
</complexType>
```

2.4.35 shd (Table Shading)

This element specifies the shading which shall be applied to the extents the current table. Similarly to paragraph shading, this shading shall be applied to the contents of the tab up to the table borders, regardless of the presence of text - unlike cell shading, table shading shall include any cell padding. This property shall be

superseded by any cell-level shading via any table-level property exceptions (§2.4.34); or on any cell in this row (§2.4.33).

This shading consists of three components:

- Background Color
- (optional) Pattern
- (optional) Pattern Color

The resulting shading is applied by setting the background color behind the paragraph, then applying the pattern color using the mask supplied by the pattern over that background.

If this element is omitted, then the cells within this table shall have the shading specified by the associated table style. If no cell shading is specified in the style hierarchy, then the cells in this table shall not have any cell shading (i.e. they shall be transparent).

[*Example*: Consider a table in which the first cell in the first row has cell-level red shading, as follows:

R1C1	R1C2
R2C1	R2C2

This table level cell shading would be specified using the following WordprocessingML:

```
<w:tbl>
  <w:tblPr>
    <w:shd w:val="clear" w:color="auto" w:fill="FF0000"/>
    …
  </w:tblPr>
  …
</w:tbl>
```

The shd element specifies cell shading with a `clear` pattern using a background color of `FF0000` (red). *end example*]

Parent Elements
tblPr (§2.7.5.3); tblPr (§2.7.5.4); tblPr (§2.4.55); tblPr (§2.4.56)

Attributes	Description
color (Shading Pattern Color)	Specifies the color used for any foreground pattern specified for this shading using the val attribute. This color may either be presented as a hex value (in RRGGBB format), or `auto` to allow a consumer to automatically determine the foreground shading color as appropriate.

Attributes	Description
	If the shading style (the val attribute) specifies the use of no shading format or is omitted, then this property has no effect. Also, if the shading specifies the use of a theme color via the themeColor attribute, then this value is superseded by the theme color value. If this attribute is omitted, then its value shall be assumed to be auto. [*Example*: Consider a shading of type pct20 with a foreground color value of auto, as follows: `<w:shd w:val="pct20"... w:color="auto"/>` The foreground color for this shading pattern therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the shading color can be distinguished against the page's background color. *end example*] The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
fill (Shading Background Color)	Specifies the color used for the background for this shading. This color may either be presented as a hex value (in RRGGBB format), or auto to allow a consumer to automatically determine the background shading color as appropriate. If this attribute is omitted, then its value shall be assumed to be auto. [*Example*: Consider a shading using a background color of hex value C3D69B, using the following WordprocessingML: `<w:shd w:fill="C3D69B" />` The background color for this shading therefore will be a color with a hex value of C3D69B. *end example*] If the shading specifies the use of a theme color via the themeFill attribute, then this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
themeColor (Shading Pattern Theme Color)	Specifies a theme color which should be applied to any foreground pattern specified for this shading using the val attribute. The specified theme color is a reference to one of the predefined theme colors, located in the document's themes part, which allows for color information to be set centrally in the document.

Attributes	Description

	If this element is omitted, then no theme color is applied, and the color attribute shall be used to determine the shading pattern color.
	
	[*Example*: Consider a paragraph which shall have a background consisting of a theme color accent3 with a theme color accent6 overlaid using a 20% fill pattern. This requirement is specified using the following WordprocessingML:
	
	`<w:pPr>`
	`  <w:shd w:val="pct20" w:themeColor="accent6"`
	`w:themeFill="accent3" />`
	`</w:pPr>`
	
	The resulting paragraph will use the foreground pattern color accent6 in the region specified by the pct20 pattern mask. *end example*]
	
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).

themeFill (Shading Background Theme Color)	Specifies a theme color which should be applied to the background for this shading.
	
	The specified theme color is a reference to one of the predefined theme colors, located in the document's themes part, which allows for color information to be set centrally in the document.
	
	If this element is omitted, then no theme color is applied, and the color attribute shall be used to determine the shading background color.
	
	[*Example*: Consider a paragraph which shall have a background consisting of a theme color accent3 with a theme color accent6 overlaid using a 20% fill pattern. This requirement is specified using the following WordprocessingML:
	
	`<w:shd w:val="pct20" w:themeColor="accent6"`
	`w:themeFill="accent3" />`
	
	The resulting shading will use the background color specified by the accent3 theme color. *end example*]
	
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).

themeFillShade (Shading Background Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this shading color. If the themeFillShade is supplied, then it is applied to the RGB value of the themeFill color (from the theme part) to determine the final color applied to this border. The themeFillShade value is stored as a hex encoding of the shade value (from 0 to 255)

Attributes	Description
	applied to the current border. [*Example*: Consider a shade of 40% applied to a background shading color in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255 = 102 = 66(hex)$$ The resulting themeFillShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$L' = L * Shade_{percentage}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L' = 0.53 * 0.75 = 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$ and converting back to RGB, we get 943634. This transformed value can be seen in the resulting shading's fill attribute: `<w:shd w:fill="943634" w:themeFill="accent2" w:themeFillShade="BF" />` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeFillTint (Shading Background Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this shading instance. If the themeFillTint is supplied, then it is applied to the RGB value of the themeFill color

Attributes	Description
	(from the theme part) to determine the final color applied to this border. The themeFillTint value is stored as a hex encoding of the tint value (from 0 to 255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255 = 153 = 99(hex)$$ The resulting themeFillTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6) = 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting shading's fill attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:fill="95B3D7" w:themeFillColor="accent2" w:themeFillTint="99" />` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).

Attributes	Description
themeShade (Shading Pattern Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this shading color. If the themeShade is supplied, then it is applied to the RGB value of the themeColor color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0 to 255) applied to the current border. [*Example*: Consider a shade of 40% applied to a background shading color in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255$$ $$= 102$$ $$= 66(hex)$$ Te resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L' = 0.53 * 0.75$$ $$= 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:shd w:color="943634" w:themeColor="accent2" w:themeShade="BF" />` *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Shading Pattern Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this shading instance. If the themeTint is supplied, then it is applied to the RGB value of the themeColor color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0 to 255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255$$ $$= 153$$ $$= 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6)$$ $$= 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting shading's color attribute: `<w:shd w:color="95B3D7" w:themeColor="accent2"`

Attributes	Description
	`w:themeTint="99" />` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Shading Pattern)	Specifies the pattern which shall be used to lay the pattern color over the background color for this paragraph shading. This pattern consists of a mask which is applied over the background shading color to get the locations where the pattern color should be shown. Each of these possible masks are shown in the simple type values referenced below. [*Example*: Consider a shaded paragraph which uses a 10 percent foreground fill, resulting in the following WordprocessingML: `<w:shd w:val="pct10" .../>` This shading val is `pct10`, indicating that the border style is a 10 percent foreground fill mask. *end example*] The possible values for this attribute are defined by the ST_Shd simple type (§2.18.85).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Shd">
   <attribute name="val" type="ST_Shd" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="fill" type="ST_HexColor" use="optional"/>
   <attribute name="themeFill" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeFillTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeFillShade" type="ST_UcharHexNumber" use="optional"/>
</complexType>
```

2.4.36 tbl (Table)

This element specifies the contents of a table present in the document. A *table* is a set of paragraphs (and other block-level content) arranged in *rows* and *columns*. Tables in WordprocessingML are defined via the tbl element, which is analogous to the HTML `table` tag.

[*Example*: Consider an empty one-cell table (i.e.; a table with one row, one column) and 1 point borders on all sides:

This table is represented by the following WordprocessingML:

```
<w:tbl>
  <w:tblPr>
    <w:tblW w:w="5000" w:type="pct"/>
    <w:tblBorders>
      <w:top w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:left w:val="single" w:sz="4 w:space="0" w:color="auto"/>
      <w:bottom w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:right w:val="single" w:sz="4" w:space="0" w:color="auto"/>
    </w:tblBorders>
  </w:tblPr>
  <w:tblGrid>
    <w:gridCol w:w="10296"/>
  </w:tblGrid>
  <w:tr>
    <w:tc>
      <w:tcPr>
        <w:tcW w:w="0" w:type="auto"/>
      </w:tcPr>
      <w:p/>
    </w:tc>
  </w:tr>
</w:tbl>
```

This table specifies table-wide properties of 100% of page width using the tblW element (§2.4.61); a the set of table borders using the tblBorders element (§2.4.38); the table grid which defines a set of shared vertical edges within the table using the tblGrid element (§2.4.44); and a single table row using the tr element (§2.4.75). *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.6); docPartBody (§2.12.6); endnote (§2.11.2); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); sdtContent (§2.5.2.32); tc (§2.4.62); txbxContent (§2.17.1.1)

Child Elements	Subclause
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Row-Level Custom XML Element)	§2.5.1.4
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4

Child Elements	Subclause
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
del (Deleted Run Content)	§2.13.5.12
ins (Inserted Run Content)	§2.13.5.20
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
sdt (Row-Level Structured Document Tag)	§2.5.2.31
tblGrid (Table Grid)	§2.4.44
tblPr (Table Properties)	§2.4.55
tr (Table Row)	§2.4.75

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Tbl">
    <sequence>
        <group ref="EG_RangeMarkupElements" minOccurs="0" maxOccurs="unbounded"/>
        <element name="tblPr" type="CT_TblPr"/>
        <element name="tblGrid" type="CT_TblGrid"/>
        <group ref="EG_ContentRowContent" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
</complexType>
```

2.4.37 tblBorders (Table Borders Exceptions)

This element specifies the set of borders for the edges of the parent table row via a set of table-level property exceptions, using the six border types defined by its child elements.

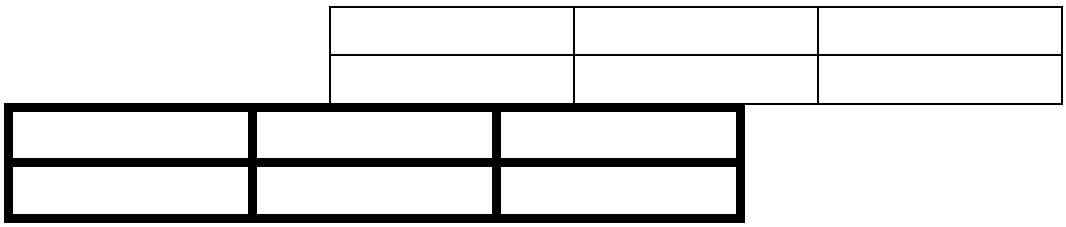
If the cell spacing for any row is non-zero as specified using the tblCellSpacing element (§2.4.41; §2.4.42; §2.4.43), then there is no border conflict and the table-level exception border shall be displayed.

If the cell spacing is zero, then there is a conflict [*Example*: Between the left border of all cells in the first column and the left border of the table-level exceptions. *end example*], which shall be resolved as follows:

- If there is a cell border, then the cell border shall be displayed
- If there is no cell border, then the table-level exception border shall be displayed

If this element is omitted, then this table shall have the borders specified by the associated table level borders (§2.4.38).

[*Example*: Consider a table in which the final two rows have a set of table-level property exceptions giving them a thicker set of table borders, as follows:



These table borders are specified via a set of table-level property exceptions using the following WordprocessingML:

```
<w:tr>
  <w:tblPrEx>
    <w:tblBorders>
      <w:top w:val="single" w:sz="24" w:space="0" w:color="000000"
w:themeColor="text1"/>
      <w:left w:val="single" w:sz="24" w:space="0" w:color="000000"
w:themeColor="text1"/>
      <w:bottom w:val="single" w:sz="24" w:space="0" w:color="000000"
w:themeColor="text1"/>
      <w:right w:val="single" w:sz="24" w:space="0" w:color="000000"
w:themeColor="text1"/>
      <w:insideH w:val="single" w:sz="24" w:space="0" w:color="000000"
w:themeColor="text1"/>
      <w:insideV w:val="single" w:sz="24" w:space="0" w:color="000000"
w:themeColor="text1"/>
    </w:tblBorders>
  </w:tblPrEx>
</w:tr>
```

The tblBorders element specifies the set of table borders applied to the final two rows in this table as part of the table-level property exceptions. *end example*]

Parent Elements
tblPrEx (§2.4.57); tblPrEx (§2.4.58)

Child Elements	Subclause
bottom (Table Bottom Border)	§2.4.4
insideH (Table Inside Horizontal Edges Border)	§2.4.17
insideV (Table Inside Vertical Edges Border)	§2.4.20
left (Table Left Border)	§2.4.27
right (Table Right Border)	§2.4.32
top (Table Top Border)	§2.4.71

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblBorders">
   <sequence>
      <element name="top" type="CT_Border" minOccurs="0"/>
      <element name="left" type="CT_Border" minOccurs="0"/>
      <element name="bottom" type="CT_Border" minOccurs="0"/>
      <element name="right" type="CT_Border" minOccurs="0"/>
      <element name="insideH" type="CT_Border" minOccurs="0"/>
      <element name="insideV" type="CT_Border" minOccurs="0"/>
   </sequence>
</complexType>
```

2.4.38 tblBorders (Table Borders)

This element specifies the set of borders for the edges of the current table, using the six border types defined by its child elements.

If the cell spacing for any row is non-zero as specified using the tblCellSpacing element (§2.4.41; §2.4.42; §2.4.43), then there is no border conflict and the table border (or table-level exception border, if one is specified) shall be displayed.

If the cell spacing is zero, then there is a conflict [*Example*: Between the left border of all cells in the first column and the left border of the table. *end example*], which shall be resolved as follows:

- If there is a cell border, then the cell border shall be displayed
- If there is no cell border but there is a table-level exception border on this table row, then the table-level exception border shall be displayed
- If there is no cell or table-level exception border, then the table border shall be displayed

If this element is omitted, then this table shall have the borders specified by the associated table style. If no borders are specified in the style hierarchy, then this table shall not have any table borders.

[*Example*: Consider a table with no associated table style, which defines a set of table borders via direct formatting as follows:

These table borders are specified using the following WordprocessingML:

```
<w:tbl>
  <w:tblPr>
    <w:tblW w:w="0" w:type="auto"/>
    <w:tblBorders>
      <w:top w:val="single" w:sz="4" w:space="0" w:color="000000"
w:themeColor="text1"/>
      <w:left w:val="single" w:sz="4" w:space="0" w:color="000000"
w:themeColor="text1"/>
      <w:bottom w:val="single" w:sz="4" w:space="0" w:color="000000"
w:themeColor="text1"/>
      <w:right w:val="single" w:sz="4" w:space="0" w:color="000000"
w:themeColor="text1"/>
      <w:insideH w:val="single" w:sz="4" w:space="0" w:color="000000"
w:themeColor="text1"/>
      <w:insideV w:val="single" w:sz="4" w:space="0" w:color="000000"
w:themeColor="text1"/>
    </w:tblBorders>
    …
  </w:tblPr>
  …
</w:tbl>
```

The tblBorders element specifies the set of table borders applied to the current table. *end example*]

Parent Elements
tblPr (§2.7.5.3); tblPr (§2.7.5.4); tblPr (§2.4.55); tblPr (§2.4.56)

Child Elements	Subclause

Child Elements	Subclause
bottom (Table Bottom Border)	§2.4.4
insideH (Table Inside Horizontal Edges Border)	§2.4.17
insideV (Table Inside Vertical Edges Border)	§2.4.20
left (Table Left Border)	§2.4.27
right (Table Right Border)	§2.4.32
top (Table Top Border)	§2.4.71

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblBorders">
   <sequence>
      <element name="top" type="CT_Border" minOccurs="0"/>
      <element name="left" type="CT_Border" minOccurs="0"/>
      <element name="bottom" type="CT_Border" minOccurs="0"/>
      <element name="right" type="CT_Border" minOccurs="0"/>
      <element name="insideH" type="CT_Border" minOccurs="0"/>
      <element name="insideV" type="CT_Border" minOccurs="0"/>
   </sequence>
</complexType>
```

2.4.39 tblCellMar (Table Cell Margin Defaults)

This element specifies the default cell margin settings for all cells in the current table. These setting may be overridden by the table cell margin definition specified by the tcMar element contained within the table cell's properties (§2.4.65) or by a set of table-level property exceptions (§2.4.40).

If this element is omitted, then it shall inherit the table cell margins from the associated table style. If table margins are never specified in the style hierarchy, then each margin shall use its default margin size (see child element definitions).

[*Example*: Consider a table defined to have default cells margins of 0.1 inches for all sides, as follows:

R1C1	R1C2
R2C1	R2C2

This set of default table cell margins would be specified using the following WordprocessingML:

```
<w:tblPr>
  <w:tblCellMar>
    <w:top w:w="144" w:type="dxa"/>
    <w:left w:w="144" w:type="dxa"/>
    <w:bottom w:w="144" w:type="dxa"/>
    <w:right w:w="144" w:type="dxa"/>
  </w:tblCellMar>
  …
</w:tblPr>
```

The tblCellMar element as a child of tblPr specifies the set of default cell margins for all cells in the current table, in this case, 144 twentieths of a point. *end example*]

Parent Elements
tblPr (§2.7.5.3); tblPr (§2.7.5.4); tblPr (§2.4.55); tblPr (§2.4.56)

Child Elements	Subclause
bottom (Table Cell Bottom Margin Default)	§2.4.5
left (Table Cell Left Margin Default)	§2.4.26
right (Table Cell Right Margin Default)	§2.4.29
top (Table Cell Top Margin Default)	§2.4.72

The following XML Schema fragment defines the contents of this element:

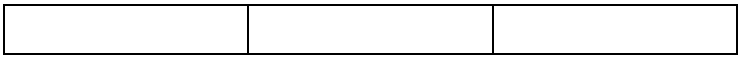
```
<complexType name="CT_TblCellMar">
   <sequence>
      <element name="top" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="left" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="bottom" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="right" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

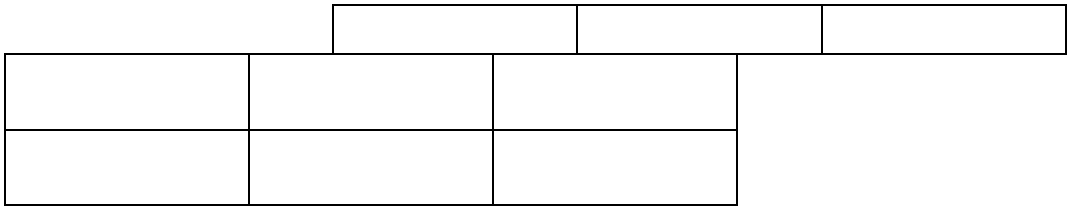
2.4.40 tblCellMar (Table Cell Margin Exceptions)

This element specifies a set of cell margins for all cells in the parent table row via a set of table-level property exceptions. These settings may be overridden by the table cell margin definition specified by the tcMar element contained within the table cell's properties (§2.4.40).

If this element is omitted, then it shall inherit the table cell margins from the table-level cell margins (§2.4.39).

[*Example*: Consider a table whose final two rows are defined to have default cell margins of 0.1 inches for all sides via a table-level property exception, as follows:





This set of table cell margin exceptions is specified using the following WordprocessingML:

```
<w:tblPrEx>
  <w:tblCellMar>
    <w:top w:w="144" w:type="dxa"/>
    <w:left w:w="144" w:type="dxa"/>
    <w:bottom w:w="144" w:type="dxa"/>
    <w:right w:w="144" w:type="dxa"/>
  </w:tblCellMar>
  …
</w:tblPrEx>
```

The tblCellMar element as a child of tblPrEx specifies the set of default cell margins for all cells in final two rows in current table, in this case, 144 twentieths of a point on all sides. *end example*]

Parent Elements
tblPrEx (§2.4.57); tblPrEx (§2.4.58)

Child Elements	Subclause
bottom (Table Cell Bottom Margin Default)	§2.4.5
left (Table Cell Left Margin Default)	§2.4.26
right (Table Cell Right Margin Default)	§2.4.29
top (Table Cell Top Margin Default)	§2.4.72

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblCellMar">
   <sequence>
      <element name="top" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="left" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="bottom" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="right" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

2.4.41 tblCellSpacing (Table Cell Spacing Exception)

This element specifies a table cell spacing exception for all cells in the parent table row as part of a set of table-level property exceptions. If specified, this element specifies the minimum amount of space which shall be left between all cells in the parent row after including the width of the table borders in the calculation. This setting shall be superseded by the row cell spacing value (§2.4.42). It is important to note that table-level cell spacing shall be added outside of the text margins, which shall be aligned with the innermost starting edge of the text extents in a table cell.

This value is specified in the units applied via its type attribute. Any width value of type `pct` or `auto` for this element shall be ignored.

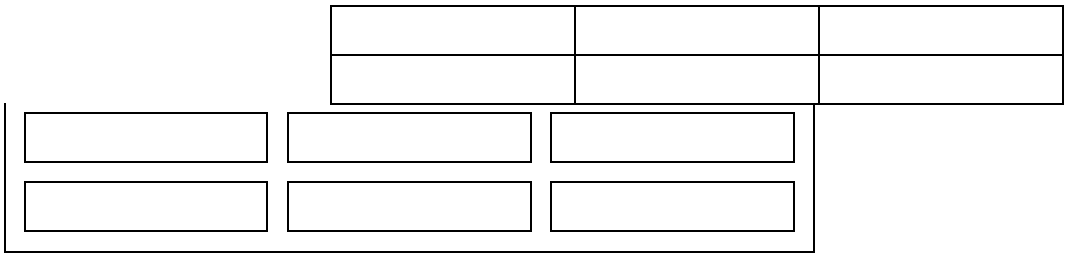
[*Example*: Consider a table whose first cell has a six point wide table border, and a table cell spacing value of 0.01 inches. The resulting table would have 0.01 inches of space between each table cell regardless of the width of the cell border, as follows (notice that no border is covered by any other border):

R1C1	R1C2
R2C1	R2C2

end example]

If this element is omitted, then the row shall inherit the table cell spacing from the table-level cell spacing setting (§2.4.39), excepting the case of a row level override.

[*Example*: Consider a table whose final two rows are defined to have cell spacing of 0.1 inches for all sides via a table-level property exception, as follows:



This table cell spacing exception is specified using the following WordprocessingML:

```
<w:tblPrEx>
  <w:tblCellSpacing w:w="144" w:type="dxa">
  …
</w:tblPrEx>
```

The tblCellSpacing element as a child of tblPrEx specifies the default cell spacing between all cells in final two rows in the current table, in this case 144 twentieths of a point. *end example*]

Parent Elements
tblPrEx (§2.4.57); tblPrEx (§2.4.58)

Attributes	Description
type (Table Width Type)	Specifies the units of the width property being defined by the parent element's w attribute. This property is used to define various properties of a table, including: cell spacing, preferred width, and table margins. If this attribute is omitted, then its value shall be assumed to be dxa (twentieths of a point). [*Example*: Consider a table with a table cell bottom cell spacing with a type of dxa, as follows: `<w:bottom ... w:type="dxa" />` This type shall therefore be used to interpret the width specified in the w attribute as a value in twentieths of a point. *end example*] The possible values for this attribute are defined by the ST_TblWidth simple type (§2.18.97).
w (Table Width Value)	Specifies the value of the width property being defined by the parent element. This property is used to define various properties of a table, including: cell spacing, preferred widths, and table margins. If this attribute is omitted, then its value shall be assumed to be 0. [*Example*: Consider a table with a bottom margin with a width of 302, as follows: `<w:bottom w:w="302" w:type="dxa" />` The value in the w attribute shall therefore be used to determine the width being specified in the context of the units specified in the type attribute. In this case, the type is twentieths of a point (dxa), so the width is 302 twentieths of a point (.2097 inches). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblWidth">
   <attribute name="w" type="ST_DecimalNumber"/>
   <attribute name="type" type="ST_TblWidth"/>
</complexType>
```

2.4.42 tblCellSpacing (Table Row Cell Spacing)

This element specifies the default table cell spacing (the spacing between adjacent cells and the edges of the table) for all cells in the parent row. If specified, this element specifies the minimum amount of space which shall be left between all cells in the table including the width of the table borders in the calculation. It is important to note that row-level cell spacing shall be added inside of the text margins, which shall be aligned with the innermost starting edge of the text extents in a cell without row-level indentation or cell spacing. Row-level cell spacing shall not increase the width of the overall table.

This value is specified in the units applied via its type attribute. Any width value of type `pct` or `auto` for this element shall be ignored.

[*Example*: Consider a table whose first cell has a six point wide table border, and a table cell spacing value of 0.01 inches. The resulting table would have 0.01 inches of space between each table cell regardless of the width of the cell border, as follows (notice that no border is covered by any other border):

R1C1	R1C2
R2C1	R2C2

end example]

If this element is omitted, then the cells in this row shall inherit the cell spacing from the associated table level properties.

[*Example*: Consider a table where the second row has a cell spacing of 0.1 inches for all sides specified via the table row properties as follows:

This table row cell spacing is specified using the following WordprocessingML:

```
<w:trPr>
  <w:tblCellSpacing w:w="144" w:type="dxa">
  …
</w:trPr>
```

The tblCellSpacing element as a child of trPr specifies the default cell spacing between all cells in the current row, in this case 144 twentieths of a point. *end example*]

Parent Elements
trPr (§2.7.5.10); trPr (§2.7.5.11); trPr (§2.4.78); trPr (§2.4.79)

Attributes	Description
type (Table Width Type)	Specifies the units of the width property being defined by the parent element's w attribute. This property is used to define various properties of a table, including: cell spacing, preferred width, and table margins. If this attribute is omitted, then its value shall be assumed to be dxa (twentieths of a point). [*Example*: Consider a table with a table cell bottom cell spacing with a type of dxa, as follows: `<w:bottom ... w:type="dxa" />` This type shall therefore be used to interpret the width specified in the w attribute as a value in twentieths of a point. *end example*] The possible values for this attribute are defined by the ST_TblWidth simple type (§2.18.97).
w (Table Width Value)	Specifies the value of the width property being defined by the parent element. This property is used to define various properties of a table, including: cell spacing, preferred widths, and table margins. If this attribute is omitted, then its value shall be assumed to be 0. [*Example*: Consider a table with a bottom margin with a width of 302, as follows: `<w:bottom w:w="302" w:type="dxa" />` The value in the w attribute shall therefore be used to determine the width being specified in the context of the units specified in the type attribute. In this case, the type is twentieths of a point (dxa), so the width is 302 twentieths of a point (.2097 inches). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type

Attributes	Description
	(§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblWidth">
   <attribute name="w" type="ST_DecimalNumber"/>
   <attribute name="type" type="ST_TblWidth"/>
</complexType>
```

2.4.43 tblCellSpacing (Table Cell Spacing Default)

This element specifies the default table cell spacing (the spacing between adjacent cells and the edges of the table) for all cells in the parent table. If specified, this element specifies the minimum amount of space which shall be left between all cells in the table including the width of the table borders in the calculation. This setting shall be superseded by a table-level exception (§2.4.41) or the row cell spacing value (§2.4.42) in that order. It is important to note that table-level cell spacing shall be added outside of the text margins, which shall be aligned with the innermost starting edge of the text extents in a table cell.

This value is specified in the units applied via its type attribute. Any width value of type `pct` or `auto` for this element shall be ignored.

[*Example*: Consider a table whose first cell has a six point wide table border, and a table cell spacing value of 0.01 inches. The resulting table would have 0.01 inches of space between each table cell regardless of the width of the cell border, as follows (notice that no border is covered by any other border):

R1C1	R1C2
R2C1	R2C2

end example]

If this element is omitted, then the table shall inherit the table cell spacing from the associated table style. If table cell spacing is never specified in the style hierarchy, no cell spacing shall be added to the parent table.

[*Example*: Consider a table with a default cell spacing of 0.1 inches for all sides as follows:

R1C1	R1C2
R2C1	R2C2

This table cell spacing deafult is specified using the following WordprocessingML:

```
<w:tblPr>
  <w:tblCellSpacing w:w="144" w:type="dxa">
  …
</w:tblPr>
```

The tblCellSpacing element as a child of tblPr specifies the default cell spacing between all cells in the current table, in this case 144 twentieths of a point. *end example*]

Parent Elements
tblPr (§2.7.5.3); tblPr (§2.7.5.4); tblPr (§2.4.55); tblPr (§2.4.56)

Attributes	Description
type (Table Width Type)	Specifies the units of the width property being defined by the parent element's w attribute. This property is used to define various properties of a table, including: cell spacing, preferred width, and table margins. If this attribute is omitted, then its value shall be assumed to be dxa (twentieths of a point). [*Example*: Consider a table with a table cell bottom cell spacing with a type of dxa, as follows: `<w:bottom ... w:type="dxa" />` This type shall therefore be used to interpret the width specified in the w attribute as a value in twentieths of a point. *end example*] The possible values for this attribute are defined by the ST_TblWidth simple type (§2.18.97).
w (Table Width Value)	Specifies the value of the width property being defined by the parent element. This property is used to define various properties of a table, including: cell spacing, preferred widths, and table margins. If this attribute is omitted, then its value shall be assumed to be 0. [*Example*: Consider a table with a bottom margin with a width of 302, as follows: `<w:bottom w:w="302" w:type="dxa" />` The value in the w attribute shall therefore be used to determine the width being specified in the context of the units specified in the type attribute. In this case, the type is twentieths of a point (dxa), so the width is 302 twentieths of a point (.2097 inches). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type

Attributes	Description
	(§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblWidth">
   <attribute name="w" type="ST_DecimalNumber"/>
   <attribute name="type" type="ST_TblWidth"/>
</complexType>
```

2.4.44 tblGrid (Table Grid)

This element specifies the table grid for the current table. The *table grid* is a definition of the set of grid columns which define all of the shared vertical edges of the table, as well as default widths for each of these grid columns. These grid column widths are then used to determine the size of the table based on the table layout algorithm used (§2.4.49;§2.4.50).

If the table grid is omitted, then a new grid shall be constructed from the actual contents of the table assuming that all grid columns have a width of `0`.

[*Example*: Consider the following table with four vertical edges (grid columns):

This table would have a table grid consisting of four grid columns as follows:

```
<w:tblGrid>
  <w:gridCol w:w="2088"/>
  <w:gridCol w:w="1104"/>
  <w:gridCol w:w="3192"/>
  <w:gridCol w:w="3192"/>
</w:tblGrid>
```

The tblGrid element contains the current definition for the table grid, consisting of all for grid columns as well as default widths for those columns. *end example*]

Parent Elements
tbl (§2.4.36)

Child Elements	Subclause
gridCol (Grid Column Definition)	§2.4.12

Child Elements	Subclause
tblGridChange (Revision Information for Table Grid Column Definitions)	§2.13.5.35

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblGrid">
   <complexContent>
      <extension base="CT_TblGridBase">
         <sequence>
            <element name="tblGridChange" type="CT_TblGridChange" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.4.45 tblGrid (Previous Table Grid)

This element specifies a previous table grid state, the modifications to which shall be attributed to a revision by a particular author and at a particular time. This element contains the table grid settings which were previously in place before a specific set of revisions by one author. The *table grid* is a definition of the set of grid columns which define all of the shared vertical edges of the table, as well as default widths for each of these grid columns. These grid column widths are then used to determine the size of the table based on the table layout algorithm used (§2.4.49;§2.4.50).

[*Example*: Consider the following table with four vertical edges (grid columns):

If we now modify this table by reducing the size of the last column without changing the overall table width, as follows:

This table would have a table grid consisting of four grid columns as follows:

```
<w:tblGrid>
  <w:gridCol w:w="2088"/>
  <w:gridCol w:w="1104"/>
  <w:gridCol w:w="3583"/>
  <w:gridCol w:w="2801"/>
  <w:tblGridChange w:id="1">
    <w:tblGrid>
      <w:gridCol w:w="2088"/>
      <w:gridCol w:w="1104"/>
      <w:gridCol w:w="3192"/>
      <w:gridCol w:w="3192"/>
    </w:tblGrid>
  </w:tblGridChange>
</w:tblGrid>
```

The tblGrid element as a child of tblGridChange contains the previous definition for the table grid, consisting of all for grid columns as well as the original widths for those columns. *end example*]

Parent Elements
tblGridChange (§2.13.5.35)

Child Elements	Subclause
gridCol (Grid Column Definition)	§2.4.12

The following XML Schema fragment defines the contents of this element:

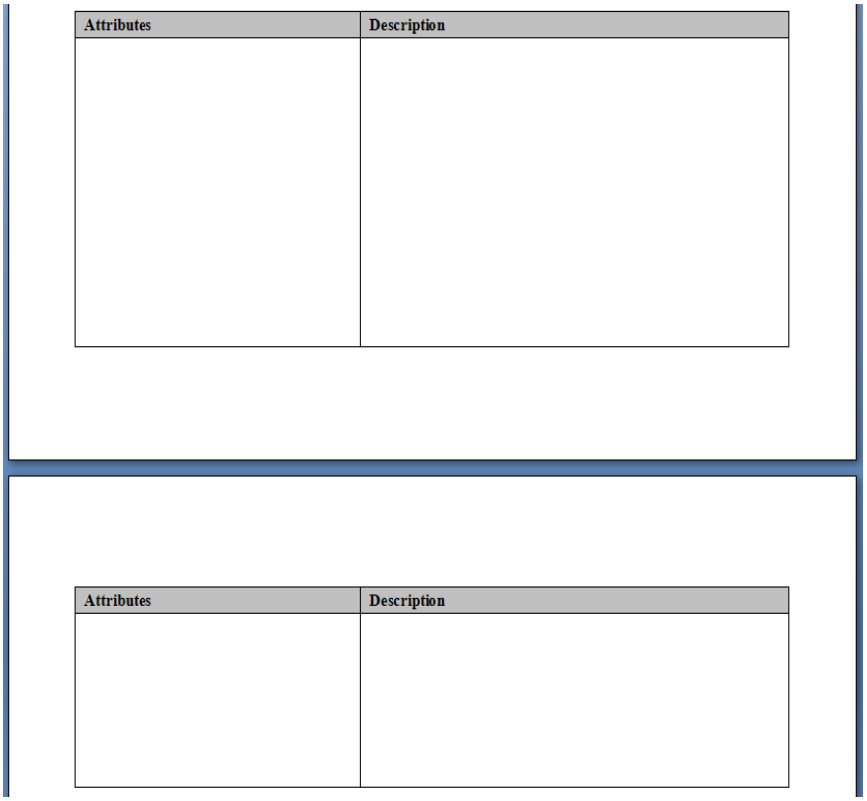
```
<complexType name="CT_TblGridBase">
   <sequence>
      <element name="gridCol" type="CT_TblGridCol" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

2.4.46 tblHeader (Repeat Table Row on Every New Page)

This element specifies that the current table row shall be repeated at the top of each new page on which part of this table is displayed. This gives this table row the behavior of a 'header' row on each of these pages.

If this element is omitted, then the table shall inherit the table cell spacing from the associated table style. If table cell spacing is never specified in the style hierarchy, then this table row shall not be repeated on each new page on which the table is displayed. As well, if this row is not contiguously connected with the first row of the table (that is, if this table row is not either the first row, or all rows between this row and the first row are not marked as header rows) then this property shall be ignored.

[*Example*: Consider a table which shall have its first row repeated on each new page, like the attribute listings in this Office Open XML Standard, for example:



Notice that the first row in the table is repeated on the top of the second page. This requirement would be specified as follows in the WordprocessingML for that row:

```
<w:trPr>
  <w:tblHeader />
</w:trPr>
```

The tblHeader element specifies that this table row will now be repeated as a header row at the top of each page. *end example*]

Parent Elements
trPr (§2.7.5.10); trPr (§2.7.5.11); trPr (§2.4.78); trPr (§2.4.79)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off.

Attributes	Description
	[*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.4.47 tblInd (Table Indent from Leading Margin Exception)

This element specifies the indentation which shall be added before the leading edge of the set of parent table rows which have this set of table-level property exceptions applied. This indentation should shift the table into the text margin by the specified amount in the document (the left edge in a left-to-right table, and the right edge in a right-to-left table).

This value is specified in the units applied via its type attribute. Any width value of type pct or auto for this element shall be ignored.

If this element is omitted, then the table shall inherit the table indentation from the associated table level property setting. If the resulting justification on the parent table row is not left after applying the value of the jc element from the three levels of this property (§2.4.21;§2.4.22;§2.4.23), then this property shall be ignored.

[*Example*: Consider a table in which the last two rows shall be indented one inch from the left margin via a table-level property exception definition, as follows:

This setting would be specified using the following WordprocessingML:

```
<w:tblPrEx>
  <w:tblInd w:val="1440" w:type="dxa"/>
</w:tblPrEx>
```

The tblInd element as a child of tblPrEx specifies that the rows with the table-level property exception shall be indented by 1440 twentieths of a point (one inch). *end example*]

Parent Elements
tblPrEx (§2.4.57); tblPrEx (§2.4.58)

Attributes	Description
type (Table Width Type)	Specifies the units of the width property being defined by the parent element's w attribute. This property is used to define various properties of a table, including: cell spacing, preferred width, and table margins. If this attribute is omitted, then its value shall be assumed to be dxa (twentieths of a point). [*Example*: Consider a table with a table cell bottom cell spacing with a type of dxa, as follows: `<w:bottom ... w:type="dxa" />` This type shall therefore be used to interpret the width specified in the w attribute as a value in twentieths of a point. *end example*] The possible values for this attribute are defined by the ST_TblWidth simple type (§2.18.97).
w (Table Width Value)	Specifies the value of the width property being defined by the parent element. This property is used to define various properties of a table, including: cell spacing, preferred widths, and table margins. If this attribute is omitted, then its value shall be assumed to be 0. [*Example*: Consider a table with a bottom margin with a width of 302, as follows: `<w:bottom w:w="302" w:type="dxa" />` The value in the w attribute shall therefore be used to determine the width being specified in the context of the units specified in the type attribute. In this case, the type is twentieths of a point (dxa), so the width is 302 twentieths of a point (.2097 inches). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblWidth">
   <attribute name="w" type="ST_DecimalNumber"/>
   <attribute name="type" type="ST_TblWidth"/>
</complexType>
```

2.4.48 tblInd (Table Indent from Leading Margin)

This element specifies the indentation which shall be added before the leading edge of the current table in the document (the left edge in a left-to-right table, and the right edge in a right-to-left table). This indentation should shift the table into the text margin by the specified amount.

This value is specified in the units applied via its type attribute. Any width value of type `pct` or `auto` for this element shall be ignored.

If this element is omitted, then the table shall inherit the table indentation from the associated table style. If table indentation is never specified in the style hierarchy, no indentation shall be added to the parent table. If the resulting justification on any table row is not `left` after applying the value of the jc element from the three levels of this property (§2.4.21;§2.4.22;§2.4.23), then this property shall be ignored.

[*Example*: Consider a table which shall be indented one inch from the left margin, as follows:

R1C1	R1C2
R2C1	R2C2

This setting would be specified using the following WordprocessingML:

```
<w:tblPr>
  <w:jc w:val="left"/>
  <w:tblInd w:val="1440" w:type="dxa"/>
</w:tblPr>
```

If the properties on this table were now modified to justify it on the right side by setting the value of the jc element to `right`, as follows:

```
<w:tblPr>
  <w:jc w:val="right"/>
  <w:tblInd w:val="1440" w:type="dxa"/>
</w:tblPr>
```

This table would now have no indent, as the justification is no longer on the leading edge (`left`):

R1C1	R1C2
R2C1	R2C2

end example]

Parent Elements
tblPr (§2.7.5.3); tblPr (§2.7.5.4); tblPr (§2.4.55); tblPr (§2.4.56)

Attributes	Description
type (Table Width Type)	Specifies the units of the width property being defined by the parent element's w attribute. This property is used to define various properties of a table, including: cell spacing, preferred width, and table margins. If this attribute is omitted, then its value shall be assumed to be dxa (twentieths of a point). [*Example*: Consider a table with a table cell bottom cell spacing with a type of dxa, as follows: `<w:bottom ... w:type="dxa" />` This type shall therefore be used to interpret the width specified in the w attribute as a value in twentieths of a point. *end example*] The possible values for this attribute are defined by the ST_TblWidth simple type (§2.18.97).
w (Table Width Value)	Specifies the value of the width property being defined by the parent element. This property is used to define various properties of a table, including: cell spacing, preferred widths, and table margins. If this attribute is omitted, then its value shall be assumed to be 0. [*Example*: Consider a table with a bottom margin with a width of 302, as follows: `<w:bottom w:w="302" w:type="dxa" />` The value in the w attribute shall therefore be used to determine the width being specified in the context of the units specified in the type attribute. In this case, the type is twentieths of a point (dxa), so the width is 302 twentieths of a point (.2097 inches). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblWidth">
   <attribute name="w" type="ST_DecimalNumber"/>
   <attribute name="type" type="ST_TblWidth"/>
</complexType>
```

2.4.49 tblLayout (Table Layout)

This element specifies the algorithm which shall be used to lay out the contents of this table within the document. When a table is displayed in a document, it can either be displayed using a fixed width or autofit layout algorithm (each discussed in the simple type referenced by the val attribute).

If this element is omitted, then the value of this element shall be assumed to be auto.

[*Example*: Consider a table which shall use the fixed width table layout algorithm. This requirement is specified using the following WordprocessingML:

```
<w:tblPr>
  <w:tblLayout w:type="fixed"/>
</w:tblPr>
```

The tblLayout element specifies that the table shall use the fixed layout algorithm. *end example*]

Parent Elements
tblPr (§2.7.5.3); tblPr (§2.7.5.4); tblPr (§2.4.55); tblPr (§2.4.56)

Attributes	Description

type (Table Layout Setting)	Specifies the algorithm which shall be used to lay out the contents of the parent table (see simple type definition for details on each algorithm used).
	
	[*Example*: Consider a table which shall use the AutoFit width table layout algorithm. This requirement is specified using the following WordprocessingML:
	
	`<w:tblPr>`
	`  <w:tblLayout w:type="auto"/>`
	`</w:tblPr>`
	
	The tblLayout element specifies that the table shall use the auto layout algorithm. *end example*]
	
	The possible values for this attribute are defined by the ST_TblLayoutType simple type (§2.18.94).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblLayoutType">
   <attribute name="type" type="ST_TblLayoutType"/>
</complexType>
```

2.4.50 tblLayout (Table Layout Exception)

This element specifies the algorithm which shall be used to lay out the contents of all rows with this table within the table which have the set of table-level property exceptions specified by the parent element. When a table is displayed in a document, it can either be displayed using a fixed width or autofit layout algorithm (each discussed in the simple type referenced by the val attribute).

If this element is omitted, then the value of this element shall be assumed to be auto.

[*Example*: Consider a table which shall use the fixed width table layout algorithm. This requirement is specified using the following WordprocessingML:

```
<w:tblPrEx>
  <w:tblLayout w:type="fixed"/>
</w:tblPrEx>
```

The tblLayout element specifies that the table shall use the `fixed` layout algorithm. *end example*]

Parent Elements
tblPrEx (§2.4.57); tblPrEx (§2.4.58)

Attributes	Description

type (Table Layout Setting)	Specifies the algorithm which shall be used to lay out the contents of the parent table (see simple type definition for details on each algorithm used).
	
	[*Example*: Consider a table which shall use the AutoFit width table layout algorithm. This requirement is specified using the following WordprocessingML:
	
	`<w:tblPr>`
	`  <w:tblLayout w:type="auto"/>`
	`</w:tblPr>`
	
	The tblLayout element specifies that the table shall use the `auto` layout algorithm. *end example*]
	
	The possible values for this attribute are defined by the ST_TblLayoutType simple type (§2.18.94).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblLayoutType">
   <attribute name="type" type="ST_TblLayoutType"/>
</complexType>
```

2.4.51 tblLook (Table Style Conditional Formatting Settings)

This element specifies the components of the conditional formatting of the referenced table style (if one exists) which shall be applied to the current table. A table style can specify up to six different optional conditional formats [*Example*: Different formatting for first column. *end example*], which then can be applied or omitted from individual tables in the document.

This element's value is hexadecimal code containing a bitmask of options, interpreted as follows:

- 0x0020=Apply first row conditional formatting
- 0x0040=Apply last row conditional formatting

- 0x0080=Apply first column conditional formatting
- 0x0100=Apply last column conditional formatting
- 0x0200=Do not apply row banding conditional formatting
- 0x0400=Do not apply column banding conditional formatting

If omitted, the bitmask of table style options on the current table shall be assumed to be `0000`.

[*Example*: Consider a table which shall use the following conditional formatting properties from the referenced table style:

- First row conditional formatting
- Last row conditional formatting

This table would then apply the following portions of the bitmask:

- 0x0020=Apply first row conditional formatting
- 0x0040=Apply last row conditional formatting
- 0x0200=Do not apply row banding conditional formatting
- 0x0400=Do not apply column banding conditional formatting

The resulting WordprocessingML would be specified as follows:

```
<w:tblPr>
  <w:tblLook w:val="0660"/>
</w:tblPr>
```

The tblLook element specifies a bitmask which determines the components of the table style applied to the current table. *end example*]

Parent Elements
tblPr (§2.7.5.3); tblPr (§2.7.5.4); tblPr (§2.4.55); tblPr (§2.4.56)

Attributes	Description

val (Two Digit Hexadecimal Value)	Specifies a value specified as a two digit hexadecimal number), whose contents are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:tblPr>`
	`  <w:tblLook w:val="0010" />`
	`</w:tblPr>`
	
	The value of `0010` is interpreted in the context of the parent element. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_ShortHexNumber simple type (§2.18.86).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShortHexNumber">
   <attribute name="val" type="ST_ShortHexNumber" use="required"/>
</complexType>
```

2.4.52 tblLook (Table Style Conditional Formatting Settings Exception)

This element specifies the components of the conditional formatting of the referenced table style (if one exists) which shall be applied to the set of table rows with the current table-level property exceptions. A table style can specify up to six different optional conditional formats [*Example*: Different formatting for first column. *end example*], which then can be applied or omitted from individual table rows in the parent table.

This element's value is hexadecimal code containing a bitmask of options, interpreted as follows:

- 0x0020=Apply first row conditional formatting
- 0x0040=Apply last row conditional formatting
- 0x0080=Apply first column conditional formatting
- 0x0100=Apply last column conditional formatting
- 0x0200=Do not apply row banding conditional formatting
- 0x0400=Do not apply column banding conditional formatting

If omitted, the bitmask of table style options on the current table row shall be assumed to be the value specified on the table-level properties.

[*Example*: Consider a table which shall use the following conditional formatting properties from the referenced table style:

- First row conditional formatting
- Last row conditional formatting

This table would then apply the following portions of the bitmask:

- 0x0020=Apply first row conditional formatting
- 0x0040=Apply last row conditional formatting
- 0x0200=Do not apply row banding conditional formatting
- 0x0400=Do not apply column banding conditional formatting

The resulting WordprocessingML would be specified as follows:

```
<w:tblPrEx>
  <w:tblLook w:val="0660"/>
</w:tblPrEx>
```

The tblLook element specifies a bitmask which determines the components of the table style applied to the current table. *end example*]

Parent Elements
tblPrEx (§2.4.57); tblPrEx (§2.4.58)

Attributes	Description

val (Two Digit Hexadecimal Value)	Specifies a value specified as a two digit hexadecimal number), whose contents are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:tblPr>`
	`  <w:tblLook w:val="0010" />`
	`</w:tblPr>`
	
	The value of `0010` is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_ShortHexNumber simple type (§2.18.86).

The following XML Schema fragment defines the contents of this element:

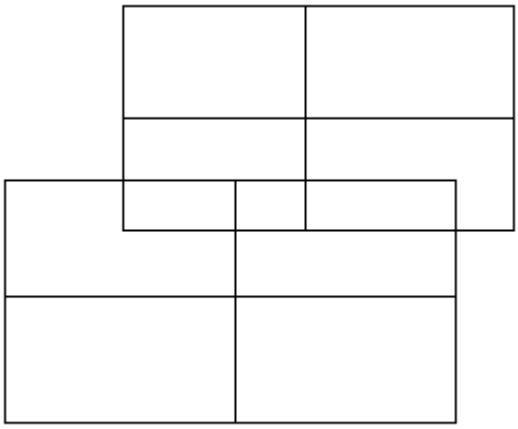
```
<complexType name="CT_ShortHexNumber">
   <attribute name="val" type="ST_ShortHexNumber" use="required"/>
</complexType>
```

2.4.53 tblOverlap (Floating Table Allows Other Tables to Overlap)

This element specifies whether the current table shall allow other floating tables to overlap its extents when the tables are displayed in a document. If specified, then no adjustment shall be made to prevent tables whose properties would normally cause them to overlap from overlapping when displayed. If turned off, then the tables shall be adjusted as needed to prevent them from overlapping when displayed by adjusting the floating table properties as needed.

If this element is omitted on a given table, then this table shall allow other tables to overlap when displayed. If the parent table is not floating via the tblpPr element (§2.4.54), then this element shall be ignored.

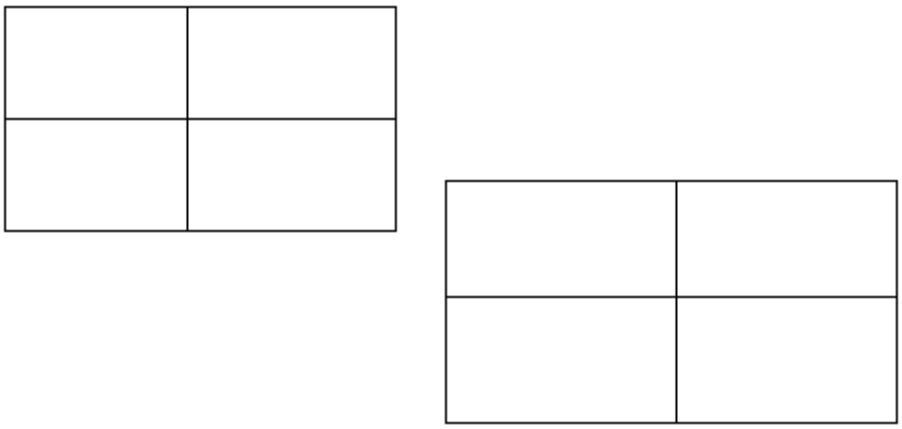
[*Example*: Consider two floating tables in a WordprocessingML document which overlap when displayed, as follows:



If either of these tables specifies that it shall not allow overlapping, using the following WordprocessingML:

```
<w:tblPr>
  <w:tblOverlap w:val="never"/>
</w:tblPr>
```

The resulting tables shall not overlap, and must be adjusted at display time to prevent any overlapping, for example:



The tblOverlap element with a value of never specifies that the specified table cannot overlap with other floating tables in the document. *end example*]

Parent Elements
tblPr (§2.7.5.3); tblPr (§2.7.5.4); tblPr (§2.4.55); tblPr (§2.4.56)

Attributes	Description
val (Floating Table Overlap Setting)	Specifies whether a floating table shall allow other floating tables in the document to overlap its extents when displayed. [*Example*: The following WordprocessingML specifies that the table is not allowed to overlap: `<w:tblOverlap w:val="never" />` *end example*] The possible values for this attribute are defined by the ST_TblOverlap simple type (§2.18.95).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblOverlap">
   <attribute name="val" type="ST_TblOverlap" use="required"/>
</complexType>
```

2.4.54 tblpPr (Floating Table Positioning)

This element specifies information about the current table with regard to floating tables. *Floating tables* are tables in a document which are not part of the main text flow in the document, and are instead absolutely positioned with a specific size and position relative to non-frame content in the current document.

The first piece of information specified by the tblpPr element is that the current table is actually a floating table. This information is specified simply by the presence of the tblpPr element in table's properties. If the tblpPr element is omitted, the table shall not floating in the document.

The second piece of information is the positioning of the table, which is specified by the attribute values stored on the tblpPr element. In all absolute positioning cases, the positioning of the table is relative to its top-left corner position. For relative positioning (e.g. center), the positioning of the table is relative to its entire frame.

Note that the table still has a logical position in the file (its location within the block-level elements in the document). This logical location shall be used to calculate the position of the table relative to a paragraph, using the next regular (non-table, non-frame) paragraph in the document.

[*Example*: Consider a floating table which is positioned three inches from the edge of the page extents on both its top and left edges (i.e. the top-left corner occurs at 3" x 3"). This floating table would be specified using the following WordprocessingML:

```
<w:tbl>
  <w:tblPr>
    <w:tblpPr w:leftFromText="144" w:rightFromText="144" w:topFromText="144"
w:bottomFromText="144" w:vertAnchor="page" w:horzAnchor="page" w:tblpX="4320"
w:tblpY="4320"/>
    …
  </w:tblPr>
  …</w:tbl>
```

The presence of the tblpPr element dictates that this table is a floating table, and its attributes specify that the floating table shall be anchored 4320 twentieths of a point (3 inches) from the top and left edges of the current page. *end example*]

Parent Elements
tblPr (§2.7.5.3); tblPr (§2.7.5.4); tblPr (§2.4.55); tblPr (§2.4.56)

Attributes	Description

bottomFromText (Distance From Bottom of Table to Text)	Specifies the minimum distance which shall be maintained between the current floating table and the top of text in the paragraph which is below this floating table.
	
	This distance is expressed in twentieths of a point.
	
	If this attribute is omitted, its value shall be assumed to be 0.
	
	[*Example*: Consider a floating table which should have a minimum of a one-half inch spacing from any text on its bottom side. This constraint would be specified using the following WordprocessingML:
	
	`<w:tblPr>`
	`  <w:tblpPr … w:bottomFromText="720" />`
	`</w:tblPr>`
	
	The bottomFromText attribute specifies that the spacing between text and this floating table shall be a minimum of 720 twentieths of a point. *end example*]
	
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

horzAnchor (Table Horizontal Anchor)	Specifies the base object from which the horizontal positioning in the tblpX and/or tblpXSpec attribute should be calculated. A floating table may be horizontally positioned relative to: • The vertical edge of the page before any runs of text (the left edge for left-to-right paragraphs, the right edge for right-to-left paragraphs) • The vertical edge of the text margin before any runs of text (the left edge for left-

Attributes	Description

	to-right paragraphs, the right edge for right-to-left paragraphs)
	• The vertical edge of the text margin for the column in which the anchor paragraph is located
	
	If this attribute is omitted, then its value shall be assumed to be page.
	
	[*Example*: Consider a floating table which should be positioned one inch to the right of its column in a left-to-right document. This floating table would be specified using the following WordprocessingML:
	
	`<w:tblPr>`
	`  <w:tblpPr … w:tblpX="1440" w:horzAnchor="column" />`
	`</w:tblPr>`
	
	These table properties specify that they are relative to the current `column`, and that relative to that column, the floating table should be `1440` twentieths of a point. *end example*]
	
	The possible values for this attribute are defined by the ST_HAnchor simple type (§2.18.40).

leftFromText (Distance From Left of Table to Text)	Specifies the minimum distance which shall be maintained between the current floating table and the edge of text in the paragraph which is to the left of this floating table.
	
	This distance is expressed in twentieths of a point.
	
	If this attribute is omitted, its value shall be assumed to be `0`.
	
	[*Example*: Consider a floating table which should have a minimum of a one-half inch spacing from any text on its left. This constraint would be specified using the following WordprocessingML:
	
	`<w:tblPr>`
	`  <w:tblpPr … w:leftFromText="720" />`
	`</w:tblPr>`
	
	The leftFromText attribute specifies that the spacing between text and this floating table shall be a minimum of `720` twentieths of a point. *end example*]
	
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

rightFromText ((Distance From Right of Table to Text)	Specifies the minimum distance which shall be maintained between the current floating table and the edge of text in the paragraph which is to the right of this floating table. This distance is expressed in twentieths of a point.

Attributes	Description

	If this attribute is omitted, its value shall be assumed to be 0.
	
	[*Example*: Consider a floating table which should have a minimum of a one-half inch spacing from any text on its right. This constraint would be specified using the following WordprocessingML:
	
	`<w:tblPr>`
	`  <w:tblpPr … w:rightFromText="720" />`
	`</w:tblPr>`
	
	The rightFromText attribute specifies that the spacing between text and this floating table shall be a minimum of 720 twentieths of a point. *end example*]
	
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

tblpX (Absolute Horizontal Distance From Anchor)	Specifies an absolute horizontal position for the floating table. This absolute position is specified relative to the horizontal anchor specified by the horzAnchor attribute for this floating table.
	
	This value is expressed in twentieths of a point. If it is positive, then the floating table is positioned after the anchor object in the direction of horizontal text flow in this document. If it is negative, then the floating table is positioned before the anchor object in the direction of horizontal text flow in this document.
	
	If the tblpXSpec attribute is also specified, then this value is ignored. If this attribute is omitted, then its value shall be assumed to be 0.
	
	[*Example*: Consider the following WordprocessingML fragment specifying a floating table:
	
	`<w:tbl>`
	`  <w:tblPr>`
	`  <w:tblpPr … w:horizAnchor="page" w:tblpX="1643"/>`
	`  </w:tblPr>`
	`  …`
	`</w:tbl>`
	
	This floating table specifies that it should be located exactly 1643 twentieths of a point after the vertical edge of the page (from the horizAnchor attribute). *end example*]
	
	The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§2.18.88).

tblpXSpec (Relative Horizontal Alignment From Anchor)	Specifies a relative horizontal position for the floating table. This relative position is specified relative to the horizontal anchor specified by the horizAnchor attribute for this floating table. If omitted, this attribute is not specified and the value of the tblpX attribute determines

Attributes	Description

	the absolute horizontal position of the floating table. If specified, the position for this attribute supersedes any value which is specified in the tblpX attribute, and that value is ignored.
	
	[*Example*: Consider the following WordprocessingML fragment specifying a floating table:
	
	`<w:tbl>`
	`  <w:tblPr>`
	`  <w:tblpPr … w:horizAnchor="page" w:tblpX="1643"`
	`w:tblpXSpec="left"/>`
	`  </w:tblPr>`
	`  …`
	`</w:tbl>`
	
	This floating table specifies that it has a horizontal placement of exactly 1643 twentieths of a point relative to the page, but that exact placement is overridden by the presence of the tblpXSpec attribute to place the table on the `left` side of the page. *end example*]
	
	The possible values for this attribute are defined by the ST_XAlign simple type (§2.18.114).

tblpY (Absolute Vertical Distance From Anchor)	Specifies an absolute vertical position for the floating table. This absolute position is specified relative to the vertical anchor specified by the vertAnchor attribute for this floating table.
	
	This value is expressed in twentieths of a point. If it is positive, then the floating table is positioned after the anchor object in the direction of vertical text flow in this document. If it is negative, then the floating table is positioned before the anchor object in the direction of vertical text flow in this document.
	
	If the tblpYSpec attribute is also specified, then this value is ignored. If this attribute is omitted, then its value shall be assumed to be `0`.
	
	[*Example*: Consider the following WordprocessingML fragment specifying a floating table:
	
	`<w:tbl>`
	`  <w:tblPr>`
	`  <w:tblpPr … w:vertAnchor="text" w:tblpY="73" />`
	`  </w:tblPr>`
	`  …`
	`</w:tbl>`
	
	This floating table specifies that it should be located exactly 79 twentieths of a point below the top vertical edge of the anchor's paragraph's `text` (from the vertAnchor attribute), assuming that the vertical text direction is top to bottom. *end example*]
	
	The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple

Attributes	Description
	type (§2.18.88).

tblpYSpec (Relative Vertical Alignment from Anchor)	Specifies a relative vertical position for the floating table. This relative position is specified relative to the vertical anchor specified by the vertAnchor attribute for this floating table.
	
	If omitted, this attribute is not specified and the value of the tblpY attribute determines the absolute horizontal position of the floating table. If specified, the position for this attribute supersedes any value which is specified in the tblpY attribute, and that value is ignored, unless the vertAnchor is set to `text`, in which case any relative positioning is not allowed, and is itself ignored.
	
	[*Example*: Consider the following WordprocessingML fragment specifying a floating table:
	
	`<w:tbl>`
	`  <w:tblPr>`
	`  <w:tblpPr … w:vertAnchor="margin" w:tblpY="73"`
	`w:tblpYSpec="center"/>`
	`  </w:tblPr>`
	`  …`
	`</w:tbl>`
	
	This floating table specifies that it has a vertical placement of exactly `73` twentieths of a point relative to the top `margin`, but that exact placement is overridden by the presence of the tblpYSpec attribute to place the table in the `center` of the margin. *end example*]
	
	The possible values for this attribute are defined by the ST_YAlign simple type (§2.18.115).

topFromText (Distance From Top of Table to Text)	Specifies the minimum distance which shall be maintained between the current floating table and the bottom edge of text in the paragraph which is above this floating table.
	
	This distance is expressed in twentieths of a point.
	
	If this attribute is omitted, its value shall be assumed to be `0`.
	
	[*Example*: Consider a floating table which should have a minimum of a one-half inch spacing from any text above it. This constraint would be specified using the following WordprocessingML:
	
	`<w:tblPr>`
	`  <w:tblpPr … w:topFromText="720" />`
	`</w:tblPr>`
	
	The topFromText attribute specifies that the spacing between text and this floating table shall be a minimum of `720` twentieths of a point. *end example*]
	
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type

Attributes	Description
	(§2.18.105).

vertAnchor (Table Vertical Anchor)	Specifies the base object from which the vertical positioning in the tblpY attribute should be calculated.
	
	A floating table may be horizontally positioned relative to:
	• The horizontal edge of the page before any runs of text (the top edge for top-to-bottom sections, the bottom for bottom-to-top sections)
	• The horizontal edge of the text margin before any runs of text (the top edge for top-to-bottom sections, the bottom for bottom-to-top sections)
	• The horizontal edge of the page before any runs of text (the top edge for top-to-bottom sections, the bottom for bottom-to-top sections)
	
	If this attribute is omitted, then its value shall be assumed to be page.
	
	[*Example*: Consider a floating table which should be positioned two inches below the page top in a top-to-bottom document. This floating table would be specified using the following WordprocessingML:
	
	`<w:tblPr>`
	`  <w:tblpPr … w:tblpY="2880" w:vertAnchor="page" />`
	`</w:tblPr>`
	
	These floating table properties specify that they are relative to the anchor page, and that relative to that page, the table should be 2880 twentieths of a point in the direction of the flow of text (down, in this case). *end example*]
	
	The possible values for this attribute are defined by the ST_VAnchor simple type (§2.18.109).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblPPr">
   <attribute name="leftFromText" type="ST_TwipsMeasure"/>
   <attribute name="rightFromText" type="ST_TwipsMeasure"/>
   <attribute name="topFromText" type="ST_TwipsMeasure"/>
   <attribute name="bottomFromText" type="ST_TwipsMeasure"/>
   <attribute name="vertAnchor" type="ST_VAnchor"/>
   <attribute name="horzAnchor" type="ST_HAnchor"/>
   <attribute name="tblpXSpec" type="ST_XAlign"/>
   <attribute name="tblpX" type="ST_SignedTwipsMeasure"/>
   <attribute name="tblpYSpec" type="ST_YAlign"/>
   <attribute name="tblpY" type="ST_SignedTwipsMeasure"/>
</complexType>
```

2.4.55 tblPr (Table Properties)

This element specifies the set of table-wide properties applied to the current table. These properties affect the appearance of all rows and cells within the parent table, but may be overridden by individual table-level exception, row, and cell level properties as defined by each property.

[*Example*: Consider the following simple WordprocessingML table:

This table defines a one point single border for all border types and is set to 100% of page width - both table-wide properties. The resulting table is represented by the following WordprocessingML:

```
<w:tbl>
  <w:tblPr>
    <w:tblW w:w="0" w:type="auto"/>
    <w:tblBorders>
      <w:top w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:left w:val="single" w:sz="4 w:space="0" w:color="auto"/>
      <w:bottom w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:right w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:insideH w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:insideV w:val="single" w:sz="4" w:space="0" w:color="auto"/>
    </w:tblBorders>
  </w:tblPr>
  …
</w:tbl>
```

In this example, the tblW element (§2.4.61) defines the total width of the table, which, in this case, is set to a type of auto, which specifies that the table should be automatically sized to fit its contents. The tblBorders element (§2.4.38) specifies each of the table's borders, and specifies a one point border on the top, left, bottom, right and inside horizontal and vertical border. *end example*]

Parent Elements
tbl (§2.4.36)

Child Elements	Subclause
bidiVisual (Visually Right to Left Table)	§2.4.1
jc (Table Alignment)	§2.4.23
shd (Table Shading)	§2.4.35
tblBorders (Table Borders)	§2.4.38

Child Elements	Subclause
tblCellMar (Table Cell Margin Defaults)	§2.4.39
tblCellSpacing (Table Cell Spacing Default)	§2.4.43
tblInd (Table Indent from Leading Margin)	§2.4.48
tblLayout (Table Layout)	§2.4.49
tblLook (Table Style Conditional Formatting Settings)	§2.4.51
tblOverlap (Floating Table Allows Other Tables to Overlap)	§2.4.53
tblpPr (Floating Table Positioning)	§2.4.54
tblPrChange (Revision Information for Table Properties)	§2.13.5.36
tblStyle (Referenced Table Style)	§2.4.59
tblStyleColBandSize (Number of Columns in Column Band)	§2.7.5.5
tblStyleRowBandSize (Number of Rows in Row Band)	§2.7.5.7
tblW (Preferred Table Width)	§2.4.61

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblPr">
   <complexContent>
      <extension base="CT_TblPrBase">
         <sequence>
            <element name="tblPrChange" type="CT_TblPrChange" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.4.56 tblPr (Previous Table Properties)

This element specifies a previous set of table properties, the modifications to which shall be attributed to a revision by a particular author and at a particular time. This element contains the table property settings which were previously in place before a specific set of revisions by one author. These properties affect the appearance of all rows and cells within the parent table, but may be overridden by individual table-level exception, row, and cell level properties, as defined by each property.

[*Example*: Consider the following simple WordprocessingML table:

If the table justification is set to `center` and the table shading to set to red with revision marking on, as follows:

The revision tracked on this table would be specified as follows in the WordprocessingML:

```
<w:tblPr>
  <w:tblStyle w:val="TableGrid"/>
  <w:tblW w:w="0" w:type="auto"/>
  <w:jc w:val="center"/>
  <w:shd w:val="clear" w:color="auto" w:fill="FF0000"/>
  <w:tblLook w:val="04A0"/>
  <w:tblPrChange w:id="0" … >
    <w:tblPr>
      <w:tblStyle w:val="TableGrid"/>
      <w:tblW w:w="0" w:type="auto"/>
      <w:tblLook w:val="04A0"/>
    </w:tblPr>
  </w:tblPrChange>
</w:tblPr>
```

The tblPr element as a child of tblPrChange contains the previous definition for the table properties, consisting of the properties set before the current tracked revision. *end example*]

Parent Elements
tblPrChange (§2.13.5.36)

Child Elements	Subclause
bidiVisual (Visually Right to Left Table)	§2.4.1
jc (Table Alignment)	§2.4.23
shd (Table Shading)	§2.4.35
tblBorders (Table Borders)	§2.4.38
tblCellMar (Table Cell Margin Defaults)	§2.4.39
tblCellSpacing (Table Cell Spacing Default)	§2.4.43
tblInd (Table Indent from Leading Margin)	§2.4.48
tblLayout (Table Layout)	§2.4.49
tblLook (Table Style Conditional Formatting Settings)	§2.4.51
tblOverlap (Floating Table Allows Other Tables to Overlap)	§2.4.53
tblpPr (Floating Table Positioning)	§2.4.54
tblStyle (Referenced Table Style)	§2.4.59
tblStyleColBandSize (Number of Columns in Column Band)	§2.7.5.5
tblStyleRowBandSize (Number of Rows in Row Band)	§2.7.5.7
tblW (Preferred Table Width)	§2.4.61

The following XML Schema fragment defines the contents of this element:

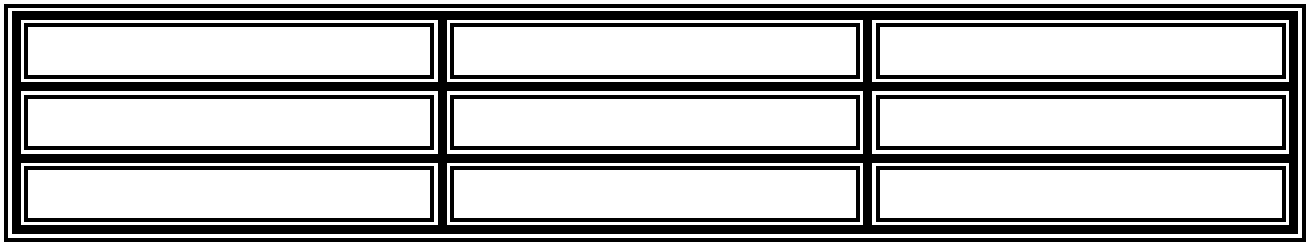
```
<complexType name="CT_TblPrBase">
   <sequence>
      <element name="tblStyle" type="CT_String" minOccurs="0"/>
      <element name="tblpPr" type="CT_TblPPr" minOccurs="0" maxOccurs="1"/>
      <element name="tblOverlap" type="CT_TblOverlap" minOccurs="0" maxOccurs="1"/>
      <element name="bidiVisual" type="CT_OnOff" minOccurs="0" maxOccurs="1"/>
      <element name="tblStyleRowBandSize" type="CT_DecimalNumber" minOccurs="0" maxOccurs="1"/>
      <element name="tblStyleColBandSize" type="CT_DecimalNumber" minOccurs="0" maxOccurs="1"/>
      <element name="tblW" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="jc" type="CT_Jc" minOccurs="0" maxOccurs="1"/>
      <element name="tblCellSpacing" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="tblInd" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="tblBorders" type="CT_TblBorders" minOccurs="0" maxOccurs="1"/>
      <element name="shd" type="CT_Shd" minOccurs="0" maxOccurs="1"/>
      <element name="tblLayout" type="CT_TblLayoutType" minOccurs="0" maxOccurs="1"/>
      <element name="tblCellMar" type="CT_TblCellMar" minOccurs="0" maxOccurs="1"/>
      <element name="tblLook" type="CT_ShortHexNumber" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

2.4.57 tblPrEx (Table-Level Property Exceptions)

This element specifies a set of table properties which shall be applied to the contents of this row in place of the table properties specified in the tblPr element.

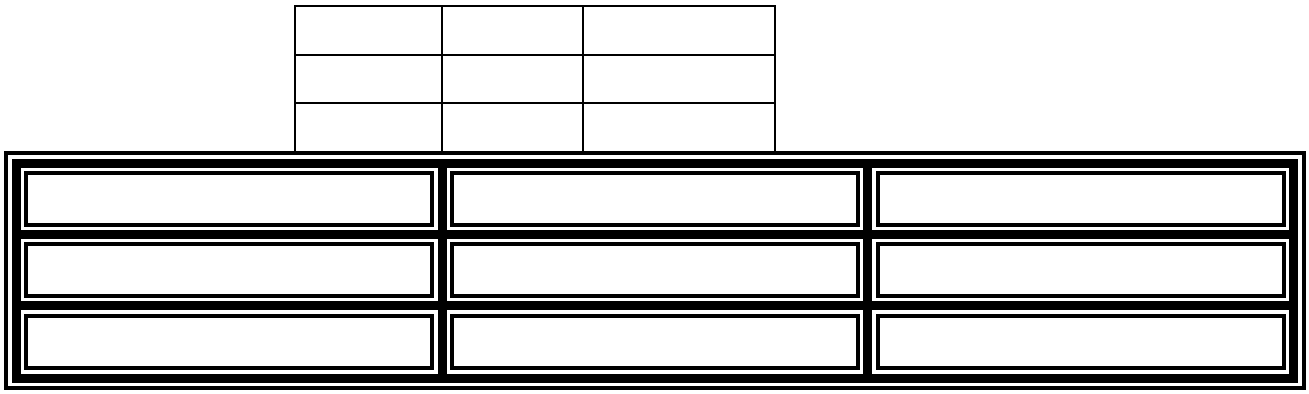
[*Note*: These properties are typically used in cases involving legacy documents, as well as cases where two existing independent tables are merged (in order to prevent the look of the second table from being superseded by the first table). *end note*]

[*Example*: Consider the following two tables in a WordprocessingML document:



These two tables each have a different set of table level borders. If the interceding paragraphs between these two tables is removed and the tables are merged together, it is obviously undesirable to have the second table

lose its formatting and match the properties of the first table. Therefore, when the tables are merged as follows (note that there is now only one table):



The resulting WordprocessingML for the last three rows of the table would include the following set of table-level property exceptions:

```
<w:tr>
  <w:tblPrEx>
    <w:tblBorders>
      <w:top w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
w:color="auto"/>
      <w:left w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
w:color="auto"/>
      <w:bottom w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
w:color="auto"/>
      <w:right w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
w:color="auto"/>
      <w:insideH w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
w:color="auto"/>
      <w:insideV w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
w:color="auto"/>
    </w:tblBorders>
  </w:tblPrEx>
  …
</w:tr>
```

The tblPrEx element contains all table-level properties which are being overridden for the current row in the table. *end example*]

Parent Elements
tr (§2.4.75)

Child Elements	Subclause
jc (Table Alignment Exception)	§2.4.21
shd (Table Shading Exception)	§2.4.34
tblBorders (Table Borders Exceptions)	§2.4.37
tblCellMar (Table Cell Margin Exceptions)	§2.4.40
tblCellSpacing (Table Cell Spacing Exception)	§2.4.41
tblInd (Table Indent from Leading Margin Exception)	§2.4.47
tblLayout (Table Layout Exception)	§2.4.50
tblLook (Table Style Conditional Formatting Settings Exception)	§2.4.52
tblPrExChange (Revision Information for Table-Level Property Exceptions)	§2.13.5.37
tblW (Preferred Table Width Exception)	§2.4.60

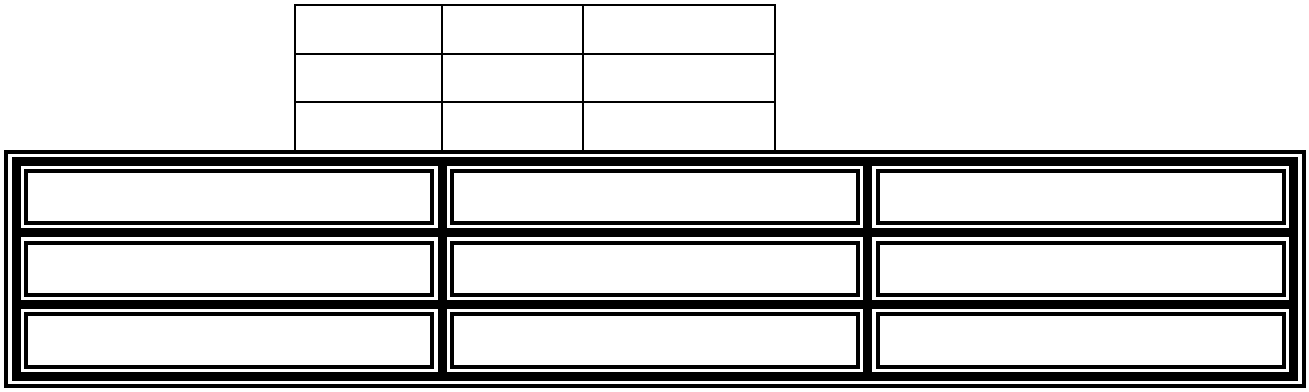
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblPrEx">
   <complexContent>
      <extension base="CT_TblPrExBase">
         <sequence>
            <element name="tblPrExChange" type="CT_TblPrExChange" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.4.58 tblPrEx (Previous Table-Level Property Exceptions)

This element specifies a previous set of table-level property exceptions, the modifications to which shall be attributed to a revision by a particular author and at a particular time. This element contains the table-level property exceptions which were previously in place before a specific set of revisions by one author.

[*Example*: Consider the following two tables in a WordprocessingML document. If the interceding paragraphs between these two tables is removed and the tables are merged together, it is obviously undesirable to have the second table lose its formatting and match the properties of the first table. Therefore, when the tables are merged as follows (note that there is now only one table):



If the border type is changed to a red border of type `thinThickThinSmallGap` with revisions tracked, as follows:

The resulting WordprocessingML for the last three rows of the table would include the following set of table-level property exceptions with revision tracking:

```
<w:tr>
  <w:tblPrEx>
    <w:tblBorders>
      <w:top w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
       w:color="auto"/>
      <w:left w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
       w:color="auto"/>
      <w:bottom w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
       w:color="auto"/>
      <w:right w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
       w:color="auto"/>
      <w:insideH w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
       w:color="auto"/>
      <w:insideV w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
       w:color="auto"/>
    </w:tblBorders>
    <w:tblPrExChange w:id="9" … >
      <w:tblPrEx>
        <w:tblBorders>
          <w:top w:val="thinThickThinSmallGap" w:sz="24" w:space="0"
           w:color="FF0000"/>
          <w:left w:val="thinThickThinSmallGap" w:sz="24" w:space="0"
           w:color="FF0000"/>
          <w:bottom w:val="thinThickThinSmallGap" w:sz="24" w:space="0"
           w:color="FF0000"/>
          <w:right w:val="thinThickThinSmallGap" w:sz="24" w:space="0"
           w:color="FF0000"/>
          <w:insideH w:val="thinThickThinSmallGap" w:sz="24" w:space="0"
           w:color="FF0000"/>
          <w:insideV w:val="thinThickThinSmallGap" w:sz="24" w:space="0"
           w:color="FF0000"/>
        </w:tblBorders>
      </w:tblPrEx>
    </w:tblPrExChange>
  </w:tblPrEx>
  …
</w:tr>
```

The tblPrEx element as a child of tblPrExChange contains the previous definition for the table-level property exceptions, consisting of the properties set before the current tracked revision. *end example*]

Parent Elements
tblPrExChange (§2.13.5.37)

Child Elements	Subclause
jc (Table Alignment Exception)	§2.4.21
shd (Table Shading Exception)	§2.4.34
tblBorders (Table Borders Exceptions)	§2.4.37
tblCellMar (Table Cell Margin Exceptions)	§2.4.40
tblCellSpacing (Table Cell Spacing Exception)	§2.4.41
tblInd (Table Indent from Leading Margin Exception)	§2.4.47
tblLayout (Table Layout Exception)	§2.4.50
tblLook (Table Style Conditional Formatting Settings Exception)	§2.4.52
tblW (Preferred Table Width Exception)	§2.4.60

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblPrExBase">
   <sequence>
      <element name="tblW" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="jc" type="CT_Jc" minOccurs="0" maxOccurs="1"/>
      <element name="tblCellSpacing" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="tblInd" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="tblBorders" type="CT_TblBorders" minOccurs="0" maxOccurs="1"/>
      <element name="shd" type="CT_Shd" minOccurs="0" maxOccurs="1"/>
      <element name="tblLayout" type="CT_TblLayoutType" minOccurs="0" maxOccurs="1"/>
      <element name="tblCellMar" type="CT_TblCellMar" minOccurs="0" maxOccurs="1"/>
      <element name="tblLook" type="CT_ShortHexNumber" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

2.4.59 tblStyle (Referenced Table Style)

This element specifies the style ID of the table style which shall be used to format the contents of this table.

This formatting is applied at the following location in the *style hierarchy*:

- Document defaults
- Table styles (this element)
- Numbering styles
- Paragraph styles
- Character styles
- Direct Formatting

This means that all properties specified in the style element (§2.7.3.17) with a styleId which corresponds to the value in this element's val attribute are applied to the table at the appropriate level in the hierarchy.

If this element is omitted, or it references a style which does not exist, then no table style shall be applied to the current table. As well, this property is ignored if the table properties are themselves part of a table style.

[*Example*: Consider the following WordprocessingML fragment:

```
<w:tblPr>
  <w:tblStyle w:val=”TestTableStyle” />
</w:tblPr>
```

This table specifies that it will inherit all of the table properties specified by the table style with a styleId of `TestTableStyle`. *end example*]

Parent Elements
tblPr (§2.7.5.3); tblPr (§2.7.5.4); tblPr (§2.4.55); tblPr (§2.4.56)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.4.60 tblW (Preferred Table Width Exception)

This element specifies the preferred width for the parent table row via a set of table-level property exceptions. This preferred width is used as part of the table layout algorithm specified by the tblLayout element (§2.4.49n; §2.4.50) - full description of the algorithm in the ST_TblLayout simple type (§2.18.94).

All widths in a table are considered preferred because:

- The table must satisfy the shared columns as specified by the tblGrid element (§2.4.44)
- Two or more widths may have conflicting values for the width of the same grid column
- The table layout algorithm (§2.18.94) may require a preference to be overridden

This value is specified in the units applied via its type attribute. Any width value of type `pct` for this element shall be calculated relative to the text extents of the page (page width excluding margins).

If this element is omitted, then the cell width shall be of type `auto`.

[*Example*: Consider a row in a WordprocessingML table defined as follows:

```
<w:tr>
  <w:trPr>
    <w:tblPrEx>
      <w:tblW w:type="fixed" w:w="1440"/>
    </w:tblPrEx>
  </w:trPr>
  …
</w:tr>
```

This table-level property exception specifies that it has a preferred table width of `1440` twentieths of a point (one inch). The resulting table row would therefore be sized such that the table maintains that preferred width, as follows:

	Hello world	

The text `Hello world` makes the middle cell larger, and the other two cells are size to maintain the preferred widths of one inch for the overall table width:

	Hello world this is a longer string.	

However, when the middle table cell requires a larger width to accommodate non-breaking text, that preference may be overridden as needed:

	Hello worlddddddddddddddddddddddddddddddd	

In this case, the middle cell's long non breaking string caused the table to be expanded to prevent breaking the string, and therefore to override the preferred width on the table row. *end example*]

Parent Elements
tblPrEx (§2.4.57); tblPrEx (§2.4.58)

Attributes	Description
type (Table Width Type)	Specifies the units of the width property being defined by the parent element's w attribute. This property is used to define various properties of a table, including: cell spacing, preferred width, and table margins. If this attribute is omitted, then its value shall be assumed to be dxa (twentieths of a point). [*Example*: Consider a table with a table cell bottom cell spacing with a type of dxa, as follows: `<w:bottom ... w:type="dxa" />` This type shall therefore be used to interpret the width specified in the w attribute as a value in twentieths of a point. *end example*] The possible values for this attribute are defined by the ST_TblWidth simple type (§2.18.97).
w (Table Width Value)	Specifies the value of the width property being defined by the parent element. This property is used to define various properties of a table, including: cell spacing, preferred widths, and table margins. If this attribute is omitted, then its value shall be assumed to be 0. [*Example*: Consider a table with a bottom margin with a width of 302, as follows: `<w:bottom w:w="302" w:type="dxa" />` The value in the w attribute shall therefore be used to determine the width being specified in the context of the units specified in the type attribute. In this case, the type is twentieths of a point (dxa), so the width is 302 twentieths of a point (.2097 inches). *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblWidth">
   <attribute name="w" type="ST_DecimalNumber"/>
   <attribute name="type" type="ST_TblWidth"/>
</complexType>
```

2.4.61 tblW (Preferred Table Width)

This element specifies the preferred width for this table. This preferred width is used as part of the table layout algorithm specified by the tblLayout element (§2.4.49; §2.4.50) - full description of the algorithm in the ST_TblLayout simple type (§2.18.94).

All widths in a table are considered preferred because:

- The table must satisfy the shared columns as specified by the tblGrid element (§2.4.44)
- Two or more widths may have conflicting values for the width of the same grid column
- The table layout algorithm (§2.18.94) may require a preference to be overridden

This value is specified in the units applied via its type attribute. Any width value of type `pct` for this element shall be calculated relative to the text extents of the page (page width excluding margins).

If this element is omitted, then the cell width shall be of type `auto`.

[*Example*: Consider a WordprocessingML table defined as follows:

```
<w:tbl>
  <w:tblPr>
    <w:tblW w:type="fixed" w:w="1440"/>
  </w:tblPr>
  …
</w:tbl>
```

This table specifies that it has a preferred table width of `1440` twentieths of a point (one inch). The resulting table would therefore be sized such that the table maintains that preferred width, as follows:

	Hello world	

The text `Hello world` makes the middle cell larger, and the other two cells are size to maintain the preferred widths of one inch for the overall table width:

	Hello world this is a longer string.	

However, when the middle table cell requires a larger width to accommodate non-breaking text, that preference may be overridden as needed:

	Hello worlddddddddddddddddddddddddddddddd	

In this case, the middle cell's long non breaking string caused the table to be expanded to prevent breaking the string, and therefore to override the preferred width on the table. *end example*]

Parent Elements
tblPr (§2.7.5.3); tblPr (§2.7.5.4); tblPr (§2.4.55); tblPr (§2.4.56)

Attributes	Description
type (Table Width Type)	Specifies the units of the width property being defined by the parent element's w attribute. This property is used to define various properties of a table, including: cell spacing, preferred width, and table margins. If this attribute is omitted, then its value shall be assumed to be dxa (twentieths of a point). [*Example*: Consider a table with a table cell bottom cell spacing with a type of dxa, as follows: `<w:bottom ... w:type="dxa" />` This type shall therefore be used to interpret the width specified in the w attribute as a value in twentieths of a point. *end example*] The possible values for this attribute are defined by the ST_TblWidth simple type (§2.18.97).
w (Table Width Value)	Specifies the value of the width property being defined by the parent element. This property is used to define various properties of a table, including: cell spacing, preferred widths, and table margins. If this attribute is omitted, then its value shall be assumed to be 0.

Attributes	Description
	[*Example*: Consider a table with a bottom margin with a width of `302`, as follows: `<w:bottom w:w="302" w:type="dxa" />` The value in the w attribute shall therefore be used to determine the width being specified in the context of the units specified in the type attribute. In this case, the type is twentieths of a point (`dxa`), so the width is `302` twentieths of a point (.2097 inches). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblWidth">
   <attribute name="w" type="ST_DecimalNumber"/>
   <attribute name="type" type="ST_TblWidth"/>
</complexType>
```

2.4.62 tc (Table Cell)

This element specifies a single cell in a table row, which contains the table's content. Table cells in WordprocessingML are analogous to HTML `td` elements.

A tc element has one formatting child element, tcPr (§2.4.67), which defines the properties for the cell. Each unique property on the table cell is specified by a child element of this element. As well, a table cell can contain any valid block-level content, which allows for the nesting of paragraphs and tables within table cells.

If a table cell does not include at least one block-level element, then this document shall be considered corrupt.

[*Example*: Consider a table consisting of a single table cell, which contains the text `Hello World`:

Hello World

This table cell's content is represented by the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    <w:tcW w:w="0" w:type="auto"/>
  </w:tcPr>
  <w:p>
    <w:r>
      <w:t>Hello, World</w:t>
    </w:r>
  </w:p>
</w:tc>
```

The tc element contains a set of cell-level properties defined using the tcPr element, and a single block-level element - in this case, a paragraph. *end example*]

Parent Elements
customXml (§2.5.1.3); sdtContent (§2.5.2.33); tr (§2.4.75)

Child Elements	Subclause
altChunk (Anchor for Imported External Content)	§2.17.3.1
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Block-Level Custom XML Element)	§2.5.1.6
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
del (Deleted Run Content)	§2.13.5.12
ins (Inserted Run Content)	§2.13.5.20
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26

Child Elements	Subclause
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
p (Paragraph)	§2.3.1.22
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
sdt (Block-Level Structured Document Tag)	§2.5.2.30
tbl (Table)	§2.4.36
tcPr (Table Cell Properties)	§2.4.67

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Tc">
   <sequence>
      <element name="tcPr" type="CT_TcPr" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_BlockLevelElts" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

2.4.63 tcBorders (Table Cell Borders)

This element specifies the set of borders for the edges of the current table cell, using the eight border types defined by its child elements.

If the cell spacing for any row is non-zero as specified using the tblCellSpacing element (§2.4.41; §2.4.42; §2.4.43), then there is never a border conflict (as the non-zero cell spacing is applied above and beyond each individual cell border's width) and all table, table-level exception, and table cell borders shall be displayed.

If the cell spacing is zero, then there may be a conflict between two adjacent cell borders [*Example*: Between the left border of all cells in the second column and the right border of all cells in the first column of the table. *end example*], which shall be resolved as follows:

1. If either conflicting table cell border is `nil` or `none` (no border), then the opposing border shall be displayed.
2. If a cell border conflicts with a table border, the cell border always wins.
3. Each border shall then be assigned a weight using the following formula, and the border value using this calculation shall be displayed over the alternative border:

$$W_{\text{border}} = \#\text{ of lines in border} * \text{border number}$$

The border number shall be determined by this list:

single	1
thick	2
double	3
dotted	4
dashed	5
dotDash	6
dotDotDash	7
triple	8
thinThickSmallGap	9
thickThinSmallGap	10
thinThickThinSmallGap	11
thinThickMediumGap	12
thickThinMediumGap	13
thinThickThinMediumGap	14
thinThickLargeGap	15
thickThinLargeGap	16
thinThickThinLargeGap	17
wave	18
doubleWave	19
dashSmallGap	20
dashDotStroked	21
threeDEmboss	22
threeDEngrave	23
outset	24
inset	25

4. If the borders have an equal weight, than the higher of the two on this precedence list shall win:

- single
- thick
- double
- dotted
- dashed
- dotDash
- dotDotDash
- triple
- thinThickSmallGap
- thickThinSmallGap

- thinThickThinSmallGap
- thinThickMediumGap
- thickThinMediumGap
- thinThickThinMediumGap
- thinThickLargeGap
- thickThinLargeGap
- thinThickThinLargeGap
- wave
- doubleWave
- dashSmallGap
- dashDotStroked
- threeDEmboss
- threeDEngrave
- outset
- inset

5. If the borders have an identical style, than each border color shall be assigned a brightness value as follows:

$$\text{Brightness} = \text{R} + \text{B} + 2 * \text{G}$$

   The color with the smaller brightness value shall win.
6. If the borders have an identical brightness value above, than each border color shall be assigned a new brightness value as follows:

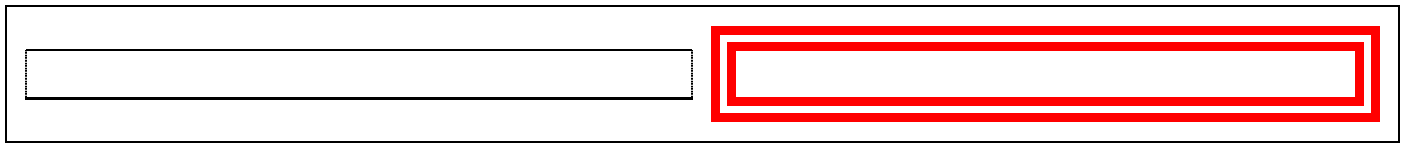
$$\text{Brightness} = \text{B} + 2 * \text{G}$$

   The color with the smaller brightness value shall win.
7. If the borders have an identical brightness value above, than each border color shall be assigned a brightness value as follows:

$$\text{Brightness} = \text{G}$$

   The color with the smaller brightness value shall win.
8. If the borders have an identical brightness value above, then they are functionally identical, and the first border in reading order should be displayed.

[*Example*: Consider the following two cell table (with exaggerated table cell spacing for clarity):



If we collapse the cell spacing, there will be conflicting borders at all edges. For each cell/table border conflict, rule #2 says that the cell border shall win. For the conflict in the center between two cell borders, rule #3 gives us a larger border weight for the right cell's border, resulting in the following table:

end example]

If this element is omitted, then this table shall have the borders specified by the associated table style. If no borders are specified in the style hierarchy, then this table shall not have any table borders.

[*Example*: Consider a table whose first cell specifies cell-level borders consisting of a think double red line, as follows:

These cell borders are specified using the following WordprocessingML:

```
<w:tcPr>
  <w:tcBorders>
    <w:top w:val="double" w:sz="24" w:space="0" w:color="FF0000"/>
    <w:left w:val="double" w:sz="24" w:space="0" w:color="FF0000"/>
    <w:bottom w:val="double" w:sz="24" w:space="0" w:color="FF0000"/>
    <w:right w:val="double" w:sz="24" w:space="0" w:color="FF0000"/>
  </w:tcBorders>
</w:tcPr>
```

The tcBorders element specifies the set of borders applied to the first cell as a 3 point double border. *end example*]

Parent Elements
tcPr (§2.7.5.8); tcPr (§2.4.66); tcPr (§2.7.5.9); tcPr (§2.4.67)

Child Elements	Subclause
bottom (Table Cell Bottom Border)	§2.4.3
insideH (Table Cell Inside Horizontal Edges Border)	§2.4.18
insideV (Table Cell Inside Vertical Edges Border)	§2.4.19
left (Table Cell Left Border)	§2.4.24
right (Table Cell Right Border)	§2.4.30
tl2br (Table Cell Top Left to Bottom Right Diagonal Border)	§2.4.70
top (Table Cell Top Border)	§2.4.74
tr2bl (Table Cell Top Right to Bottom Left Diagonal Border)	§2.4.76

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TcBorders">
   <sequence>
      <element name="top" type="CT_Border" minOccurs="0"/>
      <element name="left" type="CT_Border" minOccurs="0"/>
      <element name="bottom" type="CT_Border" minOccurs="0"/>
      <element name="right" type="CT_Border" minOccurs="0"/>
      <element name="insideH" type="CT_Border" minOccurs="0"/>
      <element name="insideV" type="CT_Border" minOccurs="0"/>
      <element name="tl2br" type="CT_Border" minOccurs="0"/>
      <element name="tr2bl" type="CT_Border" minOccurs="0"/>
   </sequence>
</complexType>
```

2.4.64 tcFitText (Fit Text Within Cell)

This element specifies that the contents of the current cell shall have their inter-character spacing increased or reduced as necessary to fit the width of the text extents of the current cell. This setting shall behave identically to placing the contents of this paragraph in a run and using the fitText element (§2.3.2.12), if the width provided on that element matched the width of the current cell.

If this element is omitted, then the text in this cell shall not be fit to the current cell extents.

[*Example*: Consider a 2 row by two column table, in which the contents of the two cells in the first row have both have the fit text property set, as follows:

```
<w:tcPr>
  <w:tcFitText w:val="true"/>
</w:tcPr>
```

The resulting table cells shall have their contents fit to the extents of the parent table cell, as follows:

Sample text in R1C1.	And this table cell instead contains a very very long string of sample text in R2C2.
R2C1	R2C2

end example]

Parent Elements
tcPr (§2.7.5.8); tcPr (§2.4.66); tcPr (§2.7.5.9); tcPr (§2.4.67)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but

Attributes	Description
	this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.4.65 tcMar (Single Table Cell Margins)

This element specifies a set of cell margins for a single table cell in the parent table.

This setting, if present, shall override the table cell margins from the table-level cell margins (§2.4.39).

[*Example*: Consider a table whose first cell is defined to have default cell margins of 0.5 inches for all sides rather then the table defaults, as follows:

R1C1
R2C1

This set of table cell margins is specified using the following WordprocessingML:

```
<w:tcPr>
  <w:tcMar>
    <w:top w:w="720" w:type="dxa"/>
    <w:left w:w="720" w:type="dxa"/>
    <w:bottom w:w="720" w:type="dxa"/>
    <w:right w:w="720" w:type="dxa"/>
  </w:tcMar>
  …
</w:tcPr>
```

The tcMar element as a child of tcPr specifies the set of table cell margins used for the first table cell, in this case, 720 twentieths of a point on all sides. *end example*]

Parent Elements
tcPr (§2.7.5.8); tcPr (§2.4.66); tcPr (§2.7.5.9); tcPr (§2.4.67)

Child Elements	Subclause
bottom (Table Cell Bottom Margin Exception)	§2.4.2
left (Table Cell Left Margin Exception)	§2.4.25
right (Table Cell Right Margin Exception)	§2.4.31
top (Table Cell Top Margin Exception)	§2.4.73

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TcMar">
   <sequence>
      <element name="top" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="left" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="bottom" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="right" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

2.4.66 tcPr (Previous Table Cell Properties)

This element specifies a previous set of table cell properties, the modifications to which shall be attributed to a revision by a particular author and at a particular time. This element contains the table cell property settings which were previously in place before a specific set of revisions by one author. Each unique property is specified by a child element of this element. In any instance where there is a conflict between the table level, table-level exception, or row level properties with a corresponding table cell property, these properties shall overwrite the table or row wide properties.

[*Example*: Consider a basic two row by two column table as follows:

If the cell shading in the first cell is set to red with revision tracking enabled, as follows:

This revision is specified as follows in the associated WordprocessingML:

```
<w:tc>
  <w:tcPr>
    <w:tcW w:w="4788" w:type="dxa"/>
    <w:shd w:val="clear" w:color="auto" w:fill="FF0000"/>
    <w:tcPrChange w:id="2" …>
      <w:tcPr>
        <w:tcW w:w="4788" w:type="dxa"/>
      </w:tcPr>
    </w:tcPrChange>
  </w:tcPr>
  <w:p/>
</w:tc>
```

The tcPr element beneath the tcPrChange element specifies the set of table cell properties which were in place before the current revision to the document. *end example*]

Parent Elements
tcPrChange (§2.13.5.38)

Child Elements	Subclause
cellDel (Table Cell Deletion)	§2.13.5.1
cellIns (Table Cell Insertion)	§2.13.5.2
cellMerge (Vertically Merged/Split Table Cells)	§2.13.5.3
cnfStyle (Table Cell Conditional Formatting)	§2.4.7
gridSpan (Grid Columns Spanned by Current Table Cell)	§2.4.13
hideMark (Ignore End Of Cell Marker In Row Height Calculation)	§2.4.15
hMerge (Horizontally Merged Cell)	§2.4.16
noWrap (Don't Wrap Cell Content)	§2.4.28
shd (Table Cell Shading)	§2.4.33
tcBorders (Table Cell Borders)	§2.4.63
tcFitText (Fit Text Within Cell)	§2.4.64
tcMar (Single Table Cell Margins)	§2.4.65
tcW (Preferred Table Cell Width)	§2.4.68
textDirection (Table Cell Text Flow Direction)	§2.4.69
vAlign (Table Cell Vertical Alignment)	§2.4.80
vMerge (Vertically Merged Cell)	§2.4.81

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TcPrInner">
   <complexContent>
      <extension base="CT_TcPrBase">
         <sequence>
            <group ref="EG_CellMarkupElements" minOccurs="0" maxOccurs="1"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.4.67 tcPr (Table Cell Properties)

This element specifies the set of properties which shall be applied a specific table cell. Each unique property is specified by a child element of this element. In any instance where there is a conflict between the table level, table-level exception, or row level properties with a corresponding table cell property, these properties shall overwrite the table or row wide properties.

[*Example*: Consider a table where the cell width overwrites the table width represented in the following WordprocessingML:

```
<w:tbl>
  <w:tblPr>
    <w:tblCellMar>
      <w:left w:w="0" w:type="dxa"/>
    </w:tblCellMar>
  </w:tblPr>
  <w:tr>
    <w:tc>
      <w:tcPr>
        <w:tcMar>
          <w:left w:w="720" w:type="dxa"/>
        </w:tcMar>
      </w:tcPr>
      …
    </w:tc>
  </w:tr>
</w:tbl>
```

This table cell will have a left cell margin of `720` twentieths of a point (one half inch) as specified in the tcMar element, which overwrites the table level setting of `0` left table cell margin. *end example*]

Parent Elements
tc (§2.4.62)

Child Elements	Subclause
cellDel (Table Cell Deletion)	§2.13.5.1
cellIns (Table Cell Insertion)	§2.13.5.2
cellMerge (Vertically Merged/Split Table Cells)	§2.13.5.3
cnfStyle (Table Cell Conditional Formatting)	§2.4.7
gridSpan (Grid Columns Spanned by Current Table Cell)	§2.4.13
hideMark (Ignore End Of Cell Marker In Row Height Calculation)	§2.4.15
hMerge (Horizontally Merged Cell)	§2.4.16
noWrap (Don't Wrap Cell Content)	§2.4.28
shd (Table Cell Shading)	§2.4.33
tcBorders (Table Cell Borders)	§2.4.63
tcFitText (Fit Text Within Cell)	§2.4.64
tcMar (Single Table Cell Margins)	§2.4.65
tcPrChange (Revision Information for Table Cell Properties)	§2.13.5.38
tcW (Preferred Table Cell Width)	§2.4.68
textDirection (Table Cell Text Flow Direction)	§2.4.69
vAlign (Table Cell Vertical Alignment)	§2.4.80
vMerge (Vertically Merged Cell)	§2.4.81

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TcPr">
   <complexContent>
      <extension base="CT_TcPrInner">
         <sequence>
            <element name="tcPrChange" type="CT_TcPrChange" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.4.68 tcW (Preferred Table Cell Width)

This element specifies the preferred width for this table cell. This preferred width is used as part of the table layout algorithm specified by the tblLayout element (§2.4.49; §2.4.50) - full description of the algorithm in the ST_TblLayout simple type (§2.18.94).

All widths in a table are considered preferred because:

- The table must satisfy the shared columns as specified by the tblGrid element (§2.4.44)
- Two or more widths may have conflicting values for the width of the same grid column
- The table layout algorithm (§2.18.94) may require a preference to be overridden

This value is specified in the units applied via its type attribute. Any width value of type pct for this element shall be calculated relative to the overall width of the table.

If this element is omitted, then the cell width shall be of type auto.

[*Example*: Consider a WordprocessingML table defined as follows:

```
<w:tbl>
  <w:tr>
    <w:tc>
      <w:tcPr>
        <w:tcW w:type="pct" w:w="1667"/>
      </w:tcPr>
      …
    </w:tc>
    <w:tc>
      <w:tcPr>
        <w:tcW w:type="pct" w:w="1667"/>
      </w:tcPr>
      …
    </w:tc>
    <w:tc>
      <w:tcPr>
        <w:tcW w:type="pct" w:w="1667"/>
      </w:tcPr>
      …
    </w:tc>
  </w:tr>
</w:tbl>
```

This table specifies that it has no preferred table width, but each cell shall be exactly 33.3 percent (1667 fiftieths of a percent) of the overall table width. The resulting table would therefore be sized such that all columns are of the width of the maximum column, as follows:

	Hello world	

The text Hello world makes the middle cell larger, and the other two cells are increased in size to maintain the preferred widths of one-third of the overall table width. However, when the middle table cell requires a larger size to accommodate non-breaking text, that preference may be overridden as needed:

	Hello worlddddddddddddddddddddddddddddddddddddd	

In this case, the middle cell's long non breaking string caused the table to reach the text margins on the page, and therefore to override the preferred widths on the empty cells. *end example*]

Parent Elements
tcPr (§2.7.5.8); tcPr (§2.4.66); tcPr (§2.7.5.9); tcPr (§2.4.67)

Attributes	Description
type (Table Width Type)	Specifies the units of the width property being defined by the parent element's w attribute. This property is used to define various properties of a table, including: cell spacing, preferred width, and table margins. If this attribute is omitted, then its value shall be assumed to be dxa (twentieths of a point). [*Example*: Consider a table with a table cell bottom cell spacing with a type of dxa, as follows: `<w:bottom ... w:type="dxa" />` This type shall therefore be used to interpret the width specified in the w attribute as a value in twentieths of a point. *end example*] The possible values for this attribute are defined by the ST_TblWidth simple type (§2.18.97).
w (Table Width Value)	Specifies the value of the width property being defined by the parent element. This property is used to define various properties of a table, including: cell spacing, preferred widths, and table margins. If this attribute is omitted, then its value shall be assumed to be 0. [*Example*: Consider a table with a bottom margin with a width of 302, as follows: `<w:bottom w:w="302" w:type="dxa" />` The value in the w attribute shall therefore be used to determine the width being specified in the context of the units specified in the type attribute. In this case, the type is twentieths of a point (dxa), so the width is 302 twentieths of a point (.2097 inches). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblWidth">
   <attribute name="w" type="ST_DecimalNumber"/>
   <attribute name="type" type="ST_TblWidth"/>
</complexType>
```

2.4.69 textDirection (Table Cell Text Flow Direction)

This element specifies the direction of the text flow for this table cell.

If this element is omitted on a given table cell, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then the table cell shall inherit the text flow settings from the parent section.

[*Example*: Consider a table with one cell in which all the table cell's text flow is top to bottom - right to left:

This table cell would specify this text flow using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    …
    <w:textDirection w:val="tbRl" />
  </w:tcPr>
  …
</w:tc>
```

The textDirection element specifies via the `tbRl` value in the val attribute that the text flow should go top to bottom, then right to left. *end example*]

Parent Elements
tcPr (§2.7.5.8); tcPr (§2.4.66); tcPr (§2.7.5.9); tcPr (§2.4.67)

Attributes	Description

val (Direction of Text Flow)	Specifies the direction of the text flow for this object.
	
	[*Example*: Consider a document with a section in which text shall flow bottom to top vertically, and left to right horizontally. This setting requires the following WordprocessingML:
	
	`<w:sectPr>`
	`…`

Attributes	Description
	<w:textDirection w:val="btLr" /> </w:sectPr> The textDirection element specifies via the btLr value in the val attribute that the text flow shall go bottom to top, and left to right. *end example*] The possible values for this attribute are defined by the ST_TextDirection simple type (§2.18.100).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextDirection">
   <attribute name="val" type="ST_TextDirection" use="required"/>
</complexType>
```

2.4.70 tl2br (Table Cell Top Left to Bottom Right Diagonal Border)

This element specifies the border which shall be displayed on the top left side to bottom right diagonal within the current table cell.

If this element is omitted, then the top left to bottom right diagonal of this table cell shall not have a cell border, and its border may use the table's border settings as appropriate.

[*Example*: Consider a table in which the first cell in the first row specifies a top left to bottom right diagonal cell border as follows:

R1C1	R1C2
R2C1	R2C2

This diagonal cell border is specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    …
    <w:tcBorders>
       <w:tl2br w:val="double" w:sz="4" w:space="0" w:color="FF0000"/>
    </w:tcBorders>
  </w:tcPr>
  <w:p/>
</w:tc>
```

The tl2br element specifies a ½ point border of type double on the table cell's diagonal. *end example*]

Parent Elements
tcBorders (§2.4.63)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or auto to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value auto, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is true, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the

Attributes	Description
	following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object.
	
	When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points.
	
	When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.
	
	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24/>`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*]
	
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).

sz (Border Width)	Specifies the width of the current border. If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value. If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value. [*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:

Attributes	Description
	<w:top w:val="dashed" w:sz="24" …/> <w:left w:val="dashed" w:sz="24" …/> <w:bottom w:val="dashed" w:sz="24" …/> <w:right w:val="dashed" w:sz="24" …/> The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*] The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).
themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border. The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document. [*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup: <w:top … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:bottom … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:left … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:right … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*] The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).
themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is

Attributes	Description
	calculated as follows: $$S_{xml} = 0.4 * 255 = 102 = 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L' = 0.53 * 0.75 = 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border.

Attributes	Description
	The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255 = 153 = 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6) = 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object.

Attributes	Description
	This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <attribute name="val" type="ST_Border" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
   <attribute name="space" type="ST_PointMeasure" use="optional"/>
   <attribute name="shadow" type="ST_OnOff" use="optional"/>
   <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.4.71 top (Table Top Border)

This element specifies the border which shall be displayed at the top of the current table. The appearance of this table border in the document shall be determined by the following settings:

- The display of the border is subject to the conflict resolution algorithm defined by the tcBorders element (§2.4.63) and the tblBorders element (§2.4.37;§2.4.38)

If this element is omitted, then the top of this table shall have the border specified by the associated table style. If no top border is specified in the style hierarchy, then this table shall not have a top border.

[*Example*: Consider a table in which the table properties specifies a top table border, as follows:

R1C1	R1C2
R2C1	R2C2

This top table border is specified using the following WordprocessingML:

```
<w:tbl>
  <w:tblPr>
    <w:tblBorders>
      <w:top w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
w:color="D0D0D0" w:themeColor="accent3" w:themeTint="99"/>
    </w:tblBorders>
  </w:tblPr>
  …
</w:tbl>
```

The top element specifies a three point top table border of type `thinThinThickMediumGap`. *end example*].

Parent Elements
tblBorders (§2.4.37); tblBorders (§2.4.38)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or `auto` to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value `auto`, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />`

Attributes	Description
	This frame's val is `true`, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object.
	
	When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points.
	
	When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.
	
	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24/>`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*]
	
	The possible values for this attribute are defined by the ST_PointMeasure simple type

Attributes	Description
	(§2.18.75).

sz (Border Width)	Specifies the width of the current border.
	
	If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value.
	
	If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value.
	
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`
	
	The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	
	The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).

themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border.
	
	The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document.
	
	[*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup:
	
	`<w:top … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:bottom … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:left … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:right … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	
	The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme

Attributes	Description
	color specified for this document. *end example*] The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).
themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255$$ $$= 102$$ $$= 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L' = 0.53 * 0.75$$ $$= 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute:

Attributes	Description
	<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/> *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255$$ $$= 153$$ $$= 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6)$$ $$= 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we

Attributes	Description
	get 95B3D7. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is single, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <attribute name="val" type="ST_Border" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
   <attribute name="space" type="ST_PointMeasure" use="optional"/>
   <attribute name="shadow" type="ST_OnOff" use="optional"/>
   <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.4.72 top (Table Cell Top Margin Default)

This element specifies the amount of space which shall be left between the top extent of the cell contents and the top border of all table cells within the parent table. This setting may be overridden by the table cell top margin definition specified by the top element contained within the table cell's properties (§2.4.73).

This value is specified in the units applied via its type attribute. Any width value of type pct or auto for this element shall be ignored.

If this element is omitted, then it shall inherit the table cell margin from the associated table style. If a top margin is never specified in the style hierarchy, then this table shall have no top cell padding by default (excepting individual cell overrides).

[*Example*: Consider a two by two table in which the default table cell top margin is specified to be exactly 0.25 inches, as follows (marked with an arrow in the first table cell below):

R1C1	R2C1
R2C1	R2C2

This table property is specified using the following WordprocessingML markup:

```
<w:tbl>
  <w:tblPr>
    <w:tblCellMar>
      <w:top w:w="360" w:type="dxa"/>
    </w:tblCellMar>
  </w:tblPr>
  …
</w:tbl>
```

Every cell in the table has a default cell margin setting it to 360 twentieths of a point. *end example*]

Parent Elements
tblCellMar (§2.4.39); tblCellMar (§2.4.40)

Attributes	Description
type (Table Width Type)	Specifies the units of the width property being defined by the parent element's w attribute. This property is used to define various properties of a table, including: cell spacing, preferred width, and table margins. If this attribute is omitted, then its value shall be assumed to be dxa (twentieths of a point). [*Example*: Consider a table with a table cell bottom cell spacing with a type of dxa, as follows: `<w:bottom ... w:type="dxa" />` This type shall therefore be used to interpret the width specified in the w attribute as a value in twentieths of a point. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_TblWidth simple type (§2.18.97).
w (Table Width Value)	Specifies the value of the width property being defined by the parent element. This property is used to define various properties of a table, including: cell spacing, preferred widths, and table margins. If this attribute is omitted, then its value shall be assumed to be `0`. [*Example*: Consider a table with a bottom margin with a width of `302`, as follows: `<w:bottom w:w="302" w:type="dxa" />` The value in the w attribute shall therefore be used to determine the width being specified in the context of the units specified in the type attribute. In this case, the type is twentieths of a point (`dxa`), so the width is `302` twentieths of a point (.2097 inches). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblWidth">
   <attribute name="w" type="ST_DecimalNumber"/>
   <attribute name="type" type="ST_TblWidth"/>
</complexType>
```

2.4.73 top (Table Cell Top Margin Exception)

This element specifies the amount of space which shall be left between the top extent of the cell contents and the top border of a specific table cell within a table. This setting shall override the table cell top margin definition specified by the top element contained within the table properties (§2.4.72).

This value is specified in the units applied via its type attribute. Any width value of type `pct` or `auto` for this element shall be ignored.

If omitted, then this table cell shall use the bottom cell margins defined in the top element contained within the table properties (§2.4.72).

[*Example*: Consider a table with two cells in which the first table cell's top margin is specified via an exception to be ten times larger (0.2 inches vs. 0.02 inches) than the other table cell margins:

This text fills the extents of the cell.
So does this

The first cell in the table would be specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    <w:tcMar>
      <w:top w:w="288" w:type="dxa" />
    </w:tcMar>
  </w:tcPr>
</w:tc>
```

The first cell in this table has an exception applied to the table cell top cell margin setting it to 288 twentieths of a point (0.2 inches). *end example*]

Parent Elements
tcMar (§2.4.65)

Attributes	Description
type (Table Width Type)	Specifies the units of the width property being defined by the parent element's w attribute. This property is used to define various properties of a table, including: cell spacing, preferred width, and table margins. If this attribute is omitted, then its value shall be assumed to be dxa (twentieths of a point). [*Example*: Consider a table with a table cell bottom cell spacing with a type of dxa, as follows: `<w:bottom ... w:type="dxa" />` This type shall therefore be used to interpret the width specified in the w attribute as a value in twentieths of a point. *end example*] The possible values for this attribute are defined by the ST_TblWidth simple type (§2.18.97).
w (Table Width Value)	Specifies the value of the width property being defined by the parent element. This property is used to define various properties of a table, including: cell spacing, preferred widths, and table margins. If this attribute is omitted, then its value shall be assumed to be 0. [*Example*: Consider a table with a bottom margin with a width of 302, as follows: `<w:bottom w:w="302" w:type="dxa" />` The value in the w attribute shall therefore be used to determine the width being

Attributes	Description
	specified in the context of the units specified in the type attribute. In this case, the type is twentieths of a point (dxa), so the width is 302 twentieths of a point (.2097 inches). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblWidth">
   <attribute name="w" type="ST_DecimalNumber"/>
   <attribute name="type" type="ST_TblWidth"/>
</complexType>
```

2.4.74 top (Table Cell Top Border)

This element specifies the border which shall be displayed at the top of the current table cell. The appearance of this table cell border in the document shall be determined by the following settings:

- If the net tblCellSpacing element value (§2.4.41;§2.4.42;§2.4.43) applied to the cell is non-zero, then the cell border shall always be displayed
- Otherwise, the display of the border is subject to the conflict resolution algorithm defined by the tcBorders element (§2.4.63) and the tblBorders element (§2.4.37;§2.4.38)

If this element is omitted, then the top of this table cell shall not have a cell border, and its border may use the table's border settings as appropriate.

[*Example*: Consider a table in which the first cell in the first row specifies a top cell border , as follows:

R1C1	R1C2
R2C1	R2C2

This top cell border is specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    …
    <w:tcBorders>
      <w:top w:val="thinThickThinSmallGap" w:sz="24" w:space="0"
w:color="FF0000"/>
    </w:tcBorders>
  </w:tcPr>
  <w:p/>
</w:tc>
```

The top element specifies a three point border of type `thinThinThickSmallGap`. *end example*]

Parent Elements
tcBorders (§2.4.63)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or `auto` to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value `auto`, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is `true`, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location.

Attributes	Description
	If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object.
	
	When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points.
	
	When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.
	
	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24/>`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*]
	
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).

sz (Border Width)	Specifies the width of the current border. If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value. If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value

Attributes	Description

	of 31. Any values outside this range may be reassigned to a more appropriate value.
	
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`
	
	The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	
	The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).

themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border.
	
	The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document.
	
	[*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup:
	
	`<w:top … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:bottom … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:left … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:right … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	
	The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*]
	
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).

themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border.

Attributes	Description
	The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255$$ $$= 102$$ $$= 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L' = 0.53 * 0.75$$ $$= 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance.

Attributes	Description
	If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255 \\ = 153 \\ = 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}^{'} = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `4F81BD`. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L^{'} = 0.53 * 0.6 + (1 - .6) \\ = 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get `95B3D7`. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <attribute name="val" type="ST_Border" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
   <attribute name="space" type="ST_PointMeasure" use="optional"/>
   <attribute name="shadow" type="ST_OnOff" use="optional"/>
   <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.4.75 tr (Table Row)

This element specifies a single table row, which contains the table's cells. Table rows in WordprocessingML are analogous to HTML `tr` elements.

A tr element has one formatting child element, trPr (§2.4.78), which defines the properties for the row. Each unique property on the table row is specified by a child element of this element. As well, a table row can contain any valid row-level content, which allows for the use of table cells.

If a table cell does not include at least one child element other than the row properties, then this document shall be considered corrupt.

[*Example*: Consider a table consisting of a single table cell, which contains the text `Hello World`:

Hello World

This table row's content is represented by the following WordprocessingML:

```
<w:tr>
  <w:tc>
    <w:tcPr>
      <w:tcW w:w="0" w:type="auto"/>
    </w:tcPr>
    <w:p>
      <w:r>
        <w:t>Hello, World</w:t>
      </w:r>
    </w:p>
  </w:tc>
</w:tr>
```

The tr element contains a single row-level element - in this case, a table cell. *end example*]

Parent Elements
customXml (§2.5.1.4); sdtContent (§2.5.2.34); tbl (§2.4.36)

Child Elements	Subclause
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Cell-Level Custom XML Element)	§2.5.1.3
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
del (Deleted Run Content)	§2.13.5.12
ins (Inserted Run Content)	§2.13.5.20
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24

Child Elements	Subclause
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
sdt (Cell-Level Structured Document Tag)	§2.5.2.28
tblPrEx (Table-Level Property Exceptions)	§2.4.57
tc (Table Cell)	§2.4.62
trPr (Table Row Properties)	§2.4.78

Attributes	Description
rsidDel (Revision Identifier for Table Row Deletion)	Specifies a unique identifier used to track the editing session when the row was deleted from the main document. All rsid* attributes throughout this document with the same value, if present, must indicate that those regions were modified during the same editing session (time between subsequent save actions). A producer may choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications in this document. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).
rsidR (Revision Identifier for Table Row)	Specifies a unique identifier used to track the editing session when the table row was added to the main document. All rsid* attributes throughout this document with the same value, if present, must indicate that those regions were modified during the same editing session (time between subsequent save actions). A producer may choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications in this document. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).

Attributes	Description
rsidRPr (Revision Identifier for Table Row Glyph Formatting)	Specifies a unique identifier used to track the editing session when the glyph character representing the table row mark was last modified in the main document. All rsid* attributes throughout this document with the same value, if present, must indicate that those regions were modified during the same editing session (time between subsequent save actions). A producer may choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications in this document. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).
rsidTr (Revision Identifier for Table Row Properties)	Specifies a unique identifier used to track the editing session when the table row's properties were last modified in this document. All rsid* attributes throughout this document with the same value, if present, must indicate that those regions were modified during the same editing session (time between subsequent save actions). A producer may choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications in this document. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Row">
    <sequence>
        <element name="tblPrEx" type="CT_TblPrEx" minOccurs="0" maxOccurs="1"/>
        <element name="trPr" type="CT_TrPr" minOccurs="0" maxOccurs="1"/>
        <group ref="EG_ContentCellContent" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="rsidRPr" type="ST_LongHexNumber"/>
    <attribute name="rsidR" type="ST_LongHexNumber"/>
    <attribute name="rsidDel" type="ST_LongHexNumber"/>
    <attribute name="rsidTr" type="ST_LongHexNumber"/>
</complexType>
```

2.4.76 tr2bl (Table Cell Top Right to Bottom Left Diagonal Border)

This element specifies the border which shall be displayed on the top right to bottom left diagonal within the current table cell.

If this element is omitted, then the top right to bottom left diagonal of this table cell shall not have a cell border, and its border may use the table's border settings as appropriate.

[*Example*: Consider a table in which the second cell in the second row specifies a top right to bottom left diagonal cell border as follows:

R1C1	R1C2
R2C1	R2C2

This diagonal cell border is specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    …
    <w:tcBorders>
       <w:tr2bl w:val="double" w:sz="4" w:space="0" w:color="FF0000"/>
    </w:tcBorders>
  </w:tcPr>
  <w:p/>
</w:tc>
```

The tr2bl element specifies a ½ point border of type `double` on the table cell's diagonal. *end example*

Parent Elements
tcBorders (§2.4.63)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or `auto` to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value `auto`, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).

Attributes	Description
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is `true`, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object. When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points. When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points. [*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:

Attributes	Description
	<w:pgBorders w:offsetFrom="page"> <w:bottom … w:space="24/> </w:pgBorders> The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*] The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).
sz (Border Width)	Specifies the width of the current border. If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value. If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value. [*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup: <w:top w:val="dashed" w:sz="24" …/> <w:left w:val="dashed" w:sz="24" …/> <w:bottom w:val="dashed" w:sz="24" …/> <w:right w:val="dashed" w:sz="24" …/> The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*] The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).
themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border. The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document. [*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup: <w:top … w:color="FFA8A0" w:themeColor="accent2"

Attributes	Description
	w:themeTint="99" /> <w:bottom … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:left … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:right … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*] The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).
themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255 = 102 = 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$.

Attributes	Description
	Applying the shade formula with a shade percentage of 75% to the luminance, we get: $L^{'} = 0.53 * 0.75$ $= 0.39698$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $T_{xml} = 0.6 * 255$ $= 153$ $= 99(hex)$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $\mathrm{L}^{'} = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$ • Convert the resultant HSL color to RGB

Attributes	Description
	[*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `4F81BD`. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $L' = 0.53 * 0.6 + (1 - .6)$ $= 0.71$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get `95B3D7`. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <attribute name="val" type="ST_Border" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
   <attribute name="space" type="ST_PointMeasure" use="optional"/>
   <attribute name="shadow" type="ST_OnOff" use="optional"/>
   <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.4.77 trHeight (Table Row Height)

This element specifies the height of the current table row within the current table. This height shall be used to determine the resulting height of the table row, which may be absolute or relative (depending on its attribute values).

If omitted, then the table row shall automatically resize its height to the height required by its contents (the equivalent of an hRule value of auto).

[*Example*: Consider the following WordprocessingML table:

Some text in R1C1.	

Examining the WordprocessingML for this table, the trHeight element is not specified, so the row heights are automatically determined by their contents (in the first row, the text `Some text in R1C1.`). If the first row shall be restricted to 0.1 inches high (144 twentieths of a point) regardless of its contents, that would be specified using the trHeight element as follows:

```
<w:trPr>
  <w:trHeight w:val="144" w:hRule="exact"/>
</w:trPr>
```

The resulting table row would be exactly 144 twentieths of a point high:

end example]

Parent Elements
trPr (§2.7.5.10); trPr (§2.7.5.11); trPr (§2.4.78); trPr (§2.4.79)

Attributes	Description

hRule (Table Row Height Type)	Specifies the meaning of the height specified for this table row.
	
	The meaning of the value of the val attribute is defined based on the value of the hRule attribute for this table row as follows:
	• If the value of hRule is auto, then the table row's height should be automatically determined based on the height of its contents. The h value is ignored.
	• If the value of hRule is atLeast, then the table row's height should be at least the value the h attribute.
	• If the value of hRule is exact, then the table row's height should be exactly the value of the h attribute.
	
	If this attribute is omitted, then its value shall be assumed to be auto.
	
	[*Example*: Consider the following paragraph containing a table row:
	
	`<w:tr>`
	`  <w:trPr>`
	`  <w:trHeight w:val="2189" w:hRule="atLeast"/>`
	`  </w:trPr>`
	`  …`
	`</w:tr>`
	
	The hRule attribute specifies a value of atLeast, so the table row will be a minimum of 2189 twentieths of a point high regardless of its contents, since its val value is 2189 twentieths of a point. *end example*]
	
	The possible values for this attribute are defined by the ST_HeightRule simple type (§2.18.42).

val (Table Row Height)	Specifies the table row's height. This height is expressed in twentieths of a point. If this attribute is omitted, then its value shall be assumed to be 0. The meaning of the value of the val attribute is defined based on the value of the hRule attribute for this table row as follows: • If the value of hRule is auto, then the table row's height should be automatically determined based on the height of its contents. This value is ignored. • If the value of hRule is atLeast, then the table row's height should be at least the value of this attribute. • If the value of hRule is exact, then the table row's height should be exactly the value of this attribute. [*Example*: Consider the following table row:

Attributes	Description
	<w:tr> <w:trPr> <w:trHeight w:val="2189" w:hRule="atLeast"/> </w:trPr> … </w:tr> The val attribute specifies a value of 2189 twentieths of a point, so this table row will be a minimum of 2189 twentieths of a point high regardless of its contents (growing if needed), since its hRule value is set to atLeast. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Height">
   <attribute name="val" type="ST_TwipsMeasure"/>
   <attribute name="hRule" type="ST_HeightRule"/>
</complexType>
```

2.4.78 trPr (Table Row Properties)

This element specifies the set of row-level properties applied to the current table row. Each unique property is specified by a child element of this element. These properties affect the appearance of all cells in the current row within the parent table, but may be overridden by individual cell-level properties, as defined by each property.

[*Example*: Consider the following WordprocessingML table:

The first row shall have a table-row level property which specifies that it shall be restricted to 0.1 inches high (144 twentieths of a point) regardless of its contents, that would be specified using the trHeight element as follows:

```
<w:trPr>
  <w:trHeight w:val="144" w:hRule="exact"/>
  …
</w:trPr>
```

The trPr element specifies the set of table row properties applied to the current table row in the document, in this case a row height requirement using the trHeight element (§2.4.77). *end example*]

Parent Elements
tr (§2.4.75)

Child Elements	Subclause
cantSplit (Table Row Cannot Break Across Pages)	§2.4.6
cnfStyle (Table Row Conditional Formatting)	§2.4.8
del (Deleted Table Row)	§2.13.5.14
divId (Associated HTML div ID)	§2.4.9
gridAfter (Grid Columns After Last Cell)	§2.4.10
gridBefore (Grid Columns Before First Cell)	§2.4.11
hidden (Hidden Table Row Marker)	§2.4.14
ins (Inserted Table Row)	§2.13.5.16
jc (Table Row Alignment)	§2.4.22
tblCellSpacing (Table Row Cell Spacing)	§2.4.42
tblHeader (Repeat Table Row on Every New Page)	§2.4.46
trHeight (Table Row Height)	§2.4.77
trPrChange (Revision Information for Table Row Properties)	§2.13.5.39
wAfter (Preferred Width After Table Row)	§2.4.82
wBefore (Preferred Width Before Table Row)	§2.4.83

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TrPr">
   <complexContent>
      <extension base="CT_TrPrBase">
         <sequence>
            <element name="ins" type="CT_TrackChange" minOccurs="0"/>
            <element name="del" type="CT_TrackChange" minOccurs="0"/>
            <element name="trPrChange" type="CT_TrPrChange" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.4.79 trPr (Previous Table Row Properties)

This element specifies a previous set of table cell properties, the modifications to which shall be attributed to a revision by a particular author and at a particular time. This element contains the table cell property settings which were previously in place before a specific set of revisions by one author. Each unique property is specified by a child element of this element. These properties affect the appearance of all cells in the current row within the parent table, but may be overridden by individual cell-level properties, as defined by each property.

[*Example*: Consider the following WordprocessingML table:

This table has a row height for row one of exactly 0.1". If we change that to a row height of at least 0.1" with revision marking enabled, the table would appear as follows:

Some text in R1C1.	

The resulting WordprocessingML would be:

```
<w:tr>
  <w:trPr>
    <w:trHeight w:val="144"/>
    <w:trPrChange w:id="2" … >
      <w:trPr>
        <w:trHeight w:hRule="exact" w:val="144"/>
      </w:trPr>
    </w:trPrChange>
  </w:trPr>
  …
</w:tr>
```

The trPr element as a child of trPrChange specifies the set of table row properties which were in place before the current revision to the document. *end example*]

Parent Elements
trPrChange (§2.13.5.39)

Child Elements	Subclause
cantSplit (Table Row Cannot Break Across Pages)	§2.4.6
cnfStyle (Table Row Conditional Formatting)	§2.4.8
divId (Associated HTML div ID)	§2.4.9
gridAfter (Grid Columns After Last Cell)	§2.4.10
gridBefore (Grid Columns Before First Cell)	§2.4.11
hidden (Hidden Table Row Marker)	§2.4.14
jc (Table Row Alignment)	§2.4.22

Child Elements	Subclause
tblCellSpacing (Table Row Cell Spacing)	§2.4.42
tblHeader (Repeat Table Row on Every New Page)	§2.4.46
trHeight (Table Row Height)	§2.4.77
wAfter (Preferred Width After Table Row)	§2.4.82
wBefore (Preferred Width Before Table Row)	§2.4.83

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TrPrBase">
   <choice maxOccurs="unbounded">
      <element name="cnfStyle" type="CT_Cnf" minOccurs="0" maxOccurs="1"/>
      <element name="divId" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="gridBefore" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="gridAfter" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="wBefore" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="wAfter" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="cantSplit" type="CT_OnOff" minOccurs="0"/>
      <element name="trHeight" type="CT_Height" minOccurs="0"/>
      <element name="tblHeader" type="CT_OnOff" minOccurs="0"/>
      <element name="tblCellSpacing" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="jc" type="CT_Jc" minOccurs="0" maxOccurs="1"/>
      <element name="hidden" type="CT_OnOff" minOccurs="0"/>
   </choice>
</complexType>
```

2.4.80 vAlign (Table Cell Vertical Alignment)

This element specifies the vertical alignment for text within the current table cell. The vertical alignment of this text is determined by the value of the val attribute.

[*Example*: Consider a table with a single cell with text vertically aligned to the bottom of the cell:

R1C1

This requirement would be specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    <w:vAlign w:val="bottom" />
  </w:tcPr>
  <w:r>
    <w:t>R1C1</w:t>
  </w:r>
```

```
</w:tc>
```

The vAlign element specifies the vertical alignment of the cell contents, in the case, the bottom of the table cell. *end example*]

Parent Elements
tcPr (§2.7.5.8); tcPr (§2.4.66); tcPr (§2.7.5.9); tcPr (§2.4.67)

Attributes	Description
val (Vertical Alignment Setting)	Specifies the vertical alignment for text between the top and bottom margins of the parent container (page or table cell). [*Example*: Consider a region where the text shall be vertically centered in the parent element. This would require a val value of center, in order to specify that all justification vertically shall be centered relative to the parent. For a section, this setting would be specified as follows: `<w:vAlign w:val="center" />` The val attribute of center specifies that the content is centered relative to its container (in this case, the page). *end example*] The possible values for this attribute are defined by the ST_VerticalJc simple type (§2.18.111).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_VerticalJc">
   <attribute name="val" type="ST_VerticalJc" use="required"/>
</complexType>
```

2.4.81 vMerge (Vertically Merged Cell)

This element specifies that this cell is part of a vertically merged set of cells in a table. The val attribute on this element determines how this cell is defined with respect to the previous cell in the table (i.e. does this cell continue the vertical merge or start a new merged group of cells).

If this element is omitted, then this cell shall not be part of any vertically merged grouping of cells, and any vertically merged group of preceding cells shall be closed. If a vertically merged group of cells do not span the same set of grid columns, then this vertical merge is invalid.

[*Example*: Consider a table with three rows and two columns with the last column completely vertically merged:

The second cell in the first row starts a vertical merge that is completed in the last cell, resulting in the following WordprocessingML:

```
<w:tbl>
  …
  <w:tr>
    <w:tc>
      …
    </w:tc>
    <w:tc>
      …
    </w:tc>
    <w:tc>
      <w:tcPr>
        <w:vmerge w:val="restart"/>
      </w:tcPr>
      …
    </w:tc>
  </w:tr>
  <w:tr>
    <w:tc>
      …
    </w:tc>
    <w:tc>
      …
    </w:tc>
    <w:tc>
      <w:tcPr>
        <w:vmerge w:val="continue"/>
      </w:tcPr>
      …
    </w:tc>
  </w:tr>
```

```
<w:tr>
  <w:tc>
    …
  </w:tc>
  <w:tc>
    …
  </w:tc>
  <w:tc>
    <w:tcPr>
      <w:vmerge w:val="continue"/>
    </w:tcPr>
    …
  </w:tc>
</w:tr>
</w:tbl>
```

The vmerge element defines the cells which are vertically merged, and how each cell is merged together. *end example*]

Parent Elements
tcPr (§2.7.5.8); tcPr (§2.4.66); tcPr (§2.7.5.9); tcPr (§2.4.67)

Attributes	Description

val (Vertical Merge Type)	Specifies how the table cell is part of a vertically merged region. This determines whether the cell should join onto an existing grouping of merged cells if any exist, or start a new group of merged cells. Refer to the simple type definition for a full description of each type.
	
	If this attribute is omitted, its value shall be assumed to be `continue`.
	
	[*Example*: Consider a table cell where a vertical cell merge begins . This setting is represented as the following WordprocessingML:
	
	`<w:tcPr>`
	`  <w:vmerge w:val="restart"/>`
	`</w:tcPr>`
	
	The attribute value of `restart` specifies that this element shall start a new vertically merged region in this table. *end example*]
	
	The possible values for this attribute are defined by the ST_Merge simple type (§2.18.64).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_VMerge">
   <attribute name="val" type="ST_Merge"/>
</complexType>
```

2.4.82 wAfter (Preferred Width After Table Row)

This element specifies the preferred width for the total number of grid columns after this table row as specified in the gridAfter element (§2.4.10). This preferred width is used as part of the table layout algorithm specified by the tblLayout element (§2.4.49; §2.4.50) - full description of the algorithm in the ST_TblLayout simple type (§2.18.94).

All widths in a table are considered preferred because:

- The table must satisfy the shared columns as specified by the tblGrid element (§2.4.44)
- Two or more widths may have conflicting values for the width of the same grid column
- The table layout algorithm (§2.18.94) may require a preference to be overridden

This value is specified in the units applied via its type attribute. Any width value of type `pct` for this element shall be calculated relative to the text extents of the page (page width excluding margins).

If this element is omitted, then the cell width shall be of type `auto`.

[*Example*: Consider a WordprocessingML table row defined as follows:

```
<w:tr>
  <w:trPr>
    <w:gridAfter w:val="2"/>
    <w:wAfter w:type="fixed" w:w="1440"/>
  </w:trPr>
  …
</w:tr>
```

This table specifies that it has a preferred table width of `1440` twentieths of a point (one inch) for the two grid columns after the end of that row. The resulting table would therefore be sized such that that set of grid columns is one inch whenever possible, for example the second row in this table:

end example]

Parent Elements
trPr (§2.7.5.10); trPr (§2.7.5.11); trPr (§2.4.78); trPr (§2.4.79)

Attributes	Description
type (Table Width Type)	Specifies the units of the width property being defined by the parent element's w attribute. This property is used to define various properties of a table, including: cell spacing, preferred width, and table margins. If this attribute is omitted, then its value shall be assumed to be dxa (twentieths of a point). [*Example*: Consider a table with a table cell bottom cell spacing with a type of dxa, as follows: `<w:bottom ... w:type="dxa" />` This type shall therefore be used to interpret the width specified in the w attribute as a value in twentieths of a point. *end example*] The possible values for this attribute are defined by the ST_TblWidth simple type (§2.18.97).
w (Table Width Value)	Specifies the value of the width property being defined by the parent element. This property is used to define various properties of a table, including: cell spacing, preferred widths, and table margins. If this attribute is omitted, then its value shall be assumed to be 0. [*Example*: Consider a table with a bottom margin with a width of 302, as follows: `<w:bottom w:w="302" w:type="dxa" />` The value in the w attribute shall therefore be used to determine the width being specified in the context of the units specified in the type attribute. In this case, the type is twentieths of a point (dxa), so the width is 302 twentieths of a point (.2097 inches). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblWidth">
   <attribute name="w" type="ST_DecimalNumber"/>
   <attribute name="type" type="ST_TblWidth"/>
</complexType>
```

2.4.83 wBefore (Preferred Width Before Table Row)

This element specifies the preferred width for the total number of grid columns before this table row as specified in the gridAfter element (§2.4.10). This preferred width is used as part of the table layout algorithm specified by the tblLayout element (§2.4.49; §2.4.50) - full description of the algorithm in the ST_TblLayout simple type (§2.18.94).

All widths in a table are considered preferred because:

- The table must satisfy the shared columns as specified by the tblGrid element (§2.4.44)
- Two or more widths may have conflicting values for the width of the same grid column
- The table layout algorithm (§2.18.94) may require a preference to be overridden

This value is specified in the units applied via its type attribute. Any width value of type `pct` for this element shall be calculated relative to the text extents of the page (page width excluding margins).

If this element is omitted, then the cell width shall be of type `auto`.

[*Example*: Consider a WordprocessingML table row defined as follows:

```
<w:tr>
  <w:trPr>
    <w:gridBefore w:val="1"/>
    <w:wBefore w:type="fixed" w:w="1440"/>
  </w:trPr>
  …
</w:tr>
```

This table specifies that it has a preferred table width of `1440` twentieths of a point (one inch) for the grid column before the start of the row. The resulting table would therefore be sized such that that grid column is one inch whenever possible, for example the second row in this table:

end example]

Parent Elements
trPr (§2.7.5.10); trPr (§2.7.5.11); trPr (§2.4.78); trPr (§2.4.79)

Attributes	Description
type (Table Width	Specifies the units of the width property being defined by the parent element's w

Attributes	Description
Type)	attribute. This property is used to define various properties of a table, including: cell spacing, preferred width, and table margins. If this attribute is omitted, then its value shall be assumed to be dxa (twentieths of a point). [*Example*: Consider a table with a table cell bottom cell spacing with a type of dxa, as follows: `<w:bottom ... w:type="dxa" />` This type shall therefore be used to interpret the width specified in the w attribute as a value in twentieths of a point. *end example*] The possible values for this attribute are defined by the ST_TblWidth simple type (§2.18.97).
w (Table Width Value)	Specifies the value of the width property being defined by the parent element. This property is used to define various properties of a table, including: cell spacing, preferred widths, and table margins. If this attribute is omitted, then its value shall be assumed to be 0. [*Example*: Consider a table with a bottom margin with a width of 302, as follows: `<w:bottom w:w="302" w:type="dxa" />` The value in the w attribute shall therefore be used to determine the width being specified in the context of the units specified in the type attribute. In this case, the type is twentieths of a point (dxa), so the width is 302 twentieths of a point (.2097 inches). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblWidth">
   <attribute name="w" type="ST_DecimalNumber"/>
   <attribute name="type" type="ST_TblWidth"/>
</complexType>
```

2.5 Custom Markup

Within a WordprocessingML document, it is often necessary for specific documents to contain semantic information beyond the presentation information specified by this Office Open XML Standard. [*Example*: An invoice document may wish to specify that a particular sentence of text is a customer name, in order for that

information to be easily extracted from the document without the need to parse the text using regular expression matching or similar. *end example*]

For these scenarios, multiple facilities are provided for the insertion and round-tripping of customer defined semantics within a WordprocessingML document. There are three distinct forms in which customer-defined semantics can be inserted into a WordprocessingML document, each with their own specific intended usage:

- Smart tags
- Custom XML markup
- Structured document tags (content controls)

The elements and attributes which define each of these forms is described in the following clauses.

2.5.1 Custom XML and Smart Tags

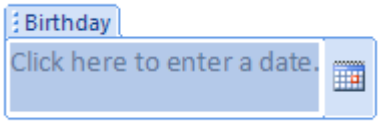
The final type of customer-defined semantics which can be embedded in a WordprocessingML document are structured document tags (SDTs).

As shown above, smart tags and custom XML markup each provide a facility for embedding customer-defined semantics into the document: smart tags, via the ability to provide a basic namespace/name for a run or set of runs within a documents; and custom XML markup, via the ability to tag the document with XML elements and attributes specified by any valid XML Schema file.

However, each of these techniques, while they each provide a way to add the desired semantic information, does not provide a way to affect the presentation or interaction within the document. To bridge these two worlds, structured document tags allow both the specification of customer semantics as well as the ability to influence the presentation of that data in the document.

This means that the customer can define the semantics and context of the tag, but can then use a rich set of pre-defined properties to define its behavior and appearance within the WordprocessingML document's presentation.

[*Example*: Consider a region which should be tagged with the semantic of "birthday", for the user to enter their date or birth into the document. Ideally, this region would also utilize a date picker to allow the user to enter the date from a calendar:



This content would be specified using the following WordprocessingML:

```
<w:sdt>
  <w:sdtPr>
    <w:alias w:val="Birthday"/>
    <w:id w:val="8775518"/>
    <w:placeholder>
      <w:docPart w:val="DefaultPlaceholder_22479095"/>
    </w:placeholder>
    <w:showingPlcHdr/>
    <w:date>
      <w:dateFormat w:val="M/d/yyyy"/>
      <w:lid w:val="EN-US"/>
    </w:date>
  </w:sdtPr>
  <w:sdtContent>
    <w:p>
      <w:r>
        <w:rPr>
          <w:rStyle w:val="PlaceholderText"/>
        </w:rPr>
        <w:t>Click here to enter a date...</w:t>
      </w:r>
    </w:p>
  </w:sdtContent>
</w:sdt>
```

end example]

As shown above, each of the structured document tags in the WordprocessingML file is represented using the sdt element.

Within a structured document tag, there are two child elements which contain the definition and the content of this SDT. The first of these is the sdtPr element, which contains the set of properties specified for this structured document tag. The second is the sdtContent element, which contains all the content which is contained within this structured document tag.

2.5.1.1 attr (Custom XML Attribute)

This element specifies a custom XML attribute which shall be located on the parent custom XML element specified via the customXml element (§2.5.1.3;§2.5.1.4;§2.5.1.5; §2.5.1.6). The attributes on this element shall be used to specify the contents of the custom XML attribute.

[*Example*: Consider a custom XML element with the following properties:

```
<w:customXmlPr>
  <w:attr w:name="companyName" … />
  <w:attr w:name="companySymbol … />
</w:customXmPr>
```

This property bag specifies that the parent custom XML element shall have two attributes associated with it, he first with a name of companyName, and the second with a name of companySymbol. *end example*]

Parent Elements
customXmlPr (§2.5.1.7)

Attributes	Description

name (Name)	Specifies the name of the current custom XML attribute or smart tag property.
	
	[*Example*: Consider a custom XML attribute which shall have a name of companyName. This requirement would be specified using the following WordprocessingML:
	
	`<w:customXmlPr>`
	`  <w:attr w:name="companyName" … />`
	`</w:customXmlPr>`
	
	The name attribute specifies that the name for this property shall be companyName. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

uri (Namespace)	Specifies the namespace URI of the current custom XML attribute or smart tag property.
	
	If this attribute is omitted, the URI shall be assumed to be null (no associated URI).
	
	[*Example*: Consider a smart tag property which shall have a namespace URI of http://schemas.openxmlformats.org/2006/example. This requirement would be specified using the following WordprocessingML:
	
	`<w:smartTagPr>`
	`  <w:attr w:uri="http://schemas.openxmlformats`
	`.org/2006/example" … />`
	`</w:smartTagPr>`
	
	The uri attribute specifies that the namespace for this property shall be http://schemas.openxmlformats.org/2006/example. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

val (Value)	Specifies the value of the current custom XML attribute or smart tag property. [*Example*: Consider a smart tag property which shall have a value of propertyValue. This

Attributes	Description

	requirement would be specified using the following WordprocessingML:
	
	`<w:smartTagPr>`
	`  <w:attr … w:val="propertyValue" />`
	`</w:smartTagPr>`
	
	The val attribute specifies that the value for this property shall be `propertyValue`. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Attr">
   <attribute name="uri" type="ST_String"/>
   <attribute name="name" type="ST_String" use="required"/>
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.5.1.2 attr (Smart Tag Property)

This element specifies a single smart tag property which shall be located on the parent smart tag, specified via the smartTag element (§2.5.1.9). The attributes on this element shall be used to specify the contents of smart tag property.

[*Example*: Consider a smart tag with the following properties:

```
<w:smartTagPr>
  <w:attr w:name="attributeOne" … />
  <w:attr w:name="attributeTwo" … />
</w:smartTagPr>
```

This property bag specifies that the parent smart tag shall have two properties associated with it, the first with a name of `attributeOne`, and the second with a name of `attributeTwo`. *end example*].

Parent Elements
smartTagPr (§2.5.1.10)

Attributes	Description
name (Name)	Specifies the name of the current custom XML attribute or smart tag property. [*Example*: Consider a custom XML attribute which shall have a name of `companyName`. This requirement would be specified using the following WordprocessingML: `<w:customXmlPr>`

Attributes	Description
	<w:attr w:name="companyName" … /> </w:customXmlPr> The name attribute specifies that the name for this property shall be companyName. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).
uri (Namespace)	Specifies the namespace URI of the current custom XML attribute or smart tag property. If this attribute is omitted, the URI shall be assumed to be null (no associated URI). [*Example*: Consider a smart tag property which shall have a namespace URI of http://schemas.openxmlformats.org/2006/example. This requirement would be specified using the following WordprocessingML: <w:smartTagPr> <w:attr w:uri="http://schemas.openxmlformats .org/2006/example" … /> </w:smartTagPr> The uri attribute specifies that the namespace for this property shall be http://schemas.openxmlformats.org/2006/example. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).
val (Value)	Specifies the value of the current custom XML attribute or smart tag property. [*Example*: Consider a smart tag property which shall have a value of propertyValue. This requirement would be specified using the following WordprocessingML: <w:smartTagPr> <w:attr … w:val="propertyValue" /> </w:smartTagPr> The val attribute specifies that the value for this property shall be propertyValue. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Attr">
   <attribute name="uri" type="ST_String"/>
   <attribute name="name" type="ST_String" use="required"/>
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.5.1.3 customXml (Cell-Level Custom XML Element)

This element specifies the presence of a custom XML element around a single table cell. The attributes on this element shall be used to specify the name and namespace URI of the current custom XML element.

[*Example*: Consider a custom XML element with the name company that shall be located around a single table cell in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:tr>
  <w:customXml w:element="company" … >
    <w:tc>
      …
    </w:tc>
  </w:customXml>
  …
</w:tr>
```

The customXml element specifies that the name of the custom XML element is company, and the custom XML element contains a single table cell (it is a cell-level element). *end example*]

Parent Elements
customXml (§2.5.1.3); sdtContent (§2.5.2.33); tr (§2.4.75)

Child Elements	Subclause
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Cell-Level Custom XML Element)	§2.5.1.3
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
customXmlPr (Custom XML Element Properties)	§2.5.1.7
del (Deleted Run Content)	§2.13.5.12

Child Elements	Subclause
ins (Inserted Run Content)	§2.13.5.20
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
sdt (Cell-Level Structured Document Tag)	§2.5.2.28
tc (Table Cell)	§2.4.62

Attributes	Description

element (Custom XML Element Name)	Specifies the name of the current custom XML element or smart tag within the document.
	
	[*Example*: Consider a custom XML element which shall have a name of `companyName`. This requirement would be specified using the following WordprocessingML:
	
	`<w:customXml w:element="companyName" … >`
	`…`
	`</w:customXml>`
	
	The element attribute specifies that the name for this element shall be `companyName`. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

uri (Custom XML Element Namespace)	Specifies the namespace URI of the current custom XML element or smart tag.
	
	If this attribute is omitted, the URI shall be assumed to be null (no associated URI).
	
	[*Example*: Consider a custom XML element which shall have a namespace URI of `urn:customXmlExample`. This requirement would be specified using the following WordprocessingML:
	
	`<w:customXml … w:uri="urn:customXmlExample" />`
	`…`

Attributes	Description
	</w:customXml> The uri attribute specifies that the namespace for this element shall be urn:customXmlExample. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustomXmlCell">
   <sequence>
      <element name="customXmlPr" type="CT_CustomXmlPr" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_ContentCellContent" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="uri" type="ST_String"/>
   <attribute name="element" type="ST_String" use="required"/>
</complexType>
```

2.5.1.4 customXml (Row-Level Custom XML Element)

This element specifies the presence of a custom XML element around a single table row. The attributes on this element shall be used to specify the name and namespace URI of the current custom XML element.

[*Example*: Consider a custom XML element with the name invoiceItem that shall be located around a single table row in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:tbl>
  <w:customXml w:element="invoiceItem" … >
    <w:tr>
      …
    </w:tr>
  </w:customXml>
  …
</w:tbl>
```

The customXml element specifies that the name of the custom XML element is invoiceItem, and the custom XML element contains a single table row (it is a row-level element). *end example*]

Parent Elements
customXml (§2.5.1.4); sdtContent (§2.5.2.34); tbl (§2.4.36)

Child Elements	Subclause
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2

Child Elements	Subclause
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Row-Level Custom XML Element)	§2.5.1.4
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
customXmlPr (Custom XML Element Properties)	§2.5.1.7
del (Deleted Run Content)	§2.13.5.12
ins (Inserted Run Content)	§2.13.5.20
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
sdt (Row-Level Structured Document Tag)	§2.5.2.31
tr (Table Row)	§2.4.75

Attributes	Description
element (Custom XML Element Name)	Specifies the name of the current custom XML element or smart tag within the document. [*Example*: Consider a custom XML element which shall have a name of `companyName`. This requirement would be specified using the following WordprocessingML:

Attributes	Description
	<w:customXml w:element="companyName" … > … </w:customXml> The element attribute specifies that the name for this element shall be companyName. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).
uri (Custom XML Element Namespace)	Specifies the namespace URI of the current custom XML element or smart tag. If this attribute is omitted, the URI shall be assumed to be null (no associated URI). [*Example*: Consider a custom XML element which shall have a namespace URI of urn:customXmlExample. This requirement would be specified using the following WordprocessingML: <w:customXml … w:uri="urn:customXmlExample" /> … </w:customXml> The uri attribute specifies that the namespace for this element shall be urn:customXmlExample. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustomXmlRow">
   <sequence>
      <element name="customXmlPr" type="CT_CustomXmlPr" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_ContentRowContent" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="uri" type="ST_String"/>
   <attribute name="element" type="ST_String" use="required"/>
</complexType>
```

2.5.1.5 customXml (Inline-Level Custom XML Element)

This element specifies the presence of a custom XML element around one or more inline level structures (runs, images, fields, etc.) within a paragraph. The attributes on this element shall be used to specify the name and namespace URI of the current custom XML element.

[*Example*: Consider a custom XML element with the name firstName that shall be located around a two text runs in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:p>
  <w:customXml w:element="firstName" … >
    <w:r>
      …
    </w:r>
    <w:r>
      …
    </w:r>
  </w:customXml>
  …
</w:p>
```

The customXml element specifies that the name of the custom XML element is `firstName`, and the custom XML element contains a two text runs (it is an inline-level element). *end example*]

Parent Elements
customXml (§2.5.1.5); del (§2.13.5.12); fldSimple (§2.16.21); hyperlink (§2.16.24); ins (§2.13.5.20); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); p (§2.3.1.22); sdtContent (§2.5.2.35); smartTag (§2.5.1.9)

Child Elements	Subclause
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Inline-Level Custom XML Element)	§2.5.1.5
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
customXmlPr (Custom XML Element Properties)	§2.5.1.7
del (Deleted Run Content)	§2.13.5.12
fldSimple (Simple Field)	§2.16.21
hyperlink (Hyperlink)	§2.16.24
ins (Inserted Run Content)	§2.13.5.20

Child Elements	Subclause
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
r (Text Run)	§2.3.2.23
sdt (Inline-Level Structured Document Tag)	§2.5.2.29
smartTag (Inline-Level Smart Tag)	§2.5.1.9
subDoc (Anchor for Subdocument Location)	§2.17.2.1

Attributes	Description

element (Element name)	Specifies the name of the current custom XML element or smart tag within the document.
	
	[*Example*: Consider a custom XML element which shall have a name of `companyName`. This requirement would be specified using the following WordprocessingML:
	
	`<w:customXml w:element="companyName" … >`
	`  …`
	`</w:customXml>`
	
	The element attribute specifies that the name for this element shall be `companyName`. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

uri (Custom XML Markup Namespace)	Specifies the namespace URI of the current custom XML element or smart tag. If this attribute is omitted, the URI shall be assumed to be null (no associated URI). [*Example*: Consider a custom XML element which shall have a namespace URI of `urn:customXmlExample`. This requirement would be specified using the following WordprocessingML:

Attributes	Description
	<w:customXml … w:uri="urn:customXmlExample" /> … </w:customXml> The uri attribute specifies that the namespace for this element shall be urn:customXmlExample. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustomXmlRun">
   <sequence>
      <element name="customXmlPr" type="CT_CustomXmlPr" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_PContent" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="uri" type="ST_String"/>
   <attribute name="element" type="ST_String" use="required"/>
</complexType>
```

2.5.1.6 customXml (Block-Level Custom XML Element)

This element specifies the presence of a custom XML element around one or more block level structures (paragraphs, tables, etc.). The attributes on this element shall be used to specify the name and namespace URI of the current custom XML element.

[*Example*: Consider a custom XML element with the name address that shall be located around a single paragraph in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:body>
  <w:customXml w:element="address" … >
    <w:p>
      …
    </w:p>
  </w:customXml>
  …
</w:body>
```

The customXml element specifies that the name of the custom XML element is address, and the custom XML element contains a single paragraph (it is a block-level element). *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.6); docPartBody (§2.12.6); endnote (§2.11.2); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); sdtContent (§2.5.2.32); tc (§2.4.62); txbxContent (§2.17.1.1)

Child Elements	Subclause
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Block-Level Custom XML Element)	§2.5.1.6
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
customXmlPr (Custom XML Element Properties)	§2.5.1.7
del (Deleted Run Content)	§2.13.5.12
ins (Inserted Run Content)	§2.13.5.20
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
p (Paragraph)	§2.3.1.22
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
sdt (Block-Level Structured Document Tag)	§2.5.2.30
tbl (Table)	§2.4.36

Attributes	Description
element (Custom XML Element	Specifies the name of the current custom XML element or smart tag within the document.

Attributes	Description

Name)	[*Example*: Consider a custom XML element which shall have a name of `companyName`. This requirement would be specified using the following WordprocessingML:
	
	`<w:customXml w:element="companyName" … >`
	`…`
	`</w:customXml>`
	
	The element attribute specifies that the name for this element shall be `companyName`. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

uri (Custom XML Element Namespace)	Specifies the namespace URI of the current custom XML element or smart tag.
	
	If this attribute is omitted, the URI shall be assumed to be null (no associated URI).
	
	[*Example*: Consider a custom XML element which shall have a namespace URI of `urn:customXmlExample`. This requirement would be specified using the following WordprocessingML:
	
	`<w:customXml … w:uri="urn:customXmlExample" />`
	`…`
	`</w:customXml>`
	
	The uri attribute specifies that the namespace for this element shall be `urn:customXmlExample`. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustomXmlBlock">
   <sequence>
      <element name="customXmlPr" type="CT_CustomXmlPr" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_ContentBlockContent" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="uri" type="ST_String"/>
   <attribute name="element" type="ST_String" use="required"/>
</complexType>
```

2.5.1.7 customXmlPr (Custom XML Element Properties)

This element specifies the set of properties which shall be applied to the parent custom XML element.

[*Example*: Consider a custom XML element with the following properties specified:

```
<w:customXmlPr>
  <w:placeholder w:val="[Fill in your name]"/>
  <w:attr w:name="status" w:val="draft"/>
</w:customXmlPr>
```

This custom XML element specifies two properties: the presence of placeholder text via the placeholder element (§2.5.1.8) and a single custom XML attribute via the attr element (§2.5.1.1). *end example*]

Parent Elements
customXml (§2.5.1.3); customXml (§2.5.1.6); customXml (§2.5.1.5); customXml (§2.5.1.4)

Child Elements	Subclause
attr (Custom XML Attribute)	§2.5.1.1
placeholder (Custom XML Element Placeholder Text)	§2.5.1.8

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustomXmlPr">
   <sequence>
      <element name="placeholder" type="CT_String" minOccurs="0"/>
      <element name="attr" type="CT_Attr" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

2.5.1.8 placeholder (Custom XML Element Placeholder Text)

This element specifies the placeholder text which shall be displayed in place of this custom XML element when the contents of this custom XML markup are empty (i.e. there are no runs of text within the current custom XML element). If this custom XML element does contain run content, then this text shall not be displayed.

The val attribute stores the string of text which shall be displayed as the placeholder text. This string may be displayed in any font face/size desired by the hosting application.

[*Example*: Consider a custom XML element with the following properties specified:

```
<w:customXmlPr>
  <w:placeholder w:val="[Fill in your name]"/>
  <w:attr w:name="status" w:val="draft"/>
</w:customXmlPr>
```

The placeholder element specifies that this custom XML element shall display the text contents `[Fill in your name]` whenever there is no run content within the parent custom XML element. For example, if the custom XML element was specified as follows:

```
<w:customXml>
  <w:customXmlPr>
    <w:placeholder w:val="[Fill in your name]"/>
  </w:customXmlPr>
  <w:p/>
</w:customXml>
```

This custom XML element has no run content and the placeholder text would be displayed. However, if there is run content, as follows:

```
<w:customXml>
  <w:customXmlPr>
    <w:placeholder w:val="[Fill in your name]"/>
  </w:customXmlPr>
  <w:p>
    <w:r>
      <w:t>Name</w:t>
    </w:r>
  </w:p>
</w:customXml>
```

This custom XML element now contains run content, and the placeholder text shall not be displayed. *end example*]

Parent Elements
customXmlPr (§2.5.1.7)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`

Attributes	Description
	<w:alias w:val="SDT Title Example" /> ... </w:sdtPr> In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.5.1.9 smartTag (Inline-Level Smart Tag)

This element specifies the presence of a smart tag around one or more inline structures (runs, images, fields, etc.) within a paragraph. The attributes on this element shall be used to specify the name and namespace URI of the current smart tag.

[*Example*: Consider a smart tag with the name `firstName` that shall be located around a two text runs in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:p>
  <w:smartTag w:element="firstName" … >
    <w:r>
      …
    </w:r>
    <w:r>
      …
    </w:r>
  </w:smartTag>
  …
</w:p>
```

The smartTag element specifies that the name of the smart tag is `firstName`, and the smart tag contains a two text runs (it is an inline-level smart tag). *end example*]

Parent Elements
customXml (§2.5.1.5); del (§2.13.5.12); fldSimple (§2.16.21); hyperlink (§2.16.24); ins (§2.13.5.20); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); p (§2.3.1.22); sdtContent (§2.5.2.35); smartTag (§2.5.1.9)

Child Elements	Subclause

Child Elements	Subclause
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Inline-Level Custom XML Element)	§2.5.1.5
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
del (Deleted Run Content)	§2.13.5.12
fldSimple (Simple Field)	§2.16.21
hyperlink (Hyperlink)	§2.16.24
ins (Inserted Run Content)	§2.13.5.20
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
r (Text Run)	§2.3.2.23
sdt (Inline-Level Structured Document Tag)	§2.5.2.29
smartTag (Inline-Level Smart Tag)	§2.5.1.9
smartTagPr (Smart Tag Properties)	§2.5.1.10
subDoc (Anchor for Subdocument Location)	§2.17.2.1

Attributes	Description

element (Smart Tag Name)	Specifies the name of the current custom XML element or smart tag within the document.
	
	[*Example*: Consider a custom XML element which shall have a name of `companyName`. This requirement would be specified using the following WordprocessingML:
	
	`<w:customXml w:element="companyName" … >`
	`…`
	`</w:customXml>`
	
	The element attribute specifies that the name for this element shall be `companyName`. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

uri (Smart Tag Namespace)	Specifies the namespace URI of the current custom XML element or smart tag.
	
	If this attribute is omitted, the URI shall be assumed to be null (no associated URI).
	
	[*Example*: Consider a custom XML element which shall have a namespace URI of `urn:customXmlExample`. This requirement would be specified using the following WordprocessingML:
	
	`<w:customXml … w:uri="urn:customXmlExample" />`
	`…`
	`</w:customXml>`
	
	The uri attribute specifies that the namespace for this element shall be `urn:customXmlExample`. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SmartTagRun">
   <sequence>
      <element name="smartTagPr" type="CT_SmartTagPr" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_PContent" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="uri" type="ST_String"/>
   <attribute name="element" type="ST_String" use="required"/>
</complexType>
```

2.5.1.10 smartTagPr (Smart Tag Properties)

This element specifies the set of properties which shall be applied to the parent smart tag.

[*Example*: Consider a smart tag with the following properties specified:

```
<w:smartTagPr>
  <w:attr w:name="date" w:val="01/01/2006"/>
  <w:attr w:name="status" w:val="draft"/>
</w:smartTagPr>
```

This smart tag specifies two properties: the presence of two smart tag properties via the attr element (§2.5.1.2). *end example*]

Parent Elements
smartTag (§2.5.1.9)

Child Elements	Subclause
attr (Smart Tag Property)	§2.5.1.2

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SmartTagPr">
   <sequence>
      <element name="attr" type="CT_Attr" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

2.5.2 Structured Document Tags

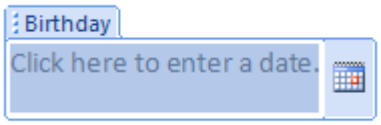
The final type of customer-defined semantics which can be embedded in a WordprocessingML document are structured document tags (SDTs).

As shown above, smart tags and custom XML markup each provide a facility for embedding customer-defined semantics into the document: smart tags, via the ability to provide a basic namespace/name for a run or set of runs within a documents; and custom XML markup, via the ability to tag the document with XML elements and attributes specified by any valid XML Schema file.

However, each of these techniques, while they each provide a way to add the desired semantic information, does not provide a way to affect the presentation or interaction within the document. To bridge these two worlds, structured document tags allow both the specification of customer semantics as well as the ability to influence the presentation of that data in the document.

This means that the customer can define the semantics and context of the tag, but can then use a rich set of pre-defined properties to define its behavior and appearance within the WordprocessingML document's presentation.

[*Example*: Consider a region which should be tagged with the semantic of "birthday", for the user to enter their date or birth into the document. Ideally, this region would also utilize a date picker to allow the user to enter the date from a calendar:



This content would be specified using the following WordprocessingML:

```
<w:sdt>
  <w:sdtPr>
    <w:alias w:val="Birthday"/>
    <w:id w:val="8775518"/>
    <w:placeholder>
      <w:docPart w:val="DefaultPlaceholder_22479095"/>
    </w:placeholder>
    <w:showingPlcHdr/>
    <w:date>
      <w:dateFormat w:val="M/d/yyyy"/>
      <w:lid w:val="EN-US"/>
    </w:date>
  </w:sdtPr>
  <w:sdtContent>
    <w:p>
      <w:r>
        <w:rPr>
          <w:rStyle w:val="PlaceholderText"/>
        </w:rPr>
        <w:t>Click here to enter a date...</w:t>
      </w:r>
    </w:p>
  </w:sdtContent>
</w:sdt>
```

end example]

As shown above, each of the structured document tags in the WordprocessingML file is represented using the sdt element.

Within a structured document tag, there are two child elements which contain the definition and the content of this SDT. The first of these is the sdtPr element, which contains the set of properties specified for this structured document tag. The second is the sdtContent element, which contains all the content which is contained within this structured document tag.

2.5.2.1 alias (Friendly Name)

This element specifies the friendly name associated with the current structured document tag. The string representing the friendly name shall be stored on this element's val attribute.

If this element is omitted, then no friendly name shall be associated with the given structured document tag.

[*Example*: Consider the following properties on a structured document tag:

```
<w:sdtPr>
  <w:alias w:val="Birthday"/>
  …
</w:sdtPr>
```

This set of properties specifies via the alias element that the friendly name for the parent structured document tag shall be `Birthday`. *end example*]

Parent Elements
sdtPr (§2.5.2.37)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.5.2.2 bibliography (Bibliography Structured Document Tag)

This element specifies that the parent structured document tag shall be of type `bibliography`.

This type setting does not require or imply that the contents of the structured document tag shall contain only a field of type `BIBLIOGRAPHY`, it shall only be used to specify that the structured document tag is of this type, which may be used by an application as desired.

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:bibliography/>
  </w:sdtPr>
  …
</w:sdt>
```

The bibliography element in this structured document tag's properties specify that the type of structured document tag is `bibliography`. *end example*]

Parent Elements
sdtPr (§2.5.2.37)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

2.5.2.3 calendar (Date Picker Calendar Type)

This element specifies the type of calendar which shall be displayed for the current date picker structured document tag, if a user interface is present for the structured document tag. The calendar type is stored on this element's val attribute.

If this element is omitted, then the calendar type shall be `gregorian`.

[*Example*: Consider the following structured document tag properties:

```
<w:sdtPr>
  <w:date w:fullDate="01-01-2006T06:30:00Z">
    <w:calendar w:val="gregorian"/>
  </w:date>
</w:sdtPr>
```

The calendar element specifies that the calendar type for a calendar which may be displayed in the document shall be the Gregorian calendar format (`gregorian`). *end example*]

Parent Elements

Parent Elements
date (§2.5.2.7)

Attributes	Description

val (Calendar Type Value)	Specifies a type of calendar, the use of which is determined by the parent XML element.
	
	If this attribute is omitted, then the calendar type shall be `gregorian`.
	
	[*Example*: Consider the following WordprocessingML for a document containing a structured document tag:
	
	`<w:sdtPr>`
	`  <w:date … >`
	`  <w:calendar w:val="japan"/>`
	`  </w:date>`
	`</w:sdtPr>`
	
	The val attribute value of `japan` specifies that the Japanese Emperor Era calendar shall be used; in this case, it is used for the calendar displayed for a date structured document tag. *end example*]
	
	The possible values for this attribute are defined by the ST_CalendarType simple type (§2.18.7).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CalendarType">
   <attribute name="val" type="ST_CalendarType"/>
</complexType>
```

2.5.2.4 citation (Citation Structured Document Tag)

This element specifies that the parent structured document tag shall be of type `citation`.

This type setting does not require or imply that the contents of the structured document tag shall contain only a field of type `CITATION`, it shall only be used to specify that the structured document tag is of this type, which may be used by an application as desired.

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:citation/>
  </w:sdtPr>
  …
</w:sdt>
```

The citation element in this structured document tag's properties specify that the type of structured document tag is `citation`. *end example*]

Parent Elements
sdtPr (§2.5.2.37)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

2.5.2.5 comboBox (Combo Box Structured Document Tag)

This element specifies that the parent structured document tag shall be a combo box when displayed in the document.

This setting specifies that the behavior for this structured document tag shall be as follows:

- The child elements of this element specify choices which shall be displayed in a standard drop-down list format
- Formatting applied to any part of this structured document tag's contents shall apply to its entire contents

As well, the structured document tag must satisfy the following restraints or the document shall be considered invalid:

- The contents shall only be contain a single run (one set of formatting properties)
- The contents shall not contain more than a single paragraph or table cell and shall not contain a table row or table

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:comboBox>
      …
    </w:comboBox>
  </w:sdtPr>
  …
</w:sdt>
```

The comboBox element in this structured document tag's properties specify that the type of structured document tag is a combo box. *end example*]

Parent Elements
sdtPr (§2.5.2.37)

Child Elements	Subclause
listItem (Combo Box List Item)	§2.5.2.20

Attributes	Description

lastValue (Combo Box Last Saved Value)	Specifies the value associated with the current display text for the combo box structured document tag.
	
	If this structured document tag is not mapped to XML using the dataBinding element (§2.5.2.6), then this attribute shall be ignored. If this structured document tag is mapped to XML, it shall be used to determine whether the current display text in the combo box structured document tag shall be retained when the document is opened, as follows:
	• When the XML mapping is created, the content in the custom XML data is retrieved
	• If this content has an associated list item (matching its value attribute), then the corresponding display text shall be displayed in the structured document tag
	• If no list item exists, this content shall be matched against the lastValue attribute value. If the values match, the current display text shall be retained. If the values do not match, the current custom XML data content shall be the new display text (since no match exists in the combo box list items)
	
	[*Example*: Consider a combo box structured document tag defined as follows:
	
	`<w:sdt>`
	`  <w:sdtPr>`
	`  <w:dataBinding … />`
	`  <w:comboBox w:lastValue="2"/>`
	`  </w:sdtPr>`
	`  <w:sdtContent>`
	`  <w:r>`
	`  <w:t>Hello world</w:t>`
	`  </w:r>`
	`  </w:sdtContent>`
	`</w:sdt>`
	
	The current run content of the structured document tag reads `Hello world`. When this document is opened, if the current value of the associated custom XML data is 2, the matching lastValue attribute specifies that the contents of the combo box shall continue to be the current display text of the combo boxeven though there is no listItem whose value is 2 (and normally, the content of the structured document tag would be set to 2. Essentially, this attribute specifies a listItem whose value is 2 and whose displayText is `Hello world` (the current structured document tag contents). *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SdtComboBox">
   <sequence>
      <element name="listItem" type="CT_SdtListItem" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="lastValue" type="ST_String" use="optional"/>
</complexType>
```

2.5.2.6 dataBinding (XML Mapping)

This element specifies the information which shall be used to establish a mapping between the parent structured document tag and an XML element stored within a Custom XML Data part in the current WordprocessingML document.

If this element is omitted, then no XML mapping shall be associated with the current structured document tag. If the parent structured document tag is of type rich text or document part gallery, then this property shall be ignored.

If this element is present and the parent structured document tag is not of a rich text type, then the current value of the structured document tag shall be determined by finding the XML element (if any) which is determined by the attributes on this element. If this information does not result in a valid XML element, then the application may use any algorithm desired to find the closest available match. If this information does result in a valid XML element, then the contents of that element shall be used to replace the current run content within the document.

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    <w:dataBinding w:xpath="/root/name/first" … />
    <w:text/>
  </w:sdtPr>
  <w:sdtContent>
    <w:r>
      <w:t>old text</w:t>
    </w:r>
  </w:sdtContent>
</w:sdt>
```

This structured document tag specifies that it contains only plain text via the text element, and that it shall be mapped to the element in the first Custom XML Data part which contains an element that matches the XPath expression `/root/name/first`. When that element is located, its contents shall replace the existing run content in the document (for example, if its contents are `new text`, then the contents of the run for this structured document tag shall be `new text` when the document is displayed. *end example*]

Parent Elements

Parent Elements
sdtPr (§2.5.2.37)

Attributes	Description

prefixMappings (XML Namespace Prefix Mappings)	Specifies the set of prefix mappings which shall be used to interpret the XPath expression specified on the xpath attribute when the XPath expression is evaluated against the custom XML data parts in the current document.
	
	This attribute's value shall be specified using the following syntax: `xmlns:prefix='namespace'`, where `prefix` is the namespace prefix to be mapped, and `namespace` is the namespace to be mapped to the current prefix. Each prefix mapping shall be delimited by one or more whitespace characters in the attribute's contents.
	
	If this attribute is omitted, then the prefix mappings specified on each of the custom XML data parts itself shall be used to evaluate the given XPath expression.
	
	[*Example*: Consider the following structured document tag properties:
	
	`<w:sdtPr>`
	`  <w:dataBinding w:xpath="//ns0:book"`
	`w:prefixMapping="xmlns:ns0=`
	`'http://example.com/example'"/>`
	`  <w:text/>`
	`</w:sdtPr>`
	
	This structured document tag specifies that it contains an XML mapping, and that mapping's prefixMapping attribute shall signify that the set of namespace prefix mappings to be used to evaluate the xpath attribute value shall be `xmlns:ns0='http://example.com/example'`. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

storeItemID (Custom XML Data Storage ID)	Specifies the custom XML data identifier for the custom XML data part which shall be used to evaluate the given XPath expression. The *custom XML data identifier*, specified using the storeItemID attribute of the dataStoreItem element (§7.5.2.1) on the Custom XML Data Properties part is a string that uniquely identifies a particular custom XML data part in a WordprocessingML document (as multiple parts may have the same namespace for their root element). If specified, then the XPath expression specified on the xpath attribute shall only be evaluated against the custom XML data part whose properties part has a matching custom XML data identifier. If no custom XML data part exists with a matching identifier, then the XML mapping shall not be connected. If omitted, then the XPath expression shall be evaluated against each custom XML data

Attributes	Description

	part in turn until the given XPath expression is resolved to an XML element.
	
	[*Example*: Consider the following structured document tag properties:
	
	`<w:sdtPr>`
	`  <w:dataBinding w:xpath="//ns0:book"`
	`w:storeItemID="testXmlPart"`
	`/>`
	`  <w:text/>`
	`</w:sdtPr>`
	
	This structured document tag specifies that it contains an XML mapping, and that mapping shall only be evaluated against the custom XML part whose identifier is equal to `testXmlPart` (if one exists). *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

xpath (XPath)	Specifies the XPath expression which shall be evaluated to find the custom XML node which is mapped to the parent structured document tag. This XPath expression shall be specified using the syntax defined in the XML Path Language (XPath) Version 1.0 specification (see Annex A for bibliographic reference information).
	
	[*Example*: Consider the following structured document tag properties:
	
	`<w:sdtPr>`
	`  <w:dataBinding w:xpath="//ns0:book"`
	`w:prefixMapping="xmlns:ns0=`
	`'http://example.com/example'"/>`
	`  <w:text/>`
	`</w:sdtPr>`
	
	This structured document tag specifies that it contains an XML mapping, and that mapping's xpath attribute shall signify that the XPath expression to be evaluated shall be `//ns0:book`. Because the prefixMapping attribute is also specified, those prefix mappings shall be used to evaluate this XPath expression. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DataBinding">
   <attribute name="prefixMappings" type="ST_String"/>
   <attribute name="xpath" type="ST_String" use="required"/>
   <attribute name="storeItemID" type="ST_String" use="required"/>
</complexType>
```

2.5.2.7 date (Date Structured Document Tag)

This element specifies that the parent structured document tag shall be a date picker when displayed in the document.

This setting specifies that the behavior for this structured document tag shall be as follows:

- The child elements of this element specify how the dates in this structured document tag shall be stored in any mapped custom XML data and displayed in the document
- Formatting applied to any part of this structured document tag's contents shall apply to its entire contents

As well, the structured document tag must satisfy the following restraints or the document shall be considered invalid:

- The contents shall only be contain a single run (one set of formatting properties)
- The contents shall not contain more than a single paragraph or table cell and shall not contain a table row or table cell

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:date>
      …
    </w:date>
  </w:sdtPr>
  …
</w:sdt>
```

The date element in this structured document tag's properties specifies that the type of structured document tag is a date picker. *end example*]

Parent Elements
sdtPr (§2.5.2.37)

Child Elements	Subclause
calendar (Date Picker Calendar Type)	§2.5.2.3
dateFormat (Date Display Mask)	§2.5.2.8
lid (Date Picker Language ID)	§2.5.2.19
storeMappedDataAs (Custom XML Data Date Storage Format)	§2.5.2.39

Attributes	Description

fullDate (Last Known Date in XML Schema DateTime Format)	Specifies the full date and time last entered into the parent structured document tag using the standard XML Schema DateTime syntax.
	
	[*Note*: This cache is used because the date display mask stored on the dateFormat element (§2.5.2.8) may not contain all of the information about the date, which may be needed if the date display mask is later changed. *end note*]
	
	If this attribute is specified, then the current fullDate attribute shall be used to populate the run content of the parent structured document tag by filtering it through the date display mask specified in the dateFormat element, if one is present.
	
	If this attribute is omitted, then the current display text shall be maintained when the document is displayed.
	
	[*Example*: Consider the following structured document tag properties:
	
	`<w:sdtPr>`
	`…`
	`<w:date w:fullDate="01-01-2006T05:30:00Z">`
	`…`
	`</w:date>`
	`</w:sdtPr>`
	
	The full XML Schema DateTime format for the current structured document tag is specified via the fullDate attribute value as `01-01-2006T05:30:00Z`. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SdtDate">
   <sequence>
      <element name="dateFormat" type="CT_String" minOccurs="0"/>
      <element name="lid" type="CT_Lang" minOccurs="0"/>
      <element name="storeMappedDataAs" type="CT_SdtDateMappingType" minOccurs="0"/>
      <element name="calendar" type="CT_CalendarType" minOccurs="0"/>
   </sequence>
   <attribute name="fullDate" type="ST_DateTime" use="optional"/>
</complexType>
```

2.5.2.8 dateFormat (Date Display Mask)

The element specifies the display format which shall be used to format any date entered into the parent structured document tag in full DateTime format [*Example*: Through a user interface (a date picker), or through custom XML data associated with this structured document tag via the dataBinding element (§2.5.2.6). *end example*] before displaying it in the structured document tag's run content.

If this element is omitted, then the date shall be formatted using the standard date display mask for the language ID specified on the lid element (§2.5.2.19) if present, or the language ID of the run contents otherwise.

The date display mask specified in the val attribute shall be interpreted using the semantics specified in §2.16.4.1 of this Office Open XML Standard.

[*Example*: Consider the following structured document tag properties:

```
<w:sdtPr>
  <w:date w:fullDate="01-01-2006T06:30:00Z">
    <w:dateFormat w:val="MM-YYYY"/>
  </w:date>
</w:sdtPr>
```

The full XML Schema DateTime format for the current structured document tag is specified via the fullDate attribute value as `01-01-2006T06:30:00Z`, and the date display mask is `MM-YYYY`, therefore the resulting date displayed in the document shall be `01-2006` (the month and long year from the full date value, respectively). *end example*]

Parent Elements
date (§2.5.2.7)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the

Attributes	Description
	parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.5.2.9 docPart (Document Part Reference)

This element specifies the name of the document part which shall be displayed in the parent structured document tag when its run contents are empty. If this element is specified, then a document part whose name element (§2.12.12) specifies a name matching the value of this element, and which belongs to the bbPlcHdr type shall be located to be used as the placeholder text for the parent structured document tag.

If no document part is located matching the criteria specified by this element, then five non-breaking spaces shall be used as the default placeholder text.

[*Example*: Consider a structured document tag defined as follows:

```
<w:sdt>
  <w:sdtPr>
    <w:placeholder>
      <w:docPart w:val="DefaultPlaceholder_22610170" />
    </w:placeholder>
    …
  </w:sdtPr>
  <w:sdtContent>
  …
  </w:sdtContent>
</w:sdt>
```

This structured document tag specifies through the docPart element that its placeholder text shall be specified in the document part of type bbPlcHdr whose name is equal to DefaultPlaceholder_22610170. *end example*]

Parent Elements
placeholder (§2.5.2.24)

Attributes	Description
val (String Value)	Specifies that its contents will contain a string. The contents of this string are interpreted based on the context of the parent XML

Attributes	Description

	element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.5.2.10 docPartCategory (Document Part Category Filter)

This element specifies the category of document parts which shall be used as the filter when determining the possible choices of document parts which are displayed for insertion into the parent structured document tag. A document part *category* is a sub-classification within a given document part gallery which may be used to further categorize the parts in a given gallery. [*Example*: Gallery `custom1` may have categories of `Legal Clauses`, `Conformance Clauses`, etc. *end example*]. The category which shall be used as a filter is stored in this element's val attribute.

If this element is omitted, then the parent structured document tag shall display all document parts in the specified gallery regardless their specified category. If this element is present, but no document parts of the specified gallery and category combination are located by the application, then no document parts shall be displayed (i.e. the application shall not fall back to showing document parts in all categories in the specified gallery).

[*Example*: Consider the following properties for a structured document tag:

```
<w:sdtPr>
  <w:docPartList>
    <w:docPartGallery w:val="custom1"/>
    <w:docPartCategory w:val="Legal Clauses"/>
  </w:docPartList>
</w:sdtPr>
```

This structured document tag specifies that it shall present a selection of document parts for insertion via the docPartList element (§2.5.2.12) , and those document parts shall only be the parts which are in the `custom1` gallery via the docPartType element (§2.5.2.11), and within that gallery, only the document parts which are in a category called `Legal Clauses` via this element. *end example*]

Parent Elements
docPartList (§2.5.2.12); docPartObj (§2.5.2.13)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.5.2.11 docPartGallery (Document Part Gallery Filter)

This element specifies the gallery of document parts which shall be used as the filter when determining the possible choices of document parts which are displayed for insertion into the parent structured document tag. A document part *gallery* is a classification of document parts, which may then be subdivided into categories. [*Example*: A gallery with a name of `custom1` may have categories of `Legal Clauses`, `Conformance Clauses`, etc. *end example*]. The gallery which shall be used is stored in this element's val attribute.

If this element is omitted, then the parent structured document tag shall display all document parts in its default gallery. If this element is present, but no document parts of the specified gallery are located by the application, then document parts in the default gallery hall be displayed (i.e. the application shall behave as if the value was omitted).

[*Example*: Consider the following properties for a structured document tag:

```
<w:sdtPr>
  <w:docPartList>
    <w:docPartGallery w:val="custom1"/>
  </w:docPartList>
</w:sdtPr>
```

This structured document tag specifies that it shall present a selection of document parts for insertion via the docPartList element (§2.5.2.12), and those document parts shall only be the parts which are in the `custom1` gallery via this element. *end example*]

Parent Elements
docPartList (§2.5.2.12); docPartObj (§2.5.2.13)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`

Attributes	Description

	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.5.2.12 docPartList (Document Part Gallery Structured Document Tag)

This element specifies that the parent structured document tag shall be of a document part gallery type.

This type setting does not require or imply that the contents of the structured document tag shall contain only the exact contents of a document part of the specified gallery and category which is present on the current machine, it shall only be used to specify that the structured document tag is of this type, which shall be used by an application to present the possible list of choices for insertion into the parent structured document tag.

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:docPartList>
      …
    </w:docPartList>
  </w:sdtPr>
  …
</w:sdt>
```

The docPartList element in this structured document tag's properties specifies that the type of structured document tag is a document part gallery. The child elements shall specify the gallery and category filters for this list, if any. *end example*]

Parent Elements

Parent Elements
sdtPr (§2.5.2.37)

Child Elements	Subclause
docPartCategory (Document Part Category Filter)	§2.5.2.10
docPartGallery (Document Part Gallery Filter)	§2.5.2.11
docPartUnique (Built-In Document Part)	§2.5.2.14

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SdtDocPart">
   <sequence>
      <element name="docPartGallery" type="CT_String" minOccurs="0"/>
      <element name="docPartCategory" type="CT_String" minOccurs="0"/>
      <element name="docPartUnique" type="CT_OnOff" minOccurs="0"/>
   </sequence>
</complexType>
```

2.5.2.13 docPartObj (Built-In Document Part Structured Document Tag)

This element specifies that the parent structured document tag shall be of a document part type.

This type setting does not require or imply that the contents of the structured document tag shall contain only the exact contents of a document part of the specified gallery and category which is present on the current machine, it shall only be used to specify that the structured document tag is of this type, which shall be used by an application to present the possible list of choices for insertion into the parent structured document tag.

This element differs from the docPartList element (§2.5.2.12) in that it may be used to semantically tag a set of block-level objects in a WordprocessingML document without requiring the ability to specify a category and gallery of objects which may be swapped with it via the user interface.

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:docPartObj>
      …
    </w:docPartObj>
  </w:sdtPr>
  …
</w:sdt>
```

The docPartObj element in this structured document tag's properties specify that the type of structured document tag is a document part. The child elements shall specify the gallery and category semantics for this part, if any. *end example*]

Parent Elements
sdtPr (§2.5.2.37)

Child Elements	Subclause
docPartCategory (Document Part Category Filter)	§2.5.2.10
docPartGallery (Document Part Gallery Filter)	§2.5.2.11
docPartUnique (Built-In Document Part)	§2.5.2.14

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SdtDocPart">
   <sequence>
      <element name="docPartGallery" type="CT_String" minOccurs="0"/>
      <element name="docPartCategory" type="CT_String" minOccurs="0"/>
      <element name="docPartUnique" type="CT_OnOff" minOccurs="0"/>
   </sequence>
</complexType>
```

2.5.2.14 docPartUnique (Built-In Document Part)

This element specifies that this structured document tag is being used to encapsulate a built-in document part (i.e. this element appears as a child element of the docPartObj element).

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:docPartObj>
      …
      <w:docPartUnique/>
    </w:docPartObj>
  </w:sdtPr>
  …
</w:sdt>
```

The docPartUnique element in this structured document tag's properties specify that the type of structured document tag is a container for a document part. *end example*]

Parent Elements
docPartList (§2.5.2.12); docPartObj (§2.5.2.13)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element.

Attributes	Description
	A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.5.2.15 dropDownList (Drop-Down List Structured Document Tag)

This element specifies that the parent structured document tag shall be a drop-down list when displayed in the document.

This setting specifies that the behavior for this structured document tag shall be as follows:

- The contents shall not be editable when displayed by a hosting application regardless of the locking settings
- The child elements of this element specify choices which shall be displayed in a standard drop-down list format

As well, the structured document tag must satisfy the following restraints or the document shall be considered invalid:

- The contents shall only be contain a single run (one set of formatting properties)
- The contents shall not contain more than a single paragraph or table cell and shall not contain a table row or table

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:dropDownList>
      …
    </w:dropDownList>
  </w:sdtPr>
  …
</w:sdt>
```

The dropDownList element in this structured document tag's properties specify that the type of structured document tag is a drop-down list. *end example*]

Parent Elements
sdtPr (§2.5.2.37)

Child Elements	Subclause
listItem (Drop-Down List Item)	§2.5.2.21

Attributes	Description

lastValue (Drop-down List Last Saved Value)	Specifies the value associated with the current display text for the drop-down list structured document tag.
	
	If this structured document tag is not mapped to XML using the dataBinding element (§2.5.2.6), then this attribute shall be ignored. If this structured document tag is mapped to XML, it shall be used to determine whether the current display text in the combo box structured document tag shall be retained when the document is opened, as follows:
	• When the XML mapping is created, the content in the custom XML data is retrieved
	• If this content has an associated list item (matching its value attribute), then the corresponding display text shall be displayed in the structured document tag
	• If no list item exists, this content shall be matched against the lastValue attribute value. If the values match, the current display text shall be retained. If the values do not match, the current custom XML data content shall be the new display text (since no match exists in the combo box list items)
	
	[*Example*: Consider a drop-down list structured document tag defined as follows:
	
	`<w:sdt>`
	`  <w:sdtPr>`
	`  <w:dataBinding … />`
	`  <w:dropDownList w:lastValue="2"/>`
	`  </w:sdtPr>`

Attributes	Description
	<w:sdtContent> <w:r> <w:t>Hello world</w:t> </w:r> </w:sdtContent> </w:sdt> The current run content of the structured document tag reads Hello world. When this document is opened, if the current value of the associated custom XML data is 2, the matching lastValue attribute specifies that the contents of the combo box shall continue to be the current display text of the combo boxeven though there is no listItem whose value is 2 (and normally, the content of the structured document tag would be set to 2. Essentially, this attribute specifies a listItem whose value is 2 and whose displayText is Hello world (the current structured document tag contents). *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SdtDropDownList">
   <sequence>
      <element name="listItem" type="CT_SdtListItem" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="lastValue" type="ST_String" use="optional"/>
</complexType>
```

2.5.2.16 equation (Equation Structured Document Tag)

This element specifies that the parent structured document tag shall be of type equation.

This type setting does not require or imply that the contents of the structured document tag shall contain only an equation or associated placeholder text, it shall only be used to specify that the structured document tag is of this type, which may be used by an application as desired.

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:equation/>
  </w:sdtPr>
  …
</w:sdt>
```

The equation element in this structured document tag's properties specify that the type of structured document tag is equation. *end example*]

Parent Elements
sdtPr (§2.5.2.37)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

2.5.2.17 group (Group Structured Document Tag)

This element specifies that the parent structured document tag shall be a restricted grouping when displayed in the document.

This setting specifies that the behavior for this structured document tag shall be as follows:

- The contents of this structured document tag shall not be editable when displayed by a hosting application regardless of the locking settings. This restriction may be superseded by any structured document tag contained within the group, as each structured document tag specifies the locking properties for its own content.

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:group/>
  </w:sdtPr>
  …
</w:sdt>
```

The group element in this structured document tag's properties specify that the type of structured document tag is a restricted group. *end example*]

Parent Elements
sdtPr (§2.5.2.37)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

2.5.2.18 id (Unique ID)

This element specifies a unique numerical ID for the parent structured document tag. This ID shall be persisted through multiple sessions (i.e. shall not be changed once specified).

If multiple structured document tags specify the same decimal number value for the id attribute, then the first structured document tag in the document shall maintain this original ID, and all subsequent structured document tags shall have new identifiers assigned to them when the document is opened.

If this element is omitted, then the parent structured document tag shall have a new unique identifier assigned to it when the document is opened.

[*Example*: Consider the following structured document tag properties:

```
<w:sdtPr>
  <w:id w:val="8775518"/>
  …
</w:sdtPr>
```

This set of properties specifies via the val attribute on the id element that the ID for the parent structured document shall be 8775518 (subject, of course, to the conflict management and resolution discussed above). *end example*]

Parent Elements
sdtPr (§2.5.2.37)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.5.2.19 lid (Date Picker Language ID)

This element specifies the language ID which shall be used for displaying a calendar for the current date picker structured document tag, if a user interface is present for the structured document tag.

If this element is omitted, then the language ID shall be the language ID of the run contents of the parent structured document tag.

[*Example*: Consider the following structured document tag properties:

```
<w:sdtPr>
  <w:date w:fullDate="01-01-2006T06:30:00Z">
    <w:lid w:val="ja-JP"/>
  </w:date>
</w:sdtPr>
```

The calendar language ID for a calendar which may be displayed in the document shall be the default calendar format for the Japanese (Japan) language format (`ja-JP`). *end example*]

Parent Elements
date (§2.5.2.7)

Attributes	Description
val (Language Code)	Specifies an ISO 639-1 letter code or 4 digit hexadecimal code for a specific language. This code is interpreted in the context of the parent XML element. [*Example*: Consider an object which shall specify the English(Canada) language. That object would use the ISO 639-1 letter code of `en-CA` to specify this language. *end example*] The possible values for this attribute are defined by the ST_Lang simple type (§2.18.51).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Lang">
   <attribute name="val" type="ST_Lang" use="required"/>
</complexType>
```

2.5.2.20 listItem (Combo Box List Item)

This element specifies a single list item within the parent combo box structured document tag. Each list item shall be displayed in the list displayed for the parent structured document tag (if a user interface is present).

[*Example*: Consider the following combo box structured document tag:

```
<w:sdt>
  <w:sdtPr>
    <w:comboBox>
      <w:listItem w:displayText="Zero" w:value="0"/>
      <w:listItem w:displayText="One" w:value="1"/>
    </w:comboBox>
  </w:sdtPr>
  …
</w:sdt>
```

Each listItem element within the comboBox element specifies a single list item entry, in this case resulting in two list items within the parent combo box structured document tag. *end example*]

Parent Elements
comboBox (§2.5.2.5)

Attributes	Description

displayText (List Entry Display Text)	Specifies the text to display in the run content (as well as any supplied user interface) in place of the value attribute contents for this drop-down list entry.
	
	This value shall be used as follows:
	• If the parent structured document tag is mapped to a custom XML element, the value in that custom XML element shall be mapped the content of the value attribute, and the resulting displayText attribute value (if one is present) shall be displayed in the run content. If no displayText attribute is present, then the value shall be displayed.
	• If the corresponding entry is selected via a user interface, this value shall be stored in the parent element's run content in the document (this is the value that shall be shown in the document's WordprocessingML content).
	
	If this attribute is omitted, then the content of the value attribute shall be used as the display text for the current list item entry.
	
	[*Example*: Consider the following drop-down list structured document tag:
	
	`<w:sdt>`
	`  <w:sdtPr>`
	`  <w:dropDownList>`
	`  <w:listItem w:displayText="The Letter A" w:value="a"/>`
	`  <w:listItem w:displayText="The Letter B" w:value="b"/>`
	`  </w:dropDownList>`
	`  </w:sdtPr>`
	`  …`
	`</w:sdt>`

Attributes	Description
	The displayText attribute for the first entry is `The Letter A` and the second is `The Letter B`, therefore, these values will be used to determine the display text if the parent structured document tag is mapped to custom XML data in a custom XML data part. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

value (List Entry Value)	Specifies the value for the current list item entry.
	
	This value shall be used as follows:
	• If the parent structured document tag is mapped to a custom XML element, the value in that custom XML element shall be mapped to this value, and the resulting displayText attribute value (if one is present) shall be displayed in the run content. If no displayText attribute is present, then the value shall be displayed.
	• If the corresponding entry is selected via a user interface, this value shall be stored in the parent element's listItem attribute value.
	
	[*Example*: Consider the following combo box structured document tag:
	
	`<w:sdt>`
	`  <w:sdtPr>`
	`  <w:comboBox>`
	`  <w:listItem w:displayText="Zero" w:value="0"/>`
	`  <w:listItem w:displayText="One" w:value="1"/>`
	`  </w:comboBox>`
	`  </w:sdtPr>`
	`  …`
	`</w:sdt>`
	
	The value attribute for the first entry is `0` and the second is `1`, therefore, these values will be used to determine the display text if the parent structured document tag is mapped to custom XML data in a custom XML data part. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SdtListItem">
   <attribute name="displayText" type="ST_String"/>
   <attribute name="value" type="ST_String"/>
</complexType>
```

2.5.2.21 listItem (Drop-Down List Item)

This element specifies a single list item within the parent drop-down list structured document tag. Each list item shall be displayed in the list displayed for the parent structured document tag (if a user interface is present).

[*Example*: Consider the following combo box structured document tag:

```
<w:sdt>
  <w:sdtPr>
    <w:dropDownList>
      <w:listItem w:displayText="The Letter A" w:value="a"/>
      <w:listItem w:displayText="The Letter B" w:value="b"/>
    </w:dropDownList>
  </w:sdtPr>
  …
</w:sdt>
```

Each listItem element within the dropDownList element specifies a single list item entry, in this case resulting in two list items within the parent drop-down list structured document tag. *end example*]

Parent Elements
dropDownList (§2.5.2.15)

Attributes	Description

displayText (List Entry Display Text)	Specifies the text to display in the run content (as well as any supplied user interface) in place of the value attribute contents for this drop-down list entry.
	
	This value shall be used as follows:
	• If the parent structured document tag is mapped to a custom XML element, the value in that custom XML element shall be mapped the content of the value attribute, and the resulting displayText attribute value (if one is present) shall be displayed in the run content. If no displayText attribute is present, then the value shall be displayed.
	• If the corresponding entry is selected via a user interface, this value shall be stored in the parent element's run content in the document (this is the value that shall be shown in the document's WordprocessingML content).
	
	If this attribute is omitted, then the content of the value attribute shall be used as the display text for the current list item entry.
	
	[*Example*: Consider the following drop-down list structured document tag:
	
	`<w:sdt>`
	`  <w:sdtPr>`
	`  <w:dropDownList>`
	`  <w:listItem w:displayText="The Letter A" w:value="a"/>`
	`  <w:listItem w:displayText="The Letter B" w:value="b"/>`
	`  </w:dropDownList>`
	`  </w:sdtPr>`
	`  …`
	`</w:sdt>`

Attributes	Description
	The displayText attribute for the first entry is `The Letter A` and the second is `The Letter B`, therefore, these values will be used to determine the display text if the parent structured document tag is mapped to custom XML data in a custom XML data part. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

value (List Entry Value)	Specifies the value for the current list item entry.
	
	This value shall be used as follows:
	• If the parent structured document tag is mapped to a custom XML element, the value in that custom XML element shall be mapped to this value, and the resulting displayText attribute value (if one is present) shall be displayed in the run content. If no displayText attribute is present, then the value shall be displayed.
	• If the corresponding entry is selected via a user interface, this value shall be stored in the parent element's listItem attribute value.
	
	[*Example*: Consider the following combo box structured document tag:
	
	`<w:sdt>`
	`  <w:sdtPr>`
	`  <w:comboBox>`
	`  <w:listItem w:displayText="Zero" w:value="0"/>`
	`  <w:listItem w:displayText="One" w:value="1"/>`
	`  </w:comboBox>`
	`  </w:sdtPr>`
	`  …`
	`</w:sdt>`
	
	The value attribute for the first entry is `0` and the second is `1`, therefore, these values will be used to determine the display text if the parent structured document tag is mapped to custom XML data in a custom XML data part. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SdtListItem">
   <attribute name="displayText" type="ST_String"/>
   <attribute name="value" type="ST_String"/>
</complexType>
```

2.5.2.22 lock (Locking Setting)

This element specifies the set of behaviors which shall be applied to the contents of the parent structured document tag when the contents of this documents are edited by an application (whether through a user

interface or directly). The type of locking applied to the structured document tag is specified via the value of the associated val attribute.

If this element is omitted, then the locking settings implied for the structured document tag shall be as follows:

- If the structured document tag specifies that it is a group via the group element (§2.5.2.17), then the contents of the structured document tag shall be editable, but the entire tag may be deleted.
- For all other types, no locking settings shall be applied to the structured document tag.

[*Example*: Consider the following plain text structured document tag:

```
<w:sdt>
  <w:sdtPr>
    <w:lock w:val="sdtLocked"/>
    …
    <w:text/>
  </w:sdtPr>
  …
</w:sdt>
```

This plain text structured document tag's properties contain a lock element, specifying locking behaviors for the structured document tag. Since the locking val attribute value is sdtLocked, this locking setting shall specify that the contents of the structured document tag may be edited, but the structured document tag itself shall not be deleted from the document. *end example*]

Parent Elements
sdtPr (§2.5.2.37)

Attributes	Description

val (Locking Type)	Specifies the type of locking which shall be applied to the parent structured document tag.
	
	If this attribute is omitted, this its value shall be assumed to be unlocked (using the defaults stated above).
	
	[*Example*: Consider the following plain text structured document tag properties:
	
	`<w:sdtPr>`
	`  <w:lock w:val="contentLocked"/>`
	`  …`
	`  <w:text/>`
	`</w:sdtPr>`
	
	The val attribute value is contentLocked, therefore this locking setting shall specify that the contents of the structured document tag shall not be edited, but the structured

Attributes	Description
	document tag itself may be deleted from the document. *end example*] The possible values for this attribute are defined by the ST_Lock simple type (§2.18.56).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Lock">
   <attribute name="val" type="ST_Lock"/>
</complexType>
```

2.5.2.23 picture (Picture Structured Document Tag)

This element specifies that the parent structured document tag shall be a picture when displayed in the document.

This setting specifies that the behavior for this structured document tag shall be as follows:

- The contents shall always be restricted to a single picture using either the DrawingML (§5.1) or VML (§6.1) syntax

As well, the structured document tag must satisfy the following restraints or the document shall be considered invalid:

- The contents shall only be a single picture using either the DrawingML (§5.1) or VML (§6.1) syntax
- The contents shall not contain more than a single paragraph or table cell and shall not contain a table row or table

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:picture/>
  </w:sdtPr>
  …
</w:sdt>
```

The text element in this structured document tag's properties specify that the type of structured document tag is a picture. *end example*]

Parent Elements
sdtPr (§2.5.2.37)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

2.5.2.24 placeholder (Structured Document Tag Placeholder Text)

This element specifies the placeholder text which should be displayed when this structured document tag's run contents are empty, the associated mapped XML element is empty as specified via the dataBinding element (§2.5.2.6) or the showingPlcHdr element (§2.5.2.38) is set in the structured document tag's properties. The placeholder text which shall be shown is itself specified via the child element docPart.

If this element is omitted, then five non-breaking spaces shall be used as the default placeholder text for this structured document tag.

[*Example*: Consider a structured document tag defined as follows:

```
<w:sdt>
  <w:sdtPr>
    <w:placeholder>
      <w:docPart w:val="DefaultPlaceholder_22610170" />
    </w:placeholder>
    …
  </w:sdtPr>
  <w:sdtContent>
    …
  </w:sdtContent>
</w:sdt>
```

This structured document tag specifies through the placeholder element that its placeholder text shall be specified in the document part of type `bbPlcHdr` whose name is equal to `DefaultPlaceholder_22610170`. *end example*]

Parent Elements
sdtPr (§2.5.2.37)

Child Elements	Subclause
docPart (Document Part Reference)	§2.5.2.9

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Placeholder">
   <sequence>
      <element name="docPart" type="CT_String"/>
   </sequence>
</complexType>
```

2.5.2.25 richText (Rich Text Structured Document Tag)

This element specifies that the parent structured document tag shall be a rich text box when displayed in the document.

If no type element (the xsd:choice block in the XML Schema fragment for the parent sdtPr element) is specified, then the parent structured document tag shall be of type richText.

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:richText/>
  </w:sdtPr>
  …
</w:sdt>
```

The richText element in this structured document tag's properties specify that the type of structured document tag is a rich text box. *end example*].

Parent Elements
sdtPr (§2.5.2.37)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

2.5.2.26 rPr (Run Properties For Structured Document Tag Contents)

This element specifies the set of run properties which shall be applied to the text entered into the parent structured document tag in replacement of placeholder text. When placeholder text is present in a structured document tag, its formatting is often different than the desired underlying formatting, and this element specifies the formatting which shall be used for non-placeholder text contents when they are initially added to the control.

If this element is not present, the inserted is unformatted, as with any other run of text - it shall not inherit the properties of the placeholder text.

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    <w:placeholder>
      <w:docPart w:val="TestPlaceholderDocPart"/>
    </w:placeholder>
    <w:showingPlcHdr/>
    <w:rPr>
      <w:rStyle w:val="UserName"/>
    </w:rPr>
    …
  </w:sdtPr>
  <w:sdtContent>
    <w:r>
      <w:rPr>
        <w:rStyle w:val="PlaceholderText"/>
      </w:rPr>
      <w:t>[Type Your Name Here]</w:t>
    </w:r>
  </w:sdtContent>
</w:sdt>
```

This structured document tag specifies that its current contents are placeholder text via the showingPlcHdr element (§2.5.2.38), and that text has the `PlaceholderText` character style applied to it.

Now, assume that that style created grey shaded text (typical for placeholder text). This formatting would clearly not be desirable for any text entered into the structured document tag. Therefore, when this text is added, the rPr element in the sdtPr is used to store the formatting on the resulting text.

In this example, the text which initially populates the control shall be formatted with the `UserName` character style. *end example*]

Parent Elements
sdtPr (§2.5.2.37)

Child Elements	Subclause
b (Bold)	§2.3.2.1
bCs (Complex Script Bold)	§2.3.2.2
bdr (Text Border)	§2.3.2.3
caps (Display All Characters As Capital Letters)	§2.3.2.4
color (Run Content Color)	§2.3.2.5
cs (Use Complex Script Formatting on Run)	§2.3.2.6

Child Elements	Subclause
dstrike (Double Strikethrough)	§2.3.2.7
eastAsianLayout (East Asian Typography Settings)	§2.3.2.8
effect (Animated Text Effect)	§2.3.2.9
em (Emphasis Mark)	§2.3.2.10
emboss (Embossing)	§2.3.2.11
fitText (Manual Run Width)	§2.3.2.12
highlight (Text Highlighting)	§2.3.2.13
i (Italics)	§2.3.2.14
iCs (Complex Script Italics)	§2.3.2.15
imprint (Imprinting)	§2.3.2.16
kern (Font Kerning)	§2.3.2.17
lang (Languages for Run Content)	§2.3.2.18
noProof (Do Not Check Spelling or Grammar)	§2.3.2.19
oMath (Office Open XML Math)	§2.3.2.20
outline (Display Character Outline)	§2.3.2.21
position (Vertically Raised or Lowered Text)	§2.3.2.22
rFonts (Run Fonts)	§2.3.2.24
rPrChange (Revision Information for Run Properties)	§2.13.5.32
rStyle (Referenced Character Style)	§2.3.2.27
rtl (Right To Left Text)	§2.3.2.28
shadow (Shadow)	§2.3.2.29
shd (Run Shading)	§2.3.2.30
smallCaps (Small Caps)	§2.3.2.31
snapToGrid (Use Document Grid Settings For Inter-Character Spacing)	§2.3.2.32
spacing (Character Spacing Adjustment)	§2.3.2.33
specVanish (Paragraph Mark Is Always Hidden)	§2.3.2.34
strike (Single Strikethrough)	§2.3.2.35
sz (Font Size)	§2.3.2.36
szCs (Complex Script Font Size)	§2.3.2.37
u (Underline)	§2.3.2.38
vanish (Hidden Text)	§2.3.2.39
vertAlign (Subscript/Superscript Text)	§2.3.2.40
w (Expanded/Compressed Text)	§2.3.2.41
webHidden (Web Hidden Text)	§2.3.2.42

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RPr">
   <sequence>
      <group ref="EG_RPrContent" minOccurs="0"/>
   </sequence>
</complexType>
```

2.5.2.27 rPr (Structured Document Tag End Character Run Properties)

This element specifies the set of run properties which shall be applied to the character present to delimit the end of the structured document tag's contents. When these properties are applied, they shall be applied in addition to the run properties specified for the entire structured document tag via the rPr element (§2.5.2.26) stored in the tag's main property container.

If this element is not present, the inserted closing tag shall be formatting identically to the start tag.

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    <w:placeholder>
      <w:docPart w:val="TestPlaceholderDocPart"/>
    </w:placeholder>
    <w:showingPlcHdr/>
    <w:rPr>
      <w:rStyle w:val="UserName"/>
    </w:rPr>
    …
  </w:sdtPr>
  <w:sdtEndPr>
    <w:rPr>
      <w:b/>
      <w:i/>
    <w:rPr>
  </w:sdtEndPr>
  <w:sdtContent>
    …
  </w:sdtContent>
</w:sdt>
```

The rPr elements under the tag's properties specify that this structured document tag specifies that its start character shall have formatting in the character style `UserName`, and that the end character shall have the formatting in the character style `UserName` as well as bold and italic direct formatting. *end example*]

Parent Elements
sdtEndPr (§2.5.2.36)

Child Elements	Subclause
b (Bold)	§2.3.2.1
bCs (Complex Script Bold)	§2.3.2.2
bdr (Text Border)	§2.3.2.3
caps (Display All Characters As Capital Letters)	§2.3.2.4
color (Run Content Color)	§2.3.2.5
cs (Use Complex Script Formatting on Run)	§2.3.2.6
dstrike (Double Strikethrough)	§2.3.2.7
eastAsianLayout (East Asian Typography Settings)	§2.3.2.8
effect (Animated Text Effect)	§2.3.2.9
em (Emphasis Mark)	§2.3.2.10
emboss (Embossing)	§2.3.2.11
fitText (Manual Run Width)	§2.3.2.12
highlight (Text Highlighting)	§2.3.2.13
i (Italics)	§2.3.2.14
iCs (Complex Script Italics)	§2.3.2.15
imprint (Imprinting)	§2.3.2.16
kern (Font Kerning)	§2.3.2.17
lang (Languages for Run Content)	§2.3.2.18
noProof (Do Not Check Spelling or Grammar)	§2.3.2.19
oMath (Office Open XML Math)	§2.3.2.20
outline (Display Character Outline)	§2.3.2.21
position (Vertically Raised or Lowered Text)	§2.3.2.22
rFonts (Run Fonts)	§2.3.2.24
rPrChange (Revision Information for Run Properties)	§2.13.5.32
rStyle (Referenced Character Style)	§2.3.2.27
rtl (Right To Left Text)	§2.3.2.28
shadow (Shadow)	§2.3.2.29
shd (Run Shading)	§2.3.2.30
smallCaps (Small Caps)	§2.3.2.31
snapToGrid (Use Document Grid Settings For Inter-Character Spacing)	§2.3.2.32
spacing (Character Spacing Adjustment)	§2.3.2.33
specVanish (Paragraph Mark Is Always Hidden)	§2.3.2.34
strike (Single Strikethrough)	§2.3.2.35

Child Elements	Subclause
sz (Font Size)	§2.3.2.36
szCs (Complex Script Font Size)	§2.3.2.37
u (Underline)	§2.3.2.38
vanish (Hidden Text)	§2.3.2.39
vertAlign (Subscript/Superscript Text)	§2.3.2.40
w (Expanded/Compressed Text)	§2.3.2.41
webHidden (Web Hidden Text)	§2.3.2.42

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RPr">
   <sequence>
      <group ref="EG_RPrContent" minOccurs="0"/>
   </sequence>
</complexType>
```

2.5.2.28 sdt (Cell-Level Structured Document Tag)

This element specifies the presence of a structured document tag around a single table cell. The two child elements of this element shall be used to specify the properties and content of the current structured document tag via the sdtPr and sdtContent elements, respectively.

[*Example*: Consider a structured document tag with the friendly name company that shall be located around a single table cell in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:tr>
  <w:sdt>
    <w:sdtPr>
      <w:alias w:val="company"/>
    </w:sdtPr>
    <w:sdtContent>
      <w:tc>
        …
      </w:tc>
    </w:sdtContent>
  </w:sdt>
  …
</w:tr>
```

The sdt element specifies the structured document tag, the child sdtPr element contains the friendly name property set to company, and the sdtContent element contains a single table cell (it is a cell-level structured document tag). *end example*]

Parent Elements
customXml (§2.5.1.3); sdtContent (§2.5.2.33); tr (§2.4.75)

Child Elements	Subclause
sdtContent (Cell-Level Structured Document Tag Content)	§2.5.2.33
sdtEndPr (Structured Document Tag End Character Properties)	§2.5.2.36
sdtPr (Structured Document Tag Properties)	§2.5.2.37

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SdtCell">
   <sequence>
      <element name="sdtPr" type="CT_SdtPr" minOccurs="0" maxOccurs="1"/>
      <element name="sdtEndPr" type="CT_SdtEndPr" minOccurs="0" maxOccurs="1"/>
      <element name="sdtContent" type="CT_SdtContentCell" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

2.5.2.29 sdt (Inline-Level Structured Document Tag)

This element specifies the presence of a structured document tag around one or more inline-level structures (runs, DrawingML objects, fields, etc.) in the current paragraph. The two child elements of this element shall be used to specify the properties and content of the current structured document tag via the sdtPr and sdtContent elements, respectively.

[*Example*: Consider a structured document tag with the friendly name `firstName` that shall be located around two runs in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:p>
  <w:sdt>
    <w:sdtPr>
      <w:alias w:val="firstName"/>
    </w:sdtPr>
    <w:sdtContent>
      <w:r>
        …
      </w:r>
      <w:r>
        …
      </w:r>
    </w:sdtContent>
  </w:sdt>
  …
</w:tr>
```

The sdt element specifies the structured document tag, the child sdtPr element contains the friendly name property set to `firstName`, and the sdtContent element contains two runs (it is an inline-level structured document tag). *end example*]

Parent Elements
customXml (§2.5.1.5); del (§2.13.5.12); fldSimple (§2.16.21); hyperlink (§2.16.24); ins (§2.13.5.20); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); p (§2.3.1.22); sdtContent (§2.5.2.35); smartTag (§2.5.1.9)

Child Elements	Subclause
sdtContent (Inline-Level Structured Document Tag Content)	§2.5.2.35
sdtEndPr (Structured Document Tag End Character Properties)	§2.5.2.36
sdtPr (Structured Document Tag Properties)	§2.5.2.37

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SdtRun">
   <sequence>
      <element name="sdtPr" type="CT_SdtPr" minOccurs="0" maxOccurs="1"/>
      <element name="sdtEndPr" type="CT_SdtEndPr" minOccurs="0" maxOccurs="1"/>
      <element name="sdtContent" type="CT_SdtContentRun" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

2.5.2.30 sdt (Block-Level Structured Document Tag)

This element specifies the presence of a structured document tag around one or more block-level structures (paragraphs, tables, etc.). The two child elements of this element shall be used to specify the properties and content of the current structured document tag via the sdtPr and sdtContent elements, respectively.

[*Example*: Consider a structured document tag with the friendly name `address` that shall be located around a single paragraph in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:body>
  <w:sdt>
    <w:sdtPr>
      <w:alias w:val="address"/>
    </w:sdtPr>
    <w:sdtContent>
      <w:p>
        …
      </w:p>
    </w:sdtContent>
  </w:sdt>
  …
</w:body>
```

The sdt element specifies the structured document tag, the child sdtPr element contains the friendly name property set to address, and the sdtContent element contains a single paragraph (it is a block-level structured document tag). *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.6); docPartBody (§2.12.6); endnote (§2.11.2); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); sdtContent (§2.5.2.32); tc (§2.4.62); txbxContent (§2.17.1.1)

Child Elements	Subclause
sdtContent (Block-Level Structured Document Tag Content)	§2.5.2.32
sdtEndPr (Structured Document Tag End Character Properties)	§2.5.2.36
sdtPr (Structured Document Tag Properties)	§2.5.2.37

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SdtBlock">
   <sequence>
      <element name="sdtPr" type="CT_SdtPr" minOccurs="0" maxOccurs="1"/>
      <element name="sdtEndPr" type="CT_SdtEndPr" minOccurs="0" maxOccurs="1"/>
      <element name="sdtContent" type="CT_SdtContentBlock" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

2.5.2.31 sdt (Row-Level Structured Document Tag)

This element specifies the presence of a structured document tag around a single table row. The two child elements of this element shall be used to specify the properties and content of the current structured document tag via the sdtPr and sdtContent elements, respectively.

[*Example*: Consider a structured document tag with the friendly name `invoiceItem` that shall be located around a single table row in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:tbl>
  <w:sdt>
    <w:sdtPr>
      <w:alias w:val="invoiceItem"/>
    </w:sdtPr>
    <w:sdtContent>
      <w:tr>
        …
      </w:tr>
    </w:sdtContent>
  </w:sdt>
  …
</w:tbl>
```

The sdt element specifies the structured document tag, the child sdtPr element contains the friendly name property set to `invoiceItem`, and the sdtContent element contains a single table row (it is a row-level structured document tag). *end example*]

Parent Elements
customXml (§2.5.1.4); sdtContent (§2.5.2.34); tbl (§2.4.36)

Child Elements	Subclause
sdtContent (Row-Level Structured Document Tag Content)	§2.5.2.34
sdtEndPr (Structured Document Tag End Character Properties)	§2.5.2.36
sdtPr (Structured Document Tag Properties)	§2.5.2.37

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SdtRow">
   <sequence>
      <element name="sdtPr" type="CT_SdtPr" minOccurs="0" maxOccurs="1"/>
      <element name="sdtEndPr" type="CT_SdtEndPr" minOccurs="0" maxOccurs="1"/>
      <element name="sdtContent" type="CT_SdtContentRow" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

2.5.2.32 sdtContent (Block-Level Structured Document Tag Content)

This element specifies the last known contents of a structured document tag around one or more block-level structures (paragraphs, tables, etc.). This element's contents shall be treated as a cache of the contents to be displayed in the structured document tag for the following reasons:

- If the structured document tag specifies an XML mapping via the dataBinding element (§2.5.2.6), changes to the custom XML data part shall be reflected in the structured document tag as needed
- If the contents of the structured document tag are placeholder text via the showingPlcHdr element (§2.5.2.38), then this content may be updated with the placeholder text stored in the Glossary Document part

[*Example*: Consider a structured document tag with the friendly name address that shall be located around a single paragraph in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:body>
  <w:sdt>
    <w:sdtPr>
      <w:alias w:val="address"/>
    </w:sdtPr>
    <w:sdtContent>
      <w:p>
        …
      </w:p>
    </w:sdtContent>
  </w:sdt>
  …
</w:body>
```

The sdtContent element contains a single paragraph (it is a block-level structured document tag content container). *end example*]

Parent Elements
sdt (§2.5.2.30)

Child Elements	Subclause
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Block-Level Custom XML Element)	§2.5.1.6
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7

Child Elements	Subclause
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
del (Deleted Run Content)	§2.13.5.12
ins (Inserted Run Content)	§2.13.5.20
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
p (Paragraph)	§2.3.1.22
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
sdt (Block-Level Structured Document Tag)	§2.5.2.30
tbl (Table)	§2.4.36

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SdtContentBlock">
    <group ref="EG_ContentBlockContent" minOccurs="0" maxOccurs="unbounded"/>
</complexType>
```

2.5.2.33 sdtContent (Cell-Level Structured Document Tag Content)

This element specifies the last known contents of a structured document tag around a single table cell. This element's contents shall be treated as a cache of the contents to be displayed in the structured document tag for the following reasons:

- If the structured document tag specifies an XML mapping via the dataBinding element (§2.5.2.6), changes to the custom XML data part shall be reflected in the structured document tag as needed
- If the contents of the structured document tag are placeholder text via the showingPlcHdr element (§2.5.2.38), then this content may be updated with the placeholder text stored in the Glossary Document part

[*Example*: Consider a structured document tag with the friendly name company that shall be located around a single table cell in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:tr>
  <w:sdt>
    <w:sdtPr>
      <w:alias w:val="company"/>
    </w:sdtPr>
    <w:sdtContent>
      <w:tc>
        …
      </w:tc>
    </w:sdtContent>
  </w:sdt>
  …
</w:tr>
```

The sdtContent element contains a single table cell (it is an cell-level structured document tag content container). *end example*]

Parent Elements
sdt (§2.5.2.28)

Child Elements	Subclause
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Cell-Level Custom XML Element)	§2.5.1.3
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
del (Deleted Run Content)	§2.13.5.12

Child Elements	Subclause
ins (Inserted Run Content)	§2.13.5.20
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
sdt (Cell-Level Structured Document Tag)	§2.5.2.28
tc (Table Cell)	§2.4.62

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SdtContentCell">
    <group ref="EG_ContentCellContent" minOccurs="0" maxOccurs="unbounded"/>
</complexType>
```

2.5.2.34 sdtContent (Row-Level Structured Document Tag Content)

This element specifies the last known contents of a structured document tag around a single table row.

[*Note*: Unlike other types of structured document tags, this type cannot show placeholder text or have mapped XML data, therefore it is never a cache. *end note*]

[*Example*: Consider a structured document tag with the friendly name `invoiceItem` that shall be located around a single table row in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:tbl>
  <w:sdt>
    <w:sdtPr>
      <w:alias w:val="invoiceItem"/>
    </w:sdtPr>
    <w:sdtContent>
      <w:tr>
        …
      </w:tr>
    </w:sdtContent>
  </w:sdt>
  …
</w:tbl>
```

The sdtContent element contains a single table row (it is an row-level structured document tag content container). *end example*]

Parent Elements
sdt (§2.5.2.31)

Child Elements	Subclause
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Row-Level Custom XML Element)	§2.5.1.4
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
del (Deleted Run Content)	§2.13.5.12
ins (Inserted Run Content)	§2.13.5.20
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23

Child Elements	Subclause
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
sdt (Row-Level Structured Document Tag)	§2.5.2.31
tr (Table Row)	§2.4.75

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SdtContentRow">
   <group ref="EG_ContentRowContent" minOccurs="0" maxOccurs="unbounded"/>
</complexType>
```

2.5.2.35 sdtContent (Inline-Level Structured Document Tag Content)

This element specifies the last known contents of a structured document tag around one or more inline-level structures (runs, DrawingML objects, fields, etc.). This element's contents shall be treated as a cache of the contents to be displayed in the structured document tag for the following reasons:

- If the structured document tag specifies an XML mapping via the dataBinding element (§2.5.2.6), changes to the custom XML data part shall be reflected in the structured document tag as needed
- If the contents of the structured document tag are placeholder text via the showingPlcHdr element (§2.5.2.38), then this content may be updated with the placeholder text stored in the Glossary Document part

[*Example*: Consider a structured document tag with the friendly name `firstName` that shall be located around two runs in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:p>
  <w:sdt>
    <w:sdtPr>
      <w:alias w:val="firstName"/>
    </w:sdtPr>
    <w:sdtContent>
      <w:r>
        …
      </w:r>
      <w:r>
        …
      </w:r>
    </w:sdtContent>
  </w:sdt>
  …
</w:p>
```

The sdtContent element contains two adjacent runs (it is an inline-level structured document tag content container). *end example*]

Parent Elements
sdt (§2.5.2.29)

Child Elements	Subclause
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Inline-Level Custom XML Element)	§2.5.1.5
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
del (Deleted Run Content)	§2.13.5.12
fldSimple (Simple Field)	§2.16.21

Child Elements	Subclause
hyperlink (Hyperlink)	§2.16.24
ins (Inserted Run Content)	§2.13.5.20
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
r (Text Run)	§2.3.2.23
sdt (Inline-Level Structured Document Tag)	§2.5.2.29
smartTag (Inline-Level Smart Tag)	§2.5.1.9
subDoc (Anchor for Subdocument Location)	§2.17.2.1

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SdtContentRun">
    <group ref="EG_PContent" minOccurs="0" maxOccurs="unbounded"/>
</complexType>
```

2.5.2.36 sdtEndPr (Structured Document Tag End Character Properties)

This element specifies the properties which shall be applied to the physical character which delimits the end of a structured document tag.

[*Example*: Consider a structured document tag with the following properties specified for the end tag:

```
<w:sdtEndPr>
  <w:rPr>
    …
  </w:rPr>
</w:sdtEndPr>
```

This structured document tag specifies properties for its end character within the sdtEndPr element. *end example*]

Parent Elements

Parent Elements
sdt (§2.5.2.30); sdt (§2.5.2.28); sdt (§2.5.2.29); sdt (§2.5.2.31)

Child Elements	Subclause
rPr (Structured Document Tag End Character Run Properties)	§2.5.2.27

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SdtEndPr">
   <choice maxOccurs="unbounded">
      <element name="rPr" type="CT_RPr" minOccurs="0"/>
   </choice>
</complexType>
```

2.5.2.37 sdtPr (Structured Document Tag Properties)

This element specifies the set of properties which shall be applied to the parent structured document tag.

[*Example*: Consider a structured document tag with the following properties specified:

```
<w:sdtPr>
  <w:alias w:val="Birthday"/>
  <w:id w:val="8775518"/>
  <w:date>
    <w:dateFormat w:val="M/d/yyyy"/>
    <w:lid w:val="EN-US"/>
  </w:date>
</w:sdtPr>
```

This structured document tag specifies three properties: the a friendly name of `Birthday` via the alias element (§2.5.2.1), a unique ID of `8775518` via the id element (§2.5.2.18), and a structured document tag type of date picker via the date element (§2.5.2.7) which itself has a set of date-specific properties. *end example*]

Parent Elements
sdt (§2.5.2.29); sdt (§2.5.2.30); sdt (§2.5.2.28); sdt (§2.5.2.31)

Child Elements	Subclause
alias (Friendly Name)	§2.5.2.1
bibliography (Bibliography Structured Document Tag)	§2.5.2.2
citation (Citation Structured Document Tag)	§2.5.2.4
comboBox (Combo Box Structured Document Tag)	§2.5.2.5
dataBinding (XML Mapping)	§2.5.2.6
date (Date Structured Document Tag)	§2.5.2.7

Child Elements	Subclause
docPartList (Document Part Gallery Structured Document Tag)	§2.5.2.12
docPartObj (Built-In Document Part Structured Document Tag)	§2.5.2.13
dropDownList (Drop-Down List Structured Document Tag)	§2.5.2.15
equation (Equation Structured Document Tag)	§2.5.2.16
group (Group Structured Document Tag)	§2.5.2.17
id (Unique ID)	§2.5.2.18
lock (Locking Setting)	§2.5.2.22
picture (Picture Structured Document Tag)	§2.5.2.23
placeholder (Structured Document Tag Placeholder Text)	§2.5.2.24
richText (Rich Text Structured Document Tag)	§2.5.2.25
rPr (Run Properties For Structured Document Tag Contents)	§2.5.2.26
showingPlcHdr (Current Contents Are Placeholder Text)	§2.5.2.38
tag (Programmatic Tag)	§2.5.2.40
temporary (Remove Structured Document Tag When Contents Are Edited)	§2.5.2.41
text (Plain Text Structured Document Tag)	§2.5.2.42

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SdtPr">
   <choice maxOccurs="unbounded">
      <element name="rPr" type="CT_RPr" minOccurs="0"/>
      <element name="alias" type="CT_String" minOccurs="0"/>
      <element name="lock" type="CT_Lock" minOccurs="0"/>
      <element name="placeholder" type="CT_Placeholder" minOccurs="0"/>
      <element name="showingPlcHdr" type="CT_OnOff" minOccurs="0"/>
      <element name="dataBinding" type="CT_DataBinding" minOccurs="0"/>
      <element name="temporary" type="CT_OnOff" minOccurs="0"/>
      <element name="id" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="tag" type="CT_String" minOccurs="0"/>
      <choice minOccurs="0" maxOccurs="1">
         <element name="equation" type="CT_Empty"/>
         <element name="comboBox" type="CT_SdtComboBox"/>
         <element name="date" type="CT_SdtDate"/>
         <element name="docPartObj" type="CT_SdtDocPart"/>
         <element name="docPartList" type="CT_SdtDocPart"/>
         <element name="dropDownList" type="CT_SdtDropDownList"/>
         <element name="picture" type="CT_Empty"/>
         <element name="richText" type="CT_Empty"/>
         <element name="text" type="CT_SdtText"/>
         <element name="citation" type="CT_Empty"/>
         <element name="group" type="CT_Empty"/>
         <element name="bibliography" type="CT_Empty"/>
      </choice>
   </choice>
</complexType>
```

2.5.2.38 showingPlcHdr (Current Contents Are Placeholder Text)

This element specifies whether the content of the sdtContent element (§2.5.2.32; §2.5.2.33; §2.5.2.34; §2.5.2.35) for the parent structured document tag shall be interpreted to contain placeholder text for this structured document tag (as opposed to regular text contents within the structured document tag). If this element is present and set to `true`, this state shall be resumed (showing placeholder text) upon opening this document.

If this element is omitted, then the structured document tag shall not be interpreted to be showing placeholder text when the document is displayed.

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    <w:showingPlcHdr/>
    …
    <w:richText/>
  </w:sdtPr>
  <w:sdtContent>
    <w:r>
      <w:t>[Type your name here]</w:t>
    </w:r>
 </w:sdtContent>
</w:sdt>
```

This structured document tag has run contents which read `[Type your name here]`, which would typically be interpreted as the current contents of the structured document tag. However, since the showingPlcHdr element has been specified in the structured document tag's properties, this content shall instead be interpreted as the placeholder text for the structured document tag. *end example*]

Parent Elements
sdtPr (§2.5.2.37)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.5.2.39 storeMappedDataAs (Custom XML Data Date Storage Format)

This element specifies the translation which shall be performed on the displayed date in a date picker structured document tag when the current contents are saved into the associated custom XML data via the dataBinding element (§2.5.2.6).

If this element is omitted, then the value of the associated custom XML element shall be placed into the custom XML data part with no translation.

[*Example*: Consider the following date picker structured document tag:

```
<w:sdt>
  <w:sdtPr>
    <w:date w:fullDate="01-01-2006T06:30:00Z">
      <w:storeMappedDateAs w:val="text"/>
      …
    </w:date>
  </w:sdtPr>
  <w:sdtContent>
    <w:r>
      <w:t>January 1</w:t>
    </w:r>
  </w:sdtContent>
</w:sdt>
```

The value of the storeMappedDateAs element's attribute value is `text`, therefore the current run contents shall be sent to the mapped XML element without any translation (in this case, the value shall be `January 1`). *end example*]

Parent Elements
date (§2.5.2.7)

Attributes	Description

val (Date Storage Type)	Specifies the type of date translation which shall be applied to the parent date picker structured document tag.
	
	If this attribute is omitted, this its value shall be assumed to be `text`.
	
	[*Example*: Consider the following date picker structured document tag:
	
	`<w:sdt>`
	`  <w:sdtPr>`
	`  <w:date … >`
	`  <w:storeMappedDateAs w:val="date"/>`
	`  …`

Attributes	Description
	</w:date> </w:sdtPr> <w:sdtContent> <w:r> <w:t>January 1</w:t> </w:r> </w:sdtContent> </w:sdt> The value of the val attribute is text, therefore the current run contents shall be sent to the mapped XML element after being translated into xsd:date format (in this case, the value shall be 01-01-2006). *end example*] The possible values for this attribute are defined by the ST_SdtDateMappingType simple type (§2.18.83).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SdtDateMappingType">
   <attribute name="val" type="ST_SdtDateMappingType"/>
</complexType>
```

2.5.2.40 tag (Programmatic Tag)

This element specifies a programmatic tag associated with the current structured document tag. A *programmatic tag* is an arbitrary string which applications may associate with a structured document tag in order to identify it without providing a visible friendly name. The string representing the programmatic tag shall be stored on this element's val attribute.

If this element is omitted, then no programmatic tag shall be associated with the given structured document tag.

[*Example*: Consider the following properties on a structured document tag:

```
<w:sdtPr>
  <w:tag w:val="Clause_3246"/>
  …
</w:sdtPr>
```

This set of properties specifies via the tag element that the programmatic tag for the parent structured document tag shall be Clause_3246. This information may then be used as needed by applications. *end example*]

Parent Elements
sdtPr (§2.5.2.37)

Attributes	Description

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.5.2.41 temporary (Remove Structured Document Tag When Contents Are Edited)

This element specifies whether the parent structured document tag shall be removed from the WordprocessingML document when the its contents are modified.

[*Note*: This setting is primarily intended for creating structured document tags whose sole purpose is one-time placeholder text, and which should not return once replaced with content. *end note*]

If this element is omitted, then the parent structured document tag shall not be automatically removed when its contents are modified.

[*Example*: Consider the following plain text structured document tag:

```
<w:sdt>
  <w:sdtPr>
    <w:temporary/>
    <w:text/>
  </w:sdtPr>
  …
</w:sdt>
```

This plain text structured document tag's properties contain a temporary element, specifying that the structured document tag itself shall be deleted from the document whenever its contents are first modified. *end example*]

Parent Elements
sdtPr (§2.5.2.37)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.5.2.42 text (Plain Text Structured Document Tag)

This element specifies that the parent structured document tag shall be a plain text box when displayed in the document.

This setting specifies that the behavior for this structured document tag shall be as follows:

- Formatting applied to any part of this structured document tag's contents shall apply to its entire contents

As well, the structured document tag must satisfy the following restraints or the document shall be considered invalid:

- The contents shall only be contain a single run (one set of formatting properties) with exceptions for soft carriage returns via the multiLine attribute on this element
- The contents shall not contain more than a single paragraph or table cell and shall not contain a table row or table

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:text/>
  </w:sdtPr>
  …
</w:sdt>
```

The text element in this structured document tag's properties specify that the type of structured document tag is a plain text box. *end example*]

Parent Elements
sdtPr (§2.5.2.37)

Attributes	Description

multiLine (Allow Soft Line Breaks)	Specifies whether soft line breaks may be added to the contents of this structured document tag when this document is modified. This setting shall not affect the ability of the structured document tag to display existing soft line breaks (which shall be preserved) and shall only affect the ability to add line breaks when the document is modified by an application.
	
	If this attribute is omitted, then the parent plain text structured document control shall not allow soft line breaks to be added to its contents.
	
	[*Example*: Consider the following structured document tag:
	
	`<w:sdt>`
	`  <w:sdtPr>`
	`  …`
	`  <w:text w:multiLine="true"/>`
	`  </w:sdtPr>`
	`  …`

Attributes	Description
	</w:sdt> The multiLine attribute on the text element in this structured document tag's properties specify that an application may allow soft line breaks to be added to the run contents of the structured document tag. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SdtText">
   <attribute name="multiLine" type="ST_OnOff"/>
</complexType>
```

2.6 Sections

WordprocessingML does not natively store the concept of pages, since it is based on paragraphs and runs (which are laid out on to pages by consumers of this content). However, although there is no concept of storing pages in the WordprocessingML format, it is often necessary to store information about a page or group of pages in a document, in order to store information that is to be used to format the pages on which a set of paragraphs will appear. In WordprocessingML, this information is stored via the use of *sections*.

In WordprocessingML, *sections* are groupings of paragraphs that have a specific set of properties used to define the pages on which the text will appear, as well as other section-level (applying to all paragraphs' appearance) properties.

[*Example*: Consider a document with four paragraphs of text that is to be printed on a page in landscape mode, followed by ten paragraphs of text that are to be printed in portrait mode. This requirement implies information about the page(s) used to lay out each grouping of text—the first four paragraphs could require one page, or ten.

Therefore, rather than try to cache knowledge of the number of pages and their properties (which is likely to become invalid if the XML is manipulated by a producer that does not understand page layout), this information is stored by breaking the document into two sections, as follows:

```
<w:p>
  …
</w:p>
<w:p>
  …
</w:p>
<w:p>
  …
</w:p>
```

```
<w:p>
  <w:sectPr>
    …
    (section one properties go here)
    <w:pgSz ... w:orient="landscape" />
    …
  </w:sectPr>
  …
</w:p>
…
<w:p>
  <w:sectPr>
    …
    (section two properties go here)
    <w:pgSz ... w:orient="landscape" />
    …
  </w:sectPr>
  …
</w:p>
```

end example]

2.6.1 bidi (Right to Left Section Layout)

This element specifies that this section shall be presented using a right-to-left page direction. This property only affects section-level properties, and does not affect the layout of text within the contents of this section.

[*Example*: Consider a section with the bidi property set as follows:

```
<w:sectPr>
  ...
  <w:bidi/>
</w:sectPr>
```

This section direction is now right-to-left, which means that all section level properties are displayed right-to-left (e.g., page numbers are displayed on the right of text; columns are populated from right-to-left). However, the layout of text is determined by properties applied at the text level. *end example*]

Parent Elements
sectPr (§2.6.17); sectPr (§2.6.18); sectPr (§2.6.19)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element.

Attributes	Description
	A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

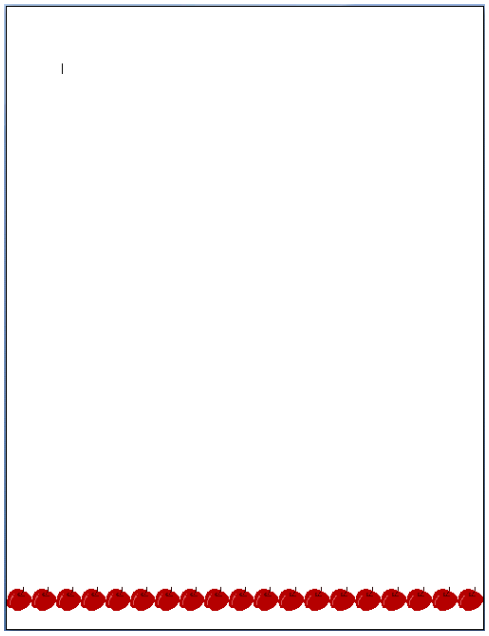
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.6.2 bottom (Bottom Border)

This element specifies the presentation and display of the page border displayed at the bottom of each page in this section.

[*Example*: Consider a section in which all pages shall have a bottom border consisting of a repeated image of an apple, like this:



This border would result in the following WordprocessingML:

```
<w:sectPr>
  …
  <w:pgBorders>
    <w:bottom w:val="apples" .../>
  </w:pgBorders>
  …
</w:sectPr>
```

Because the page only has a border at the bottom, only the bottom element is specified within the set of page borders. *end example*]

When a document has a bottom border that is relative to the page edges (using the offsetFrom attribute on pgBorders), it shall span the bottom edge of the page at the location defined by its properties, stopping when:

- It intersects with the corresponding left or right page border (if one is specified).
- It reaches the edge of the page.

[*Example*: In the example above, no left or right border was specified in the WordprocessingML, so a consumer shall draw the border from one edge of the page to the other. *end example*]

When a document has a bottom border that is relative to the text (using the offsetFrom attribute on pgBorders), it shall span only the necessary width to satisfy the requirement of spanning the width of the text.

Parent Elements
pgBorders (§2.6.10)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or auto to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value auto, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type

Attributes	Description
	(§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is `true`, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object. When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points. When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points. [*Example*: Consider a document with a set of page borders all specified to appear 24

Attributes	Description

	points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24/>`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*]
	
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).

sz (Border Width)	Specifies the width of the current border.
	
	If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value.
	
	If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value.
	
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`
	
	The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	
	The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).

themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border. The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document. [*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup:

Attributes	Description
	<w:top … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:bottom … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:left … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:right … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*] The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).
themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255 = 102 = 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D.

Attributes	Description
	The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $L' = 0.53 * 0.75$ $= 0.39698$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $T_{xml} = 0.6 * 255$ $= 153$ $= 99(hex)$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$ • Convert the resultant HSL color to RGB

Attributes	Description
	[*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `4F81BD`. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $L' = 0.53 * 0.6 + (1 - .6)$ $= 0.71$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get `95B3D7`. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
    <attribute name="val" type="ST_Border" use="required"/>
    <attribute name="color" type="ST_HexColor" use="optional"/>
    <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
    <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
    <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
    <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
    <attribute name="space" type="ST_PointMeasure" use="optional"/>
    <attribute name="shadow" type="ST_OnOff" use="optional"/>
    <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.6.3 col (Single Column Definition)

This element specifies the properties for a single column of text within this section.

[*Example*: Consider a single column with a width of two inches, which also has a one-inch space after the column, resulting in the following WordprocessingML:

```
<w:cols … >
  <w:col w:w="2880" w:space="1440"/>
  …
</w:cols>
```

The resulting column specifies its width of 2,880 twentieths of a point and space following of 1,440 twentieths of a point. *end example*]

The contents of the col element are only used to calculate the number and size of columns if the fixedWidth attribute is set to `false` or omitted.

Parent Elements
cols (§2.6.4)

Attributes	Description
space (Space Before Following Column)	Specifies the spacing (in twentieths of a point) between the current column and the next column. [*Example*: Consider a text column that is to have a one-inch space after it. This text column spacing would therefore be 1x72=144 points wide, which translates to 1,440 twentieths of a point. The resulting WordprocessingML specifies that spacing width in twentieths of a point: `<w:col … w:space="1440"/>` *end example*]

Attributes	Description
	For the last text column in the section, no spacing is allowed after the column, and, if present, any space value is ignored. The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).
w (Column Width)	Specifies the width (in twentieths of a point) of this text column. [*Example*: Consider a text column, which is to be two inches wide. This text column would therefore be 2x72=144 points wide, which translates to 2,880 twentieths of a point. The resulting WordprocessingML specifies that column width in twentieths of a point: `<w:col … w:w="2880"/>` *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Column">
   <attribute name="w" type="ST_TwipsMeasure" use="optional"/>
   <attribute name="space" type="ST_TwipsMeasure" use="optional"/>
</complexType>
```

2.6.4 cols (Column Definitions)

This element specifies the set of columns defined for this section in the document.

[*Example*: Consider a document in which a section defines two columns of 4.16" and 1.83", respectively, resulting in the following WordprocessingML:

```
<w:cols w:equalWidth="0">
  <w:col w:w="2640" w:space="720"/>
  <w:col w:w="6000"/>
</w:cols>
```

The cols element defines the set of columns defined for this section, which because equalWidth is `0`, are defined by the number of col elements contained in the column definition. In this case, the first column is `2,640` twentieths of a point wide (as 2640/1440ths of an inch equals 1.83 inches) with one-half of an inch space after, and the second column is `6,000` twentieths of a point wide (4.16 inches). *end example*]

Based on the presence of the equalWidth attribute, a consumer shall render the columns using:

- If equalWidth is `true`, then the columns are defined using the data stored as attributes of the cols element (defined below).

- If equalWidth is `false`, then the columns are defined using the presence and data on each child col element (§2.6.3).

Parent Elements
sectPr (§2.6.17); sectPr (§2.6.18); sectPr (§2.6.19)

Child Elements	Subclause
col (Single Column Definition)	§2.6.3

Attributes	Description
equalWidth (Equal Column Widths)	Specifies whether all text columns in the current section are of equal width. If this attribute is present and its value is set to `true`, `on`, or `1`, then all columns for this text section are of an equal width and are calculated as follows: • Take width of page (from margin to margin) • Divide by number of columns specified in num attribute • For each column, leave space after as defined in the space attribute • Remaining width of each column is the text column width. If this attribute is present and its value is set to `false`, `off`, or `0`, then all columns for this text section are of different widths and are defined by each col element as follows: • Each col element defines a single column • Each w attribute defines the text column width • Each space attribute defines the space after the text column [*Example*: Consider a section with column information defined as follows: `<w:cols w:num="3" w:space="1440" w:equalWidth="1"> <w:col w:w="2880" w:space="2880" /> <w:col w:w="2880" w:space="1440" /> <w:col w:w="2880" /> </w:cols>` This set of columns has a equalWidth value set to `1`, therefore the col elements are ignored, and there are three equally sized columns (num value of `3`), each with one inch (space value of `1440` twentieths of a point) of space after. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
num (Number of Equal Width Columns)	Specifies the number of text columns in the current section. If all columns are not of equal width (the equalWidth attribute is not set), then this element is ignored, and the number of columns is defined by the number of col elements

Attributes	Description
	defined under the cols element. [*Example*: Consider a section with column information defined as follows: `<w:cols w:num="3" w:space="1440" w:equalWidth="1"> … </w:cols>` This set of columns has a equalWidth value set to 1, therefore there are three equally sized columns, as the num attribute has a value of 3. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
sep (Draw Line Between Columns)	Specifies if a vertical line is drawn between each of the text columns in this section. If set to true, on, or 1, then a vertical line shall be drawn in the center of the spacing between each column in this section. [Example: Consider a section with column information defined as follows: `<w:cols w:sep="1"> … </w:cols>` This set of columns has a sep value set to 1, therefore there shall be a vertical line separating each column in this section. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
space (Spacing Between Equal Width Columns)	Specifies the spacing between text columns in the current section. If all columns are not of equal width (the equalWidth attribute is not set), then this element is ignored, and the spacing after columns is defined by the space attribute on each of the col elements defined under the cols element. [*Example*: Consider a section with column information defined as follows: `<w:cols w:num="3" w:space="1440" w:equalWidth="1"> … </w:cols>` This set of columns has a equalWidth value set to 1, therefore there are three equally sized columns, each with one inch (space value of 1440 twentieths of a point) of space after. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

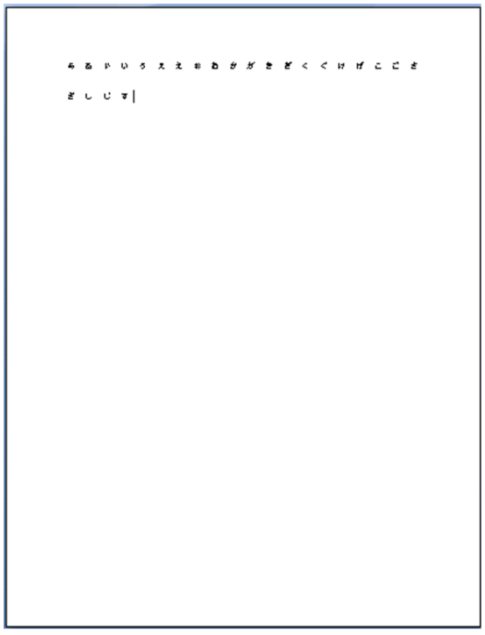
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Columns">
   <sequence minOccurs="0">
      <element name="col" type="CT_Column" maxOccurs="45"/>
   </sequence>
   <attribute name="equalWidth" type="ST_OnOff" use="optional"/>
   <attribute name="space" type="ST_TwipsMeasure" use="optional"/>
   <attribute name="num" type="ST_DecimalNumber" use="optional"/>
   <attribute name="sep" type="ST_OnOff" use="optional"/>
</complexType>
```

2.6.5 docGrid (Document Grid)

This element specifies the settings for the document grid, which enables precise layout of full-width East Asian language characters within a document by specifying the desired number of characters per line and lines per page for all East Asian text content in this section.

[*Example*: Consider a document with the document grid defined to allow 20 characters per line, and 20 lines per page by snapping characters to the grid (type attribute of `snapToChars`) as follows:



As shown, this document allows for only 20 East Asian characters per line by adjusting the inter-character spacing to ensure that there are only 20 characters per line. *end example*]

If Latin text is interspersed on this line, then it is placed across the number of grid units needed to fit the content, but all other grid positions are unaffected.

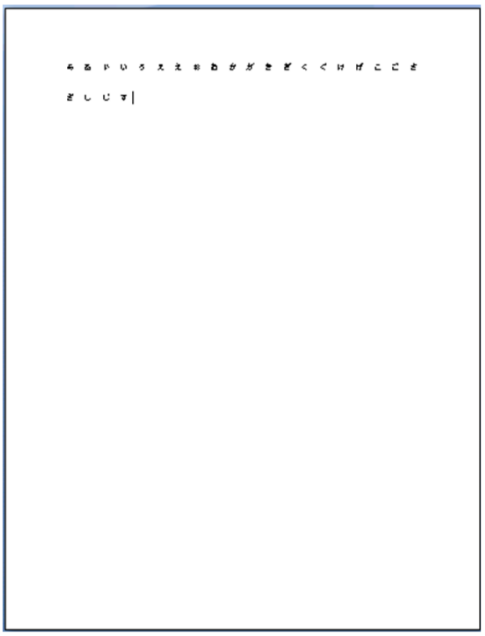
[*Example*: Consider the example above with the addition of the text "Latin text" in English, as follows:

The Latin text spans two grid units, so it is placed in the center of those two units; no other grid positions are affected, so the text on the second line now spans two additional grid units. *end example*]

Parent Elements
sectPr (§2.6.17); sectPr (§2.6.18); sectPr (§2.6.19)

Attributes	Description
charSpace (Document Grid Character Pitch)	Specifies the number of characters to be allowed on the document grid for each line in this section. This attribute's value shall be specified by multiplying the difference between the desired character pitch and the character pitch for that character in the font size of the Normal font by 4096. This value shall then be used to add the character pitch for the specified point size to each character in the section [Note: This results in text in the Normal style having a specific number of characters per line. end note] [*Example*: Consider a section with a Normal font size of 11 points on which a 21 point pitch document grid has been defined.. The resulting WordprocessingML would be defined as follows: `<w:docGrid w:charSize="40960" …/>` The charSpace attribute specifies a value of 40960, which means that the delta between

Attributes	Description
	the character pitch of each character in the grid and the Normal font is 10 points, resulting in a character pitch of 11+10 = 21 points for all characters in this section. *end example*] Individual runs of text can override the line pitch information specified for the document grid by specifying that the run text shall not snap to the document grid via the snapToGrid element (§2.3.2.32). The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
linePitch (Document Grid Line Pitch)	Specifies the number of lines to be allowed on the document grid for the current page assuming all lines have equal line pitch applied to them. This line pitch shall not be added to any line which appears within a table cell unless the adjustLineHeightInTable element (§2.15.3.1) is present in the document's compatibility settings. This attribute is specified in twentieths of a point, and defines the pitch for each line of text on this page such that the desired number of single spaced lines of text fits on the current page. [*Example*: Consider a standard 8.5x11" page on which a 20 character wide, 20 line document grid has been defined. The resulting WordprocessingML would be defined as follows: `<w:docGrid w:linePitch="684" …/>` The linePitch attribute specifies that 34.2 points is to the amount of pitch allowed for each line on this page in order to maintain the specific document grid. *end example*] Individual paragraphs can override the line pitch information specified for the document grid by either: • Specifying an exact line spacing value using the lineRule attribute of value exact on the spacing element (§2.3.1.33). • Specifying that the paragraph text shall not snap to the document grid via the snapToGrid element (§2.3.1.32). The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
type (Document Grid Type)	Specifies the type of the current document grid, which defines the grid behavior. The grid can define a grid which snaps all East Asian characters to grid positions, but leaves Latin text with its default spacing; a grid which adds the specified character pitch to each character on each row; or a grid which affects only the line pitch for the current section. [*Example*: Consider the document discussed above with the document grid defined to

Attributes	Description
	allow 20 characters per line, and 20 lines per page by snapping characters to the grid as follows:  This document has a type attribute of `snapToChars`, which specifies that the grid shall force East Asian characters to fit 20 to a line. *end example*] The possible values for this attribute are defined by the ST_DocGrid simple type (§2.18.18).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DocGrid">
   <attribute name="type" type="ST_DocGrid"/>
   <attribute name="linePitch" type="ST_DecimalNumber"/>
   <attribute name="charSpace" type="ST_DecimalNumber"/>
</complexType>
```

2.6.6 formProt (Only Allow Editing of Form Fields)

This element specifies that the contents of the current section shall be protected such that they cannot be edited by a user (if the consumer is displaying the document and allowing the user to make modification) except for the text contained in any form field or embedded control that is part of the current section.

[*Example*: Consider a section consisting of three paragraphs of text and a single text form field, located at the beginning of the second paragraph. If this section is protected in this manner, a user would only be permitted to edit the contents of the text form field, and all other contents would be locked to prevent user edits. *end example*]

The enforcement of this property is determined by the documentProtection element (§2.15.1.28), as it is possible to specify protection without turning it on.

Parent Elements
sectPr (§2.6.17); sectPr (§2.6.18); sectPr (§2.6.19)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

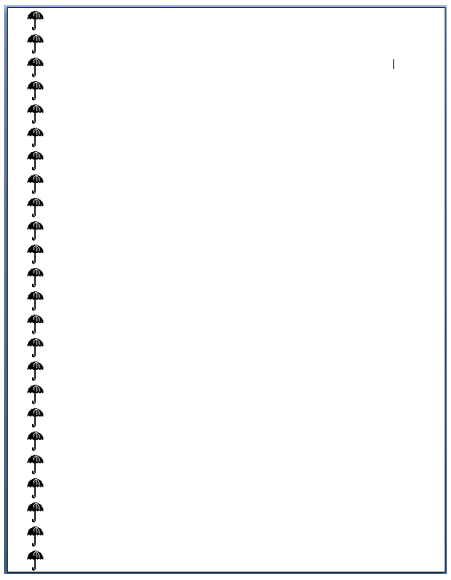
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.6.7 left (Left Border)

This element specifies the presentation and display of the page border displayed at the left of each page in this section.

[*Example*: Consider a section in which all pages have a left border consisting of a repeated image of an umbrella, like this:



This border would result in the following WordprocessingML:

```
<w:sectPr>
  …
  <w:pgBorders>
    <w:left w:val="seattle" …/>
  </w:pgBorders>
  …
</w:sectPr>
```

Because the page only has a border at the left, only the left element is specified within the set of page borders. *end example*]

When a document has a left border that is relative to the page edges (using the offsetFrom attribute value of page on pgBorders), it shall span the left edge of the page at the location defined by its properties, stopping when:

- It intersects with the corresponding top or bottom page border (if one is specified).
- It reaches the edge of the page.

[*Example*: In the example above, no top or bottom border was specified in the WordprocessingML, so a consumer shall draw the border from one edge of the page to the other. *end example*]

When a document has a left border that is relative to the text (using the offsetFrom attribute value of `text` on pgBorders), it shall span only the necessary width to satisfy the requirement of spanning the width of the text.

Parent Elements

Parent Elements
pgBorders (§2.6.10)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or `auto` to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value `auto`, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is `true`, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect.

Attributes	Description
	[*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object.
	
	When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points.
	
	When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.
	
	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24/>`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*]
	
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).

sz (Border Width)	Specifies the width of the current border. If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value. If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value.

Attributes	Description

	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`
	
	The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	
	The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).

themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border.
	
	The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document.
	
	[*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup:
	
	`<w:top … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:bottom … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:left … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:right … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	
	The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*]
	
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).

themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border.

Attributes	Description
	[*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255 \\ = 102 \\ = 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}^{'} = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L^{'} = 0.53 * 0.75 \\ = 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from

Attributes	Description
	the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255 = 153 = 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `4F81BD`. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6) = 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get `95B3D7`. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).

Attributes	Description
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <attribute name="val" type="ST_Border" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
   <attribute name="space" type="ST_PointMeasure" use="optional"/>
   <attribute name="shadow" type="ST_OnOff" use="optional"/>
   <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.6.8 lnNumType (Line Numbering Settings)

This element specifies the settings for line numbering to be displayed before each column of text in this section in the document.

[*Example*: Consider the line numbering used on each page of this document, which specifies: line numbering for each line, restarting at one at the top of each new page. This line-numbering scheme would be defined using the following WordprocessingML:

```
<w:sectPr>
  …
  <w:lnNumType w:countBy="1" />
</w:sectPr>
```

This content specifies that line numbers shall be included on each line, restart on each page (the default), be placed automatically based on the text (the default), and shall restart at one (the default). *end example*]

Parent Elements
sectPr (§2.6.17); sectPr (§2.6.18); sectPr (§2.6.19)

Attributes	Description
countBy (Line Number Increments to Display)	Specifies the line number increments to be displayed in the current document. Although each line has an associated line number, only lines which are an even multiple of this value shall be displayed. [*Example*: Consider a document in which only every fifth line shall have a line number. The resulting WordprocessingML for this setting would be: `<w:lnNumType … w:countBy="5"/>` This setting ensures that only lines whose number is a multiple of (e.g. 5, 10, and 15) will have a line number displayed. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
distance (Distance Between Text and Line Numbering)	Specifies the distance between the text margin and the edge of any line numbers appearing in that section. [*Example*: Consider a document in which the line numbering shall appear one-half inch from the text margin. The WordprocessingML for this setting is: `<w:lnNumType … w:distance="720"/>` The distance attribute specifies that there shall be a `720` twip spacing between the text margin and the *line numbering. end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

restart (Line Numbering Restart Setting)	Specifies when the line numbering in this section shall be reset to the line number specified by the start attribute's value.
	
	The line numbering increments for each line (even if it is not displayed) until it reaches the restart point specified by this element.
	
	[*Example*: Consider the line numbering used on each page of this document, which specifies that line numbering shall restart at the top of each new page. This line numbering setting would be defined using the following WordprocessingML:
	
	`<w:sectPr>`
	`…`
	`<w:lnNumType w:restart="newPage" ... />`
	`</w:sectPr>`
	
	The value of newPage specifies that the line numbers shall restart at the top of each page

Attributes	Description
	to the value specified by the start attribute. In this case, newPage is the default, so this value could have been omitted entirely. *end example*] The possible values for this attribute are defined by the ST_LineNumberRestart simple type (§2.18.54).
start (Line Numbering Starting Value)	Specifies the starting value used for the first line whenever the line numbering is restarted by use of the restart attribute. [*Example*: Consider a document in which line numbering shall appear on every fifth line, but the first line shall be treated as line number . This setting would require the following WordprocessingML syntax: `<w:lnNumType w:start="3" w:countBy="5"/>` The start attribute specifies that line numbers shall be counted starting from the number 3. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LineNumber">
    <attribute name="countBy" type="ST_DecimalNumber" use="optional"/>
    <attribute name="start" type="ST_DecimalNumber" use="optional"/>
    <attribute name="distance" type="ST_TwipsMeasure" use="optional"/>
    <attribute name="restart" type="ST_LineNumberRestart" use="optional"/>
</complexType>
```

2.6.9 paperSrc (Paper Source Information)

This element specifies printer-specific settings for the printer tray(s) that shall be used to print different pages in this section in the document.

[Example: Consider a section which shall use the best possible tray when printing all pages in this section. This information is specified using the following WordprocessingML:

```
<w:paperSrc w:first="1" w:other="1" />
```

The attributes on the paperSrc element specify the printer codes for the trays to be used when printing this section. end example]

Parent Elements
sectPr (§2.6.17); sectPr (§2.6.18); sectPr (§2.6.19)

Attributes	Description

Attributes	Description
first (First Page Printer Tray Code)	Specifies a printer-specific code that uniquely identifies a specific printer tray to be used to print the first page of this section in the document. A first value of 1 (the default) is specifically used to indicate that the printer shall automatically select the appropriate printer tray based on the printed page size. [*Example*: Consider a section which shall use the best possible tray when printing the first page in this section. This information is specified using the following WordprocessingML: `<w:paperSrc w:first="1" w:other="1" />` The first attribute on the paperSrc element specifies that the printer shall automatically select the appropriate printer tray based on the printed page size when printing the first page in this section. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
other (Non-First Page Printer Tray Code)	Specifies a printer-specific code that uniquely identifies a specific printer tray to be used to print the each subsequent (non-first) page of this section in the document. An value of 1 (the default) is specifically used to indicate that the printer shall automatically select the appropriate printer tray based on the printed page size. [*Example*: Consider a section which shall use the best possible tray when printing the all pages in this section. This information is specified using the following WordprocessingML: `<w:paperSrc w:first="1" w:other="1" />` The other attribute on the paperSrc element specifies that the printer shall automatically select the appropriate printer tray based on the printed page size when printing all pages after the first in this section. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PaperSource">
   <attribute name="first" type="ST_DecimalNumber"/>
   <attribute name="other" type="ST_DecimalNumber"/>
</complexType>
```

2.6.10 pgBorders (Page Borders)

This element specifies the page borders for each page in this section. Each child element of the pgBorders element specifies a specific of border (left, right, bottom, or top).

[*Example*: Consider a page that specifies a dashed line border around each of the four sides of the page, as follows:

This page border setting would be specified using the following WordprocessingML:

```
<w:pgBorders w:offsetFrom="page">
  <w:top w:val="dashed" w:sz="4" w:space="24" w:color="auto" />
  <w:left w:val="dashed" w:sz="4" w:space="24" w:color="auto" />
  <w:bottom w:val="dashed" w:sz="4" w:space="24" w:color="auto" />
  <w:right w:val="dashed" w:sz="4" w:space="24" w:color="auto" />
</w:pgBorders>
```

The four page borders are each uniquely defined by the top, left, bottom, and right elements, respectively. Global settings that define the placement of all page borders are stored on the pgBorders element directly. *end example*]

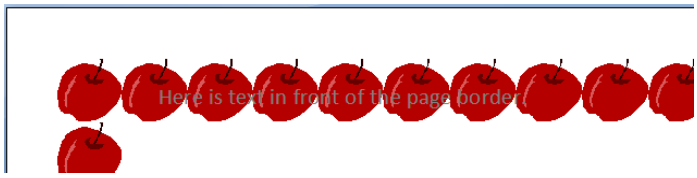
Parent Elements
sectPr (§2.6.17); sectPr (§2.6.18); sectPr (§2.6.19)

Child Elements	Subclause
bottom (Bottom Border)	§2.6.2
left (Left Border)	§2.6.7
right (Right Border)	§2.6.15
top (Top Border)	§2.6.21

Attributes	Description

display (Pages to Display Page Borders)	Specifies the pages in this section on which the page border shall be printed.
	
	If this attribute is omitted, then the page borders shall be displayed on all pages in this section (equivalent to a value of `allPages`).
	
	[*Example*: Consider a section in a document for which the page border shall only be printed on the first page. This setting is specified using the following WordprocessingML:
	
	`<w:pgBorders w:display="firstPage">`
	`  …`
	`</w:pgBorders>`
	
	The display attribute specifies that only the first page shall display the page border defined for this section. *end example*]
	
	The possible values for this attribute are defined by the ST_PageBorderDisplay simple type (§2.18.68).

offsetFrom (Page Border Positioning)	Specifies how the relative positioning of the page borders shall be calculated.
	
	If the value of this attribute is page, then the space attribute on each page border shall be interpreted as the distance from the edge of the page that shall be left before the page border.
	
	If the value of this attribute is `text`, then the space attribute on each page border shall be interpreted as the distance from the text margins that shall be left before the page border.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:top w:val="dashed" w:space="24" />`
	`  <w:left w:val="dashed" w:space="24" />`
	`  <w:bottom w:val="dashed" w:space="24"/>`
	`  <w:right w:val="dashed" w:space="24"/>`
	`</w:pgBorders>`
	
	This fragment specifies that the page borders shall be indented 24 points from the page extents.
	
	This is distinct from the following fragment with identical space attribute values:
	
	`<w:pgBorders w:offsetFrom="text">`
	`  <w:top w:val="dashed" w:space="24" />`
	`  <w:left w:val="dashed" w:space="24" />`
	`  <w:bottom w:val="dashed" w:space="24"/>`
	`  <w:right w:val="dashed" w:space="24"/>`

Attributes	Description
	</w:pgBorders> In this case, the page borders will be offset by 24 points, but in this case, that offset will be calculated relative to the text margins. *end example*] The possible values for this attribute are defined by the ST_PageBorderOffset simple type (§2.18.69).
zOrder (Z-Ordering of Page Border)	Specifies whether the page border is positioned above or below intersecting texts and objects in this document. [*Example*: Consider a document in which the page border shall be displayed below any intersecting text as follows:  This setting is specified by setting the zOrder attribute to back, which specifies that the page border shall be display behind all intersecting text and objects. *end example*] The possible values for this attribute are defined by the ST_PageBorderZOrder simple type (§2.18.70).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PageBorders">
    <sequence>
        <element name="top" type="CT_Border" minOccurs="0"/>
        <element name="left" type="CT_Border" minOccurs="0"/>
        <element name="bottom" type="CT_Border" minOccurs="0"/>
        <element name="right" type="CT_Border" minOccurs="0"/>
    </sequence>
    <attribute name="zOrder" type="ST_PageBorderZOrder" use="optional"/>
    <attribute name="display" type="ST_PageBorderDisplay" use="optional"/>
    <attribute name="offsetFrom" type="ST_PageBorderOffset" use="optional"/>
</complexType>
```

2.6.11 pgMar (Page Margins)

This element specifies the page margins for all pages in this section.

[*Example*: Consider a page with a one-inch margin on all sides. Specifying these margins requires the following WordprocessingML:

```
<w:sectPr>
  <w:pgMar w:bottom="1440" w:top="1440" w:right="1440" w:left="1440"/>
  …
</w:sectPr>
```

This section specifies page margins of 1,440 twentieths of a point (one inch) on all sides. *end example*]

Parent Elements
sectPr (§2.6.17); sectPr (§2.6.18); sectPr (§2.6.19)

Attributes	Description

bottom (Page Bottom Spacing)	Specifies the distance (in twentieths of a point) between the bottom of the text margins for the main document and the bottom of the page for all pages in this section.
	
	If the value of bottom is non-negative, then the text will be placed at the greater of:
	• The value of bottom
	• The extent of the footer text
	
	[*Example*: Consider a document where the footer shall start one inch of the bottom of the page extent, but the contents of the main document story specify that they shall start one-half of an inchh from the page extents. To specify these boundaries, the following page margins may specified in the WordprocessingML:
	
	`<w:pgMar ... w:bottom="720" w:footer="1440"/>`
	
	This fragment specifies that the footer shall start 1440 twentieths of a point from the bottom of the page, but the main document story shall start 720 twentieths of a point from the bottom of the page. Since the footer extent is guaranteed to be greater in this case, the bottom text extent ends at the top of the footer region. *end example*]
	
	A negative value indicates that the contents of the main document shall be measured from the bottom of the page extent regardless of the footer for that document, and therefore shall overlap the footer text.
	
	[*Example*: Consider a document where the footer shall start one inch of the bottom of the page extent, but the contents of the main document story shall start one-half of an inch from the page extents. To specify these boundaries, the following page margins may specified in the WordprocessingML:
	
	`<w:pgMar ... w:bottom="-720" w:footer="1440"/>`
	
	This fragment specifies that the footer shall start 1440 twentieths of a point from the bottom of the page, and the main document story shall start 720 twentieths of a point

Attributes	Description
	from the bottom of the page. Since the value of bottom is negative in this case, the bottom text extent starts one-half of an inch from the bottom of the page and overlaps any footer text. *end example*] The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§2.18.88).
footer (Spacing to Bottom of Footer)	Specifies the distance (in twentieths of a point) from the bottom edge of the page to the bottom edge of the footer. [*Example*: Consider a document where the footer shall start one inch above the bottom of the page extent. To specify this boundary, the following page margins must specified in the WordprocessingML: `<w:pgMar … w:footer="1440"/>` This fragment specifies that the footer shall start 1440 twentieths of a point from the bottom of the page. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).
gutter (Page Gutter Spacing)	Specifies the page gutter for each page in the current section. The *page gutter* defines the amount of extra space added to the specified margin, above any existing margin values. [*Note*: This setting is typically used when a document is being created for binding, in order to ensure that the resulting margins are present after the binding gutter is consumed by the printed matter binding. *end note*] [*Example*: Consider a document where the margin shall start one inch of the left edge of the page extent after one-half of an inch is hidden by the page binding. To specify this condition, a user could simply use a left margin of 1.5 inches, which would be lost if the margins are later changed, or could specify a one-half inch gutters follows in the WordprocessingML: `<w:pgMar … w:gutter="720"/>` This fragment specifies that the gutter shall span 720 twentieths of a point, after which any margin value shall be added. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).
header (Spacing to Top of Header)	Specifies the distance (in twentieths of a point) from the top edge of the page to the top edge of the header.

Attributes	Description
	[*Example*: Consider a document where the header shall start two inches below the top of the page extent. To specify this boundary, the following page margins must specified in the WordprocessingML: `<w:pgMar … w:header="2880"/>` This fragment specifies that the header shall start 2880 twentieths of a point from the top of the page. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).
left (Left Margin Spacing)	Specifies the distance (in twentieths of a point) between the left edge of the page and the left edge of the text extents for this document. [*Example*: Consider a document where the left text extent shall start two inches inside the page extent. To specify this boundary, the following page margins must specified in the WordprocessingML: `<w:pgMar … w:left="2880"/>` This fragment specifies that the left margin shall span 2880 twentieths of a point from the left edge of the page. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).
right (Right Margin Spacing)	Specifies the distance (in twentieths of a point) between the right edge of the page and the right edge of the text extents for this document. [*Example*: Consider a document where the right text extent shall start one inch inside the page. To specify this boundary, the following page margins must specified in the WordprocessingML: `<w:pgMar … w:right="1440"/>` This fragment specifies that the right margin shall span 1440 twentieths of a point from the right edge of the page. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).
top (Top Margin Spacing)	Specifies the distance (in twentieths of a point) between the top of the text margins for the main document and the top of the page for all pages in this section.

Attributes	Description

	If the value of top is non-negative, then the text will be placed at the greater of:
	• The value of top
	• The extent of the header text
	
	[*Example*: Consider a document where the header shall start one inch from the top of the page extent, but the contents of the main document story specify that they shall start one-half of an inch from the page extents. To specify these boundaries, the following page margins may specified in the WordprocessingML:
	
	`<w:pgMar … w:top="720" w:header="1440"/>`
	
	This fragment specifies that the header shall start 1440 twentieths of a point from the top of the page, but the main document story shall start 720 twentieths of a point from the top of the page. Since the header extent is guaranteed to be greater in this case, the main text extent ends at the bottom of the header region. *end example*]
	
	A negative value indicates that the contents of the main document shall be measured from the top of the page extent regardless of the header for that document, and therefore shall overlap the header text.
	
	[*Example*: Consider a document where the header shall start one inch from the top of the page extent, but the contents of the main document story shall start one-half of an inch from the page extents. To specify these boundaries, the following page margins may specified in the WordprocessingML:
	
	`<w:pgMar ... w:top="-720"`
	`w:header="1440"/>`
	
	This fragment specifies that the header shall start 1440 twentieths of a point from the top of the page, and the main document story shall start 720 twentieths of a point from the top of the page. Since the value of top is negative in this case, the top text extent starts one-half of an inch from the top of the page and overlaps any header text. *end example*]
	
	The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§2.18.88).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PageMar">
   <attribute name="top" type="ST_SignedTwipsMeasure" use="required"/>
   <attribute name="right" type="ST_TwipsMeasure" use="required"/>
   <attribute name="bottom" type="ST_SignedTwipsMeasure" use="required"/>
   <attribute name="left" type="ST_TwipsMeasure" use="required"/>
   <attribute name="header" type="ST_TwipsMeasure" use="required"/>
   <attribute name="footer" type="ST_TwipsMeasure" use="required"/>
   <attribute name="gutter" type="ST_TwipsMeasure" use="required"/>
</complexType>
```

2.6.12 pgNumType (Page Numbering Settings)

This element specifies the page numbering settings for all page numbers that appear in the contents of the current section.

[*Example*: Consider a section in which the page numbers shall start at page 25. The following WordprocessingML syntax specifies that requirement:

```
<w:sectPr>
  …
  <w:pgNumType w:start="25"/>
</w:sectPr>
```

The pgNumType element specifies that numbering on this section shall start from page number 25. *end example*]

Parent Elements
sectPr (§2.6.17); sectPr (§2.6.18); sectPr (§2.6.19)

Attributes	Description
chapSep (Chapter Separator Character)	Specifies the separator character that shall appear between the chapter and page number, if a chapter style has been set for page numbers in this section. If the chapStyle attribute is not present, or its specified heading level does not have an associated numbering format, then this value is ignored, since no chapter number is output by the field. [Example: Consider a section in a document in which the chapter shall be separated from the page number using a colon character. This constraint would be specified using the following WordprocessingML: `<w:pgNumType w:chapSep="colon" chapStyle="1" />` The chapSep attribute declares that the chapter and page number shall be separated by a colon (e.g. 1:1 for chapter one, page one). end example] The possible values for this attribute are defined by the ST_ChapterSep simple type (§2.18.9).
chapStyle (Chapter Heading Style)	Specifies the one-based index of the heading style applied to chapter titles in the document which shall be used as chapter headings in all page numbers for this section, by locating the nearest heading of that style and extracting the numbering information. If the specified heading style does not exist in the current section, or does not have a numbering format, then any previous level heading format shall be used as needed as the specified chapter number. If no heading has numbering information and/or is used in the

Attributes	Description
	section, then the chapter and chapter separator shall be omitted from the page numbering data. [*Example*: Consider a page number in a section with page numbering properties that specify a chapStyle of 1 (Heading 1 style) and a chapSep of dash. This means that for each page number in this section, the numbering value of the nearest Heading 1 style is used for the chapter value, and is followed by a dash, then the page n umber in that section. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
fmt (Page Number Format)	Specifies the number format that shall be used for all page numbering in this section. [*Example*: A fmt value of lowerLetter indicates that a consumer shall use lowercase letters for each page in this section: a,b,c… *end example*] The possible values for this attribute are defined by the ST_NumberFormat simple type (§2.18.66).
start (Starting Page Number)	Specifies the page number that appears on the first page of the section. If this value is omitted, numbering will continue from the highest page number in the previous section. [*Example*: Consider the following WordprocessingML: `<w:pgNumType w:fmt="2"/>` Because the start value is omitted, the page numbers in this section begin at the value of the highest page in the previous section. This means that if the previous section ended in page 7, this section would start with page 8. *end example]* The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PageNumber">
   <attribute name="fmt" type="ST_NumberFormat" use="optional"/>
   <attribute name="start" type="ST_DecimalNumber" use="optional"/>
   <attribute name="chapStyle" type="ST_DecimalNumber" use="optional"/>
   <attribute name="chapSep" type="ST_ChapterSep" use="optional"/>
</complexType>
```

2.6.13 pgSz (Page Size)

This element specifies the properties (size and orientation) for all pages in the current section.\

[*Example*: Consider a section that shall be printed on A4 paper. The WordprocessingML for this paper size is as follows:

```
<w:pgSz w:w="11907" w:h="16839" />
```

This output states that all pages in this section shall be 11907 twentieths of a point wide (11907 twentieths of a point = 8.269") and 16839 twentieths of a point high (16939 twentieths of a point = 11.694"). *end example*]

Parent Elements
sectPr (§2.6.17); sectPr (§2.6.18); sectPr (§2.6.19)

Attributes	Description
code (Printer Paper Code)	Specifies a printer-specific paper code for the paper type, which shall be used by the printer for pages in this section. This code is stored to ensure the proper paper type is chosen if the specified paper size matches the sizes of multiple paper types supported by the current printer. [Example: Consider the following WordprocessingML fragment: `<w:pgSz … w:code="240" />` The code attribute specifies a value of 240, which will be sent to the printer and used by the printer to determine the appropriate paper type to use when printing. This value is not interpreted or modified other than storing it as specified by the printer. end example] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
h (Page Height)	Specifies the height (in twentieths of a point) for all pages in the current section. [*Example*: Consider the following WordprocessingML: `<w:pgSz w:w="15840" w:h="12240" />` All pages in this section are displayed on a page that is 12240 twentieths of a point (8.5") tall. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

Attributes	Description
orient (Page Orientation)	Specifies the orientation of all pages in this section. This information is used to determine the actual paper size to use on the printer. [*Example*: Pages 11" wide by 8.5" long in landscape mode use 8.5"x11" paper, because the width and height are reversed for pages in this landscape section with respect to the printed page. *end example*] This implies that the actual paper size width and height are reversed for pages in this section. If this attribute is omitted, then `portrait` shall be implied. [*Example*: Consider the following WordprocessingML: `<w:pgSz w:w="15840" w:h="12240" w:orient="landscape" />` Although the page width is 11", and page height is 8.5", according to the w and h attributes, because the orient attribute is set to `landscape`, pages in this section are printed on 8.5x11" paper in landscape mode. *end example*] The possible values for this attribute are defined by the ST_PageOrientation simple type (§2.18.71).
w (Page Width)	This attribute indicates the width (in twentieths of a point) for all pages in the current section. [*Example*: Consider the following WordprocessingML: `<w:pgSz w:w="15840" w:h="12240" />` All pages in this section are displayed on a page that is `15840` twentieths of a point (11") wide. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PageSz">
   <attribute name="w" type="ST_TwipsMeasure" use="optional"/>
   <attribute name="h" type="ST_TwipsMeasure" use="optional"/>
   <attribute name="orient" type="ST_PageOrientation" use="optional"/>
   <attribute name="code" type="ST_DecimalNumber"/>
</complexType>
```

2.6.14 printerSettings (Reference to Printer Settings Data)

This element specifies an explicit relationship to a Printer Settings part containing information about the printer settings used for this section.

If this element is omitted, than no additional settings are associated with this section.

[*Example*: Consider a producer which needed to store additional printer settings for each section. A document from such a producer would have the following section properties:

```
<w:sectPr>
  ...
  <w:printerSettings r:id="rId10" />
</w:sectPr>
```

The resulting Main Document part would a relationship to the appropriate Printer Settings part with a relationship ID of `rId10`. *end example*]

Parent Elements
sectPr (§2.6.17); sectPr (§2.6.18); sectPr (§2.6.19)

Attributes	Description
id (Relationship to Part) Namespace: .../officeDocument/2006/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the type required by the parent element: • http://schemas.openxmlformats.org/officeDocument/2006/relationships/footer for the footerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/header for the headerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://schemas.openxmlformats.org/officeDocument/2006/relationships/printerSettings for the printerSettings element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

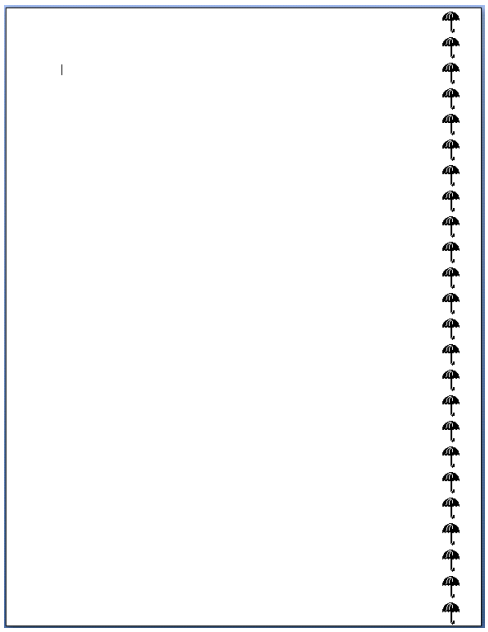
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Rel">
   <attribute ref="r:id" use="required"/>
</complexType>
```

2.6.15 right (Right Border)

This element specifies the presentation and display of the page border displayed at the right of each page in this section.

[*Example*: Consider a section in which all pages shall have a right border consisting of a repeated image of an umbrella, like this:



This border would result in the following WordprocessingML:

```
<w:sectPr>
  …
  <w:pgBorders>
    <w:right w:val="seattle" …/>
  </w:pgBorders>
  …
</w:sectPr>
```

Because the page has a border at the right only, only the right element is specified within the set of page borders. *end example*]

When a document has a right border that is relative to the page edges (using the offsetFrom attribute value of page on pgBorders), it shall span the right edge of the page at the location defined by its properties, stopping when:

- It intersects with the corresponding top or bottom page border (if one is specified)
- It reaches the edge of the page.

[*Example*: In the example above, no top or bottom border was specified in the WordprocessingML, so a consumer shall draw the border from one edge of the page to the other. *end example*]

When a document has a right border that is relative to the text (using the offsetFrom attribute value of `text` on pgBorders), it shall only span the necessary width to satisfy the requirement of spanning the width of the text.

Parent Elements
pgBorders (§2.6.10)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or `auto` to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value `auto`, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is `true`, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border	Specifies whether this border should be modified to create the appearance of a shadow.

Attributes	Description
Shadow)	For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object.
	
	When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points.
	
	When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.
	
	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24/>`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*]
	
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).

sz (Border Width)	Specifies the width of the current border. If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this

Attributes	Description

	range may be reassigned to a more appropriate value.
	
	If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value.
	
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`
	
	The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	
	The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).

themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border.
	
	The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document.
	
	[*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup:
	
	`<w:top … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:bottom … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:left … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:right … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	
	The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*]
	
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).

themeShade (Border Theme	Specifies the shade value applied to the supplied theme color (if any) for this border instance.

Attributes	Description
Color Shade)	If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255$$ $$= 102$$ $$= 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L' = 0.53 * 0.75$$ $$= 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255$$ $$= 153$$ $$= 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6)$$ $$= 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2"`

Attributes	Description
	w:themeTint="99"/> *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: <w:left w:val="single" …/> This border's val is single, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <attribute name="val" type="ST_Border" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
   <attribute name="space" type="ST_PointMeasure" use="optional"/>
   <attribute name="shadow" type="ST_OnOff" use="optional"/>
   <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.6.16 rtlGutter (Gutter on Right Side of Page)

This element specifies that the page gutter shall be placed on the right side of the page for this section only. The *page gutter* defines the amount of extra space added to the specified margin, above any existing margin values. [*Note*: This setting is typically used when a document is being created for binding, in order to ensure that the resulting margins are present after the binding gutter is consumed by the printed matter binding. *end note*]

If the gutter is set to the side of the page by the omission of the gutterAtTop element (§2.15.1.49), then each section's gutter is placed at the left by default, unless that default is overridden by the rtlGutter element.

[*Example*: Consider a document with three sections, with gutter properties defined as follows:

```
<w:p>
  <w:pPr>
    <w:sectPr>
      <w:pgMar w:gutter="1440" …/>
      …
    </w:sectPr>
  </w:pPr>
</w:p>
...
<w:p>
  <w:pPr>
    <w:sectPr>
      <w:pgMar w:gutter="1440" …/>
      <w:rtlGutter w:val="0" />
      …
    </w:sectPr>
  </w:pPr>
</w:p>
...
<w:p>
  <w:pPr>
    <w:sectPr>
      <w:pgMar w:gutter="1440" …/>
      <w:rtlGutter />
      …
    </w:sectPr>
  </w:pPr>
</w:p>
```

The first and second sections both place the gutter on the left side, the first by omission of the rtlGutter attribute, and the second by explicitly turning it off. The third section, however, moves the gutter to the right side via the use of the rtlGutter attribute. *end example*]

If the gutterAtTop element (§2.15.1.49) is specified and true, then each section's gutter is at the top and this setting is ignored.

Parent Elements
sectPr (§2.6.17); sectPr (§2.6.18); sectPr (§2.6.19)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element.

Attributes	Description
	A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.6.17 sectPr (Previous Section Properties)

When specified as a child element of sectPrChange, the sectPr element specifies a set of section properties that were modified when the document was set to track all revisions.

[*Example*: If the page orientation was changed with revision tracking enabled, the following WordprocessingML defines the contents of that change:

```
<w:sectPr>
  …
  <w:sectPrChange …>
    <w:sectPr>
      <w:pgSz w:w="15840" w:h="12240"/>
    </w:sectPr>
  </w:sectPrChange>
</w:sectPr>
```

The properties that were changed as part of this revision are stored in this sectPr element. *end example*]

Parent Elements
sectPrChange (§2.13.5.34)

Child Elements	Subclause
bidi (Right to Left Section Layout)	§2.6.1

Child Elements	Subclause
cols (Column Definitions)	§2.6.4
docGrid (Document Grid)	§2.6.5
endnotePr (Section-Wide Endnote Properties)	§2.11.5
footnotePr (Section-Wide Footnote Properties)	§2.11.12
formProt (Only Allow Editing of Form Fields)	§2.6.6
lnNumType (Line Numbering Settings)	§2.6.8
noEndnote (Suppress Endnotes In Document)	§2.11.16
paperSrc (Paper Source Information)	§2.6.9
pgBorders (Page Borders)	§2.6.10
pgMar (Page Margins)	§2.6.11
pgNumType (Page Numbering Settings)	§2.6.12
pgSz (Page Size)	§2.6.13
printerSettings (Reference to Printer Settings Data)	§2.6.14
rtlGutter (Gutter on Right Side of Page)	§2.6.16
textDirection (Text Flow Direction)	§2.6.20
titlePg (Different First Page Headers and Footers)	§2.10.6
type (Section Type)	§2.6.22
vAlign (Vertical Text Alignment on Page)	§2.6.23

Attributes	Description
rsidDel (Section Deletion Revision ID)	Specifies a unique identifier used to track the *editing session* when the section mark for this section was deleted from the document. All rsid* attributes throughout this document of an equal value, if present, shall indicate that those regions were modified during the same editing session. A producer may choose to increment the revision save ID value to indicate subsequent editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).
rsidR (Section Addition Revision ID)	Specifies a unique identifier used to track the *editing session* when the section mark for this section was added to the document. All rsid* attributes throughout this document of an equal value, if present, shall indicate that those regions were modified during the same editing session.

Attributes	Description
	A producer may choose to increment the revision save ID value to indicate subsequent editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).
rsidRPr (Physical Section Mark Character Revision ID)	Specifies a unique identifier used to track the editing session when the physical character representing this section mark was last formatted. All rsid* attributes throughout this document of an equal value, if present, shall indicate that those regions were modified during the same editing session. A producer may choose to increment the revision save ID value to indicate subsequent editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).
rsidSect (Section Properties Revision ID)	Specifies a unique identifier used to track the editing session when the physical character representing this section mark was last formatted. All rsid* attributes throughout this document of an equal value, if present, shall indicate that those regions were modified during the same editing session. A producer may choose to increment the revision save ID value to indicate subsequent editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SectPrBase">
   <sequence>
      <group ref="EG_SectPrContents" minOccurs="0"/>
   </sequence>
   <attributeGroup ref="AG_SectPrAttributes"/>
</complexType>
```

2.6.18 sectPr (Document Final Section Properties)

This element defines the section properties for the final section of the document. [*Note*: For any other section the properties are stored as a child element of the paragraph element corresponding to the last paragraph in the given section. *end note*]

[*Example*: Consider a document with multiple sections. For all sections except the final section, the sectPr element is stored as a child element of the last paragraph in the section. For the final section, this information is stored as the last child element of the body element, as follows:

```
<w:body>
  <w:p>
  …
  </w:p>
  …
  <w:sectPr>
    (final section's properties)
  </w:sectPr>
</w:body>
```

end example]

Parent Elements
body (§2.2.2); docPartBody (§2.12.6)

Child Elements	Subclause
bidi (Right to Left Section Layout)	§2.6.1
cols (Column Definitions)	§2.6.4
docGrid (Document Grid)	§2.6.5
endnotePr (Section-Wide Endnote Properties)	§2.11.5
footerReference (Footer Reference)	§2.10.2
footnotePr (Section-Wide Footnote Properties)	§2.11.12
formProt (Only Allow Editing of Form Fields)	§2.6.6
headerReference (Header Reference)	§2.10.5
lnNumType (Line Numbering Settings)	§2.6.8
noEndnote (Suppress Endnotes In Document)	§2.11.16
paperSrc (Paper Source Information)	§2.6.9
pgBorders (Page Borders)	§2.6.10
pgMar (Page Margins)	§2.6.11
pgNumType (Page Numbering Settings)	§2.6.12
pgSz (Page Size)	§2.6.13
printerSettings (Reference to Printer Settings Data)	§2.6.14
rtlGutter (Gutter on Right Side of Page)	§2.6.16
sectPrChange (Revision Information for Section Properties)	§2.13.5.34

Child Elements	Subclause
textDirection (Text Flow Direction)	§2.6.20
titlePg (Different First Page Headers and Footers)	§2.10.6
type (Section Type)	§2.6.22
vAlign (Vertical Text Alignment on Page)	§2.6.23

Attributes	Description
rsidDel (Section Deletion Revision ID)	Specifies a unique identifier used to track the *editing session* when the section mark for this section was deleted from the document. All rsid* attributes throughout this document of an equal value, if present, shall indicate that those regions were modified during the same editing session. A producer may choose to increment the revision save ID value to indicate subsequent editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).
rsidR (Section Addition Revision ID)	Specifies a unique identifier used to track the *editing session* when the section mark for this section was added to the document. All rsid* attributes throughout this document of an equal value, if present, shall indicate that those regions were modified during the same editing session. A producer may choose to increment the revision save ID value to indicate subsequent editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).
rsidRPr (Physical Section Mark Character Revision ID)	Specifies a unique identifier used to track the editing session when the physical character representing this section mark was last formatted. All rsid* attributes throughout this document of an equal value, if present, shall indicate that those regions were modified during the same editing session. A producer may choose to increment the revision save ID value to indicate subsequent editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).

Attributes	Description
rsidSect (Section Properties Revision ID)	Specifies a unique identifier used to track the editing session when the physical character representing this section mark was last formatted. All rsid* attributes throughout this document of an equal value, if present, shall indicate that those regions were modified during the same editing session. A producer may choose to increment the revision save ID value to indicate subsequent editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SectPr">
   <sequence>
      <group ref="EG_HdrFtrReferences" minOccurs="0" maxOccurs="6"/>
      <group ref="EG_SectPrContents" minOccurs="0"/>
      <element name="sectPrChange" type="CT_SectPrChange" minOccurs="0"/>
   </sequence>
   <attributeGroup ref="AG_SectPrAttributes"/>
</complexType>
```

2.6.19 sectPr (Section Properties)

This element defines the section properties for the a section of the document. [*Note*: For the last section in the document, the section properties are stored as a child element of the body element. *end note*]

[*Example*: Consider a document with multiple sections. For all sections except the final section, the sectPr element is stored as a child element of the last paragraph in the section, as follows:

```
<w:body>
  <w:p>
    <w:pPr>
      <w:sectPr>
        (final section's properties)
      </w:sectPr>
    </w:pPr>
  …
  </w:p>
  …
  <w:sectPr>
    (final section's properties)
  </w:sectPr>
</w:body>
```

end example]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Child Elements	Subclause
bidi (Right to Left Section Layout)	§2.6.1
cols (Column Definitions)	§2.6.4
docGrid (Document Grid)	§2.6.5
endnotePr (Section-Wide Endnote Properties)	§2.11.5
footerReference (Footer Reference)	§2.10.2
footnotePr (Section-Wide Footnote Properties)	§2.11.12
formProt (Only Allow Editing of Form Fields)	§2.6.6
headerReference (Header Reference)	§2.10.5
lnNumType (Line Numbering Settings)	§2.6.8
noEndnote (Suppress Endnotes In Document)	§2.11.16
paperSrc (Paper Source Information)	§2.6.9
pgBorders (Page Borders)	§2.6.10
pgMar (Page Margins)	§2.6.11
pgNumType (Page Numbering Settings)	§2.6.12
pgSz (Page Size)	§2.6.13
printerSettings (Reference to Printer Settings Data)	§2.6.14
rtlGutter (Gutter on Right Side of Page)	§2.6.16
sectPrChange (Revision Information for Section Properties)	§2.13.5.34
textDirection (Text Flow Direction)	§2.6.20
titlePg (Different First Page Headers and Footers)	§2.10.6
type (Section Type)	§2.6.22
vAlign (Vertical Text Alignment on Page)	§2.6.23

Attributes	Description
rsidDel (Section Deletion Revision ID)	Specifies a unique identifier used to track the *editing session* when the section mark for this section was deleted from the document. All rsid* attributes throughout this document of an equal value, if present, shall indicate that those regions were modified during the same editing session.

Attributes	Description
	A producer may choose to increment the revision save ID value to indicate subsequent editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).
rsidR (Section Addition Revision ID)	Specifies a unique identifier used to track the *editing session* when the section mark for this section was added to the document. All rsid* attributes throughout this document of an equal value, if present, shall indicate that those regions were modified during the same editing session. A producer may choose to increment the revision save ID value to indicate subsequent editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).
rsidRPr (Physical Section Mark Character Revision ID)	Specifies a unique identifier used to track the editing session when the physical character representing this section mark was last formatted. All rsid* attributes throughout this document of an equal value, if present, shall indicate that those regions were modified during the same editing session. A producer may choose to increment the revision save ID value to indicate subsequent editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).
rsidSect (Section Properties Revision ID)	Specifies a unique identifier used to track the editing session when the physical character representing this section mark was last formatted. All rsid* attributes throughout this document of an equal value, if present, shall indicate that those regions were modified during the same editing session. A producer may choose to increment the revision save ID value to indicate subsequent editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SectPr">
   <sequence>
      <group ref="EG_HdrFtrReferences" minOccurs="0" maxOccurs="6"/>
      <group ref="EG_SectPrContents" minOccurs="0"/>
      <element name="sectPrChange" type="CT_SectPrChange" minOccurs="0"/>
   </sequence>
   <attributeGroup ref="AG_SectPrAttributes"/>
</complexType>
```

2.6.20 textDirection (Text Flow Direction)

This element specifies the direction of the text flow for this section.

[*Example*: Consider a document with a section in which text shall flow bottom to top vertically, and left to right horizontally. This setting requires the following WordprocessingML:

```
<w:sectPr>
  …
  <w:textDirection w:val="btLr" />
</w:sectPr>
```

The textDirection element specifies via the `btLr` value in the val attribute that the text flow shall go bottom to top, and left to right. *end example*]

Parent Elements
sectPr (§2.6.17); sectPr (§2.6.18); sectPr (§2.6.19)

Attributes	Description

val (Direction of Text Flow)	Specifies the direction of the text flow for this object.
	
	[*Example*: Consider a document with a section in which text shall flow bottom to top vertically, and left to right horizontally. This setting requires the following WordprocessingML:
	
	`<w:sectPr>`
	`  …`
	`  <w:textDirection w:val="btLr" />`
	`</w:sectPr>`
	
	The textDirection element specifies via the `btLr` value in the val attribute that the text flow shall go bottom to top, and left to right. *end example*]
	
	The possible values for this attribute are defined by the ST_TextDirection simple type (§2.18.100).

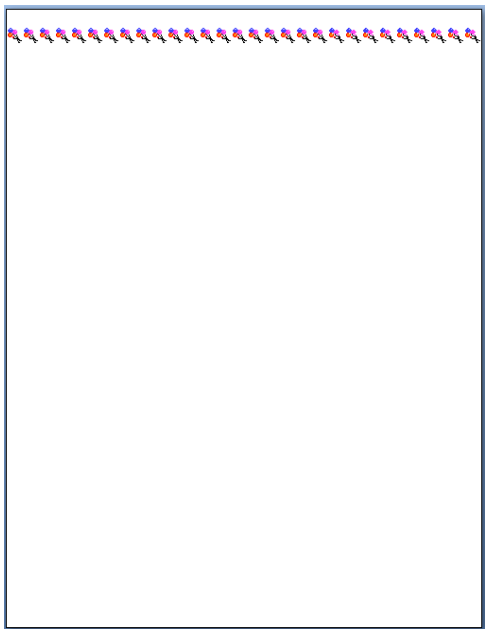
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextDirection">
   <attribute name="val" type="ST_TextDirection" use="required"/>
</complexType>
```

2.6.21 top (Top Border)

This element specifies the presentation and display of the page border displayed at the top of each page in this section.

[*Example*: Consider a section in which all pages shall have a top border consisting of a repeated image of balloons, like this:



This border would result in the following WordprocessingML:

```
<w:sectPr>
  …
  <w:pgBorders>
    <w:top w:val="balloons" .../>
  </w:pgBorders>
  …
</w:sectPr>
```

Because the page only has a border at the top, only the top element is specified within the set of page borders. *end example*]

When a document has a top border that is relative to the page edges (using an offsetFrom attribute value of page on pgBorders), it shall span the top edge of the page at the location defined by its properties, stopping when:

- It intersects with the corresponding left or right page border (if one is specified)
- It reaches the edge of the page.

[*Example*: In the example above, no left or right border was specified in the WordprocessingML, so a consumer shall draw the border from one edge of the page to the other. *end example*]

When a document has a top border that is relative to the text (using the offsetFrom attribute value of `text` on pgBorders), it shall only span the necessary width to satisfy the requirement of spanning the width of the text.

Parent Elements
pgBorders (§2.6.10)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or `auto` to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value `auto`, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is `true`, indicating that the border frame effect shall be applied. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object.
	
	When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points.
	
	When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.
	
	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24/>`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*]
	
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).

sz (Border Width)	Specifies the width of the current border.

Attributes	Description

	If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value.
	
	If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value.
	
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`
	
	The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	
	The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).

themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border.
	
	The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document.
	
	[*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup:
	
	`<w:top … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:bottom … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:left … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:right … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	
	The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*]
	
	The possible values for this attribute are defined by the ST_ThemeColor simple type

Attributes	Description
	(§2.18.104).
themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255 = 102 = 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L' = 0.53 * 0.75 = 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>`

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255 = 153 = 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$L' = L * Tint_{pct} + (1 - Tint_{pct})$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6) = 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$ and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting background's color attribute:

Attributes	Description
	<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/> *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: <w:left w:val="single" …/> This border's val is single, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <attribute name="val" type="ST_Border" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
   <attribute name="space" type="ST_PointMeasure" use="optional"/>
   <attribute name="shadow" type="ST_OnOff" use="optional"/>
   <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.6.22 type (Section Type)

This element specifies the type of the current section. The section type specifies how the contents of the current section shall be placed relative to the previous section.

WordprocessingML supports five distinct types of section breaks:

- *Next page section breaks* (the default if type is not specified), which begin the new section on the following page.
- *Odd page section breaks*, which begin the new section on the next odd-numbered page.
- *Even page section breaks*, which begin the new section on the next even-numbered page.

- *Continuous section breaks*, which begin the new section on the following paragraph. This means that continuous section breaks might not specify certain page-level section properties, since they must be inherited from the following section. These breaks, however, can specify other section properties, such as line numbering and footnote/endnote settings.
- *Column section breaks*, which begin the new section on the next column on the page.

Parent Elements
sectPr (§2.6.17); sectPr (§2.6.18); sectPr (§2.6.19)

Attributes	Description

val (Section Type Setting)	Specifies the type of the current section.
	
	[*Example*: Consider a section that shall start on the next page in the document. The WordprocessingML specifying this would look like:
	
	`<w:sectPr>`
	`  …`
	`  <w:type w:val="nextPage"/>`
	`</w:sectPr>`
	
	The nextPage value specifies that this section starts on the next page. *end example*]
	
	The possible values for this attribute are defined by the ST_SectionMark simple type (§2.18.84).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SectType">
   <attribute name="val" type="ST_SectionMark"/>
</complexType>
```

2.6.23 vAlign (Vertical Text Alignment on Page)

This element specifies the vertical alignment for text on pages in the current section, relative to the top and bottom margins in the main document story on each page.

[*Example*: Consider a section used as a title page on which text shall be vertically centered. In order to center the text vertically on the page, the following WordprocessingML is used:

```
<w:sectPr>
  …
  <w:vAlign w:val="center" />
</w:sectPr>
```

The vAlign value of center specifies that text shall be laid out in the center of the top and bottom text margins for all pages in this section. *end example*]

Parent Elements
sectPr (§2.6.17); sectPr (§2.6.18); sectPr (§2.6.19)

Attributes	Description
val (Vertical Alignment Setting)	Specifies the vertical alignment for text between the top and bottom margins of the parent container (page or table cell). [*Example*: Consider a region where the text shall be vertically centered in the parent element. This would require a val value of `center`, in order to specify that all justification vertically shall be centered relative to the parent. For a section, this setting would be specified as follows: `<w:vAlign w:val="center" />` The val attribute of `center` specifies that the content is centered relative to its container (in this case, the page). *end example*] The possible values for this attribute are defined by the ST_VerticalJc simple type (§2.18.111).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_VerticalJc">
   <attribute name="val" type="ST_VerticalJc" use="required"/>
</complexType>
```

2.7 Styles

Within a WordprocessingML file, *styles* are predefined sets of table, numbering, paragraph, and/or character properties which can be applied to text within the document. This allows the formatting properties to be stored and managed independently from the content, allowing the look of document content to be changed in a single location (e.g. the look of all first-level headings is changed by changing the style with styleId `Heading1` rather than looking for and changing each paragraph in the document).

[*Example*: The Normal paragraph style in a word processing document can have any number of formatting properties, e.g. font face = Times New Roman; font size = 12pt; paragraph justification = left). All paragraphs which reference this paragraph style would automatically inherit these properties. *end example*]

Each style defined within a WordprocessingML document requires a *style definition*. The style definition contains all of the information needed by a consumer to store and display that style within a WordprocessingML document, and is defined using the style element. The style definition for any style in WordprocessingML can be divided into three segments The complete definition of style properties can be found on the reference for the style element (§2.7.3.17):

- General style properties
- Style types

- Type specific formatting properties

Each of these three segments are discussed in the following subclauses.

2.7.1 Style Inheritance

In order to compile the complete set of paragraph and character properties specified by any given style (as appropriate), a consumer must follow the rule of style inheritance to determine each property in that set.

Style inheritance states that styles of any given type may inherit from other styles of that type, and therefore a consumer must 'build up' the style information by following the inheritance tree. This inheritance is defined via the basedOn element, which specifies the styleId of the parent style.

[*Example*: The "Tristan Test" paragraph style can inherit properties from the "Heading 1" paragraph style, which itself can inherit properties from the "Normal" paragraph style. *end example*]

To build up the resulting style, a consumer must trace the hierarchy (following each basedOn value) back to a style which has no basedOn element (is not based on another style). The resulting style is then constructed by following each level in the tree, applying the specified paragraph and/or character properties as appropriate. When properties conflict, they are overridden by each subsequent level (this includes turning OFF a property set at an earlier level). Properties which are not specified simply do not change those specified at earlier levels.

[*Example*: Consider a character style Green which specifies only that the text color is green, but inherits from another character style Base which defines a font face of Arial, as well as bold:

```
<w:style w:type="character" w:styleId="Green">
  <w:name w:val="Green" />
  <w:basedOn w:val="Base" />
  <w:rPr>
    <w:color w:val="22B14C" />
  </w:rPr>
</w:style>
...
../Local Settings/Temp/styles.xml<w:style w:type="character" w:styleId="Base">
  <w:name w:val="Base" />
  <w:rPr>
    <w:rFonts w:ascii="Arial" w:hAnsi="Arial" />
    <w:b />
  </w:rPr>
</w:style>
```

The definition of the Green character style has a basedOn element which specifies the Base style. This means that any use of the Green style is defined as bold, green, Arial text. *end example*]

Conversely, a producer should not output any property on a style which has already been set by a previous level of the style hierarchy, as well as those which match the document defaults. This means that if the document

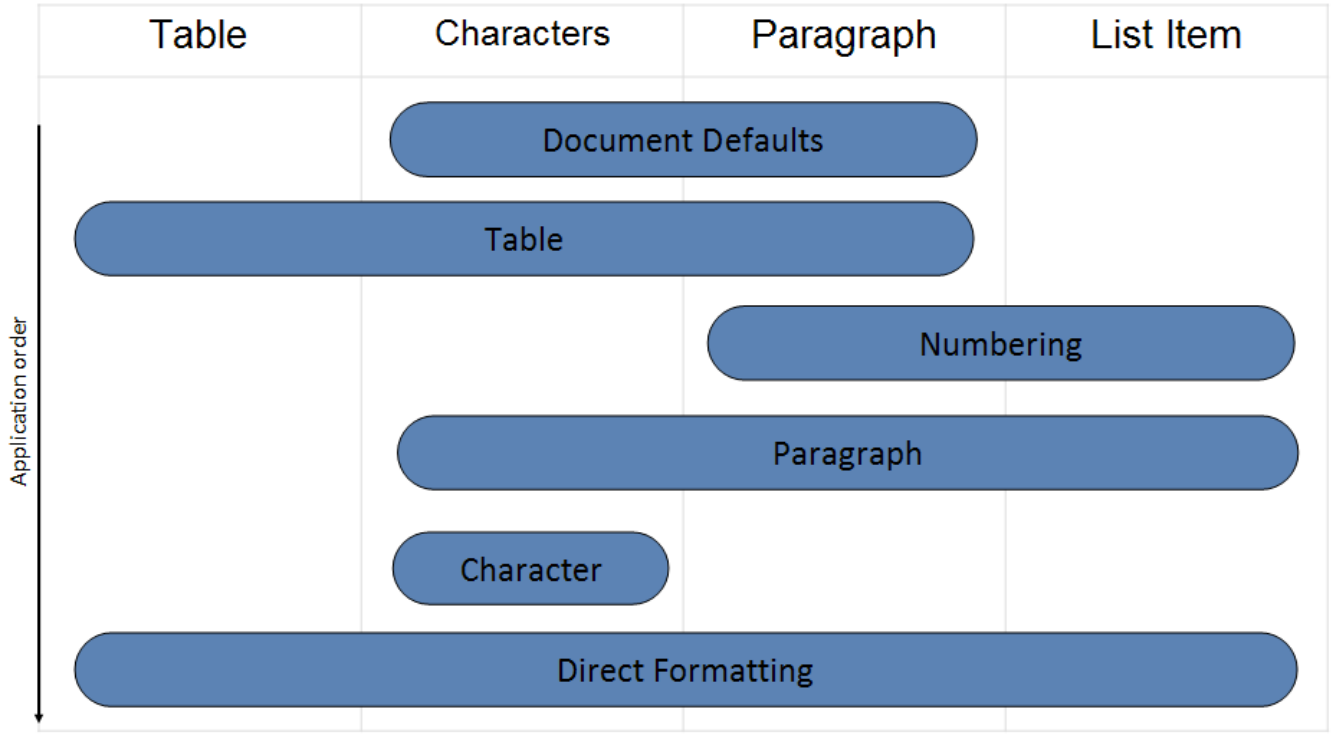
defaults or any previous level in a style's hierarchy specify a property which is unchanged at this level, that property should not be part of the style definition in the resulting WordprocessingML.

[*Example*: If the document default font is `Bauhaus 93` and the `Heading 1` style also specifies the Bauhaus 93 font, then a producer should not output any rFonts element for the `Heading 1` style definition, because that formatting is inherited from the document defaults. *end example*]

2.7.2 Style Hierarchy

With the various flavors of styles available (see each of the subclauses below), multiple style types can be applied to the same content within a file, which means that properties must be applied in a specific deterministic order. As with inheritance, the resulting formatting properties set by one type can be unchanged, removed, or altered by following types.

The following table illustrates the order of application of these defaults, and which properties are impacted by each:



This process can be described as follows:

- First, the document defaults are applied to all runs and paragraphs in the document.
- Next, the table style properties are applied to each table in the document, following the conditional formatting inclusions and exclusions specified per table.
- Next, numbered item and paragraph properties are applied to each paragraph formatted with a numbering style.

- Next, paragraph and run properties are applied to each paragraph as defined by the paragraph style.
- Next, run properties are applied to each run with a specific character style applied.
- Finally, we apply direct formatting (paragraph or run properties not from styles). If this direct formatting includes numbering, that numbering + the associated paragraph properties are applied.

2.7.3 General Style Properties

General style properties refer to the set of properties which can be used regardless of the type of style. [*Example*: Within a style definition the style name, additional aliases for the style, a style ID (used by the document content to refer to the style), if style is hidden, if style is locked, etc. are general style properties. *end example*]

[*Example:* Consider a style called Heading 1 in a document as follows:

```
<w:style w:type="paragraph" w:styleId="Heading1">
  <w:name w:val="heading 1"/>
  <w:basedOn w:val="Normal"/>
  <w:next w:val="Normal"/>
  <w:link w:val="Heading1Char"/>
  <w:priority w:val="1"/>
  <w:qformat/>
  <w:rsid w:val="00F303CE"/>
  ...
</w:style>
```

Above the formatting information specific to this style type are a set of general style properties which define information shared by all style types. *end example*]

2.7.3.1 aliases (Alternate Style Names)

This element specifies the set of alternative names for the parent style definition. These names may be used in an application's user interface as desired. The alternate names shall be stored in this element's val attribute, and each name shall be separated by one or more consecutive comma characters (Unicode character value 002C). All commas present shall be interpreted as separator character and never as part of an alternate style name.

If present, the alternate style names shall be used in the user interface in place of the built-in name specified in the name element (§2.7.3.9) when the appropriate value is set in the stylePaneFormatFilter element (§2.15.1.87).

If this element is omitted, then the style shall not have any alternate style names.

[*Example*: Consider a style with a primary name and two alternate names, defined using the name and aliases elements, as follows:

```
<w:style w:styleId="TestStyle" … >
  <w:name w:val="GD20Complex"/>
  <w:aliases w:val="Regional Growth,Complex Growth"/>
  …
</w:style>
```

This style specifies that it has the primary name `GD20Complex` using the name element (§2.7.3.9), as well as two alternate names `Regional Growth` and `Complex Growth` using the aliases element. *end example*]

Parent Elements
style (§2.7.3.17)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.7.3.2 autoRedefine (Automatically Merge User Formatting Into Style Definition)

This element specifies whether an application shall automatically modify this style when the contents of an entire paragraph in the document with this style applied are modified, ensuring that although only a single instance of text with this style was modified, that change is stored on the style and therefore propagated to all locations where the style is in use.

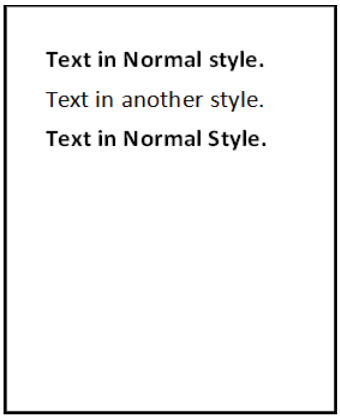
If this element is omitted, then formatting shall not automatically be merged back into the style definition.

[*Example*: Consider a style defined as follows in a WordprocessingML document:

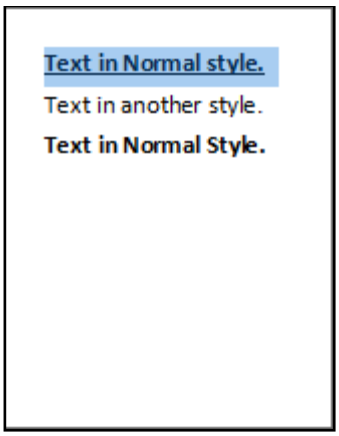
```
<w:style w:styleId="Normal" … >
  <w:name w:val="Normal"/>
  <w:autoRedefine/>
  <w:rPr>
    <w:b/>
  </w:rPr>
  …
</w:style>
```

This style specifies via the use of the autoRedefine element that any formatting applied to text which uses this style shall be merged back into the style definition (assuming, of course, that this is a paragraph style).

For example, consider a document which uses the `Normal` style as defined above:



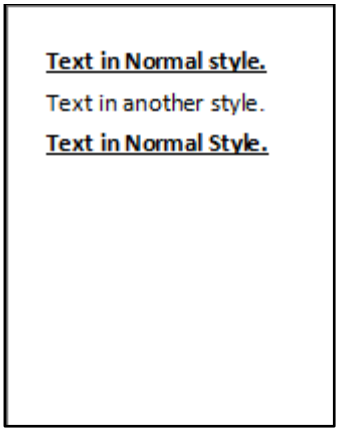
The first and third paragraphs use the `Normal` style, and hence have the bold property applied. If an application were to add the underline formatting to the entire first paragraph, as follows:



That property, rather than being saved as direct formatting, shall be used to update the associated `Normal` style to add this property, specified using the `u` element (§2.3.2.38).

```
<w:style w:styleId="Normal" … >
  <w:name w:val="Normal"/>
  <w:autoRedefine/>
  <w:rPr>
    <w:b/>
    <w:u/>
  </w:rPr>
  …
</w:style>
```

Since this property is automatically merged into the style, it would also appear on the third paragraph (note that the step above would normally be automatically modified into the state shown below, and not discrete as shown above).



end example]

Parent Elements
style (§2.7.3.17)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.7.3.3 basedOn (Parent Style ID)

This element specifies the style ID of the parent style from which this style inherits in the style inheritance. The *style inheritance* refers to a set of styles which inherit from one another to produce the resulting set of properties for a single style. The val attribute of this element specifies the styleId attribute for the parent style in the style inheritance.

If this element is omitted, then this style shall not be based on any other style in the current document (i.e. this element is the root of the style inheritance for a style). If no style in the current document specifies the styleId present in the val attribute, then this element shall be ignored (i.e. this element is the root of the style inheritance for a style).

If a style with this styleId is present, then it shall be subject to the following restrictions:

- If the current style is a table style, then the parent style must also be a table style, or this element shall be ignored.
- If the current style is a paragraph style, then the parent style must also be a paragraph style, or this element shall be ignored.
- If the current style is a character style, then the parent style must also be a character style, or this element shall be ignored.
- If the current style is a numbering style, then this element shall be ignored.

[*Example*: Consider three WordprocessingML character styles defined as follows:

- A character style with a styleId value of Strong whose properties consist of the bold property
- A character style with a styleId value of Underline whose properties consist of the underline property
- A character style with a styleId value of Emphasis whose properties consist of the italics property

Each of these character styles defines a single character formatting property. If the basedOn values for each element were defined as follows:

```
<w:style w:styleId="Strong">
  <w:basedOn w:val="Underline"/>
  …
  <w:rPr>
    <w:b/>
  </w:rPr>
</w:style>
<w:style w:styleId="Underline">
  <w:basedOn w:val="Emphasis"/>
  …
  <w:rPr>
    <w:u/>
  </w:rPr>
</w:style>
<w:style w:styleId="Emphasis">
  …
  <w:rPr>
    <w:i/>
  </w:rPr>
</w:style>
```

The Strong style is based on the Underline style which is in turn based on the Emphasis style. This means that the actual definition of the Strong style would be as follows:

- Bold
- Underline (inherited from Underline)
- Italics (inherited from Emphasis)

The style chain for the Strong style would be defined as follows:

- Emphasis
- Underline
- Strong

Similarly, the style chain for the Underline style would be defined as follows:

- Emphasis

- `Underline`

In each case, the style chain is the list of all styles which are combined in order to produce the entire set of properties for any given style. *end example*]

Parent Elements
style (§2.7.3.17)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.7.3.4 hidden (Hide Style From User Interface)

This element specifies whether this style shall be hidden from any and all user interfaces when this document is loaded by an application. If this element is set, then this style may be used to format content (i.e. any content which references this style shall have its properties as normal), but the style shall be hidden from all user

interface associated with that application. [*Note*: This setting is typically used to hide styles which are being used internally by an application which should not be used as formatting in a typical case. *end note*]

If this element is omitted, then the style shall not be required to be hidden from the user interface.

[*Example*: Consider a style with a primary name of `InternalStyle` that should not be displayed in any user interface. This requirement would be specified using the following WordprocessingML:

```
<w:style … w:styleId="Style2">
  <w:name w:val="InternalStyle"/>
  <w:hidden/>
  …
</w:style>
```

The hidden element specifies that this style definition shall be round-tripped with the file (since it is part of the document) but should not be displayed in any user interface associated with an application which processes this document. *end example*]

Parent Elements
style (§2.7.3.17)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.7.3.5 latentStyles (Latent Style Information)

This element specifies the properties which shall be applied to a set of latent styles for this document. *Latent styles* refer to any set of style definitions known to an application which have not been included in the current document. [*Example*: Latent styles may include additional styles known by a particular hosting application. *end example*]

When a style definition is embedded in a document, it specifies two distinct groups of properties:

- Behavior properties
- Formatting properties

Obviously, embedding all the styles known to a particular application in each document which it produces would drastically increase the file size. Latent styles provide a way to store pieces of information for the first group (behavior properties) which must be specified for all styles known to an application without requiring the storage of the second group (formatting properties).

[*Example*: Consider a WordprocessingML document which contains text specified in one of two styles: `Heading1` or `Normal`. Based on this, the document only needs to store the formatting properties for those two styles, saving the additional overhead which would be required to save all of the styles supported by the hosting application.

However, if the documentProtection element (§2.15.1.28) specifies that the hosting application shall prevent the use of any style whose locked element (§2.7.3.7) is set to `false`, then the locking state of all styles known to that application become useful and necessary to maintain the current state of the document. Using latent styles, this information may be stored without storing any formatting properties for those styles.

For example, if all styles which are not stored in the document shall be locked except for the style with a primary name (§2.7.3.9) of `Heading 2`. This requirement would be specified using latent styles as follows:

```
<w:latentStyles … w:defLockedState="true">
  <w:lsdException w:name="Heading 2" w:locked="false"/>
</w:latentStyles>
```

The latentStyles element specifies that all latent styles known to any hosting application shall have a default locking state of `true` except for any style known to the hosting application with a primary name of `Heading 2`, whose latent style definition specifies that its locked state shall be `false`. *end example*]

Parent Elements
styles (§2.7.3.18)

Child Elements	Subclause
lsdException (Latent Style Exception)	§2.7.3.8

Attributes	Description

count (Latent Style Count)	Specifies the number of known styles which shall be initialized to the current latent style defaults when this document is first processed. [*Note*: This property may be used by an application as needed to ensure that only the number of styles known when this document was created are initialized with the defaults on the parent element, and that all new known styles use their default values. *end note*]
	
	[*Example*: Consider a WordprocessingML document in which only the first 20 latent styles shall be initialized. This requirement would be specified as follows:
	
	`<w:latentStyles w:count="20" … >`
	`  …`
	`</w:latentStyles>`
	
	The count attribute specifies that 20 known styles shall be initialized to the default settings when the document is first opened, and any additional styles should use the defaults defined by the application. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

defLockedState (Default Style Locking Setting)	Specifies the default setting for the locked element (§2.7.3.7) which shall be applied to any style made available by the hosting application which is not explicitly defined in the current document. This setting shall be overridden for every style for which a latent style exception (§2.7.3.8) exists.
	
	If this element is omitted, the default locked state for all latent styles in the current document shall be false.
	
	[*Example*: Consider a WordprocessingML document in which all styles which are not stored in the document shall be locked. This requirement would be specified using latent styles as follows:
	
	`<w:latentStyles … w:defLockedState="true">`
	`  …`
	`</w:latentStyles>`
	
	The defLockedState attribute specifies that all latent styles in the current document shall have a locked element setting of true by default. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

defQFormat (Default Primary Style Setting)	Specifies the default setting for the qFormat element (§2.7.3.14) which shall be applied to any style made available by the hosting application which is not explicitly defined in the current document. This setting shall be overridden for every style for which a latent style exception (§2.7.3.8) exists. If this element is omitted, the default qFormat state for all latent styles in the current document shall be false.

Attributes	Description

	[*Example*: Consider a WordprocessingML document in which all styles which are not stored in the document shall not be marked as primary styles. This requirement would be specified using latent styles as follows:
	
	`<w:latentStyles … w:defQFormat="false">`
	`  …`
	`</w:latentStyles>`
	
	The defQFormat attribute specifies that all latent styles in the current document shall have a qFormat element setting of `false` by default. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

defSemiHidden (Default Semi-Hidden Setting)	Specifies the default setting for the semiHidden element (§2.7.3.16) which shall be applied to any style made available by the hosting application which is not explicitly defined in the current document. This setting shall be overridden for every style for which a latent style exception (§2.7.3.8) exists.
	
	If this element is omitted, the default semiHidden state for all latent styles in the current document shall be `false`.
	
	[*Example*: Consider a WordprocessingML document in which all styles which are not stored in the document shall not be marked as semi-hidden. This requirement would be specified using latent styles as follows:
	
	`<w:latentStyles … w:defSemiHIdden="false">`
	`  …`
	`</w:latentStyles>`
	
	The defSemiHidden attribute specifies that all latent styles in the current document shall have a semiHidden element setting of `false` by default. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

defUIPriority (Default User Interface Priority Setting)	Specifies the default setting for the uiPriority element (§2.7.3.19) which shall be applied to any style made available by the hosting application which is not explicitly defined in the current document. This setting shall be overridden for every style for which a latent style exception (§2.7.3.8) exists. If this element is omitted, the default uiPriority state for all latent styles in the current document shall be 99. [*Example*: Consider a WordprocessingML document in which all styles which are not stored in the document shall not be marked as semi-hidden. This requirement would be specified using latent styles as follows: `<w:latentStyles … w:defUIPriority="10">`

Attributes	Description
	… </w:latentStyles> The defUIPriority attribute specifies that all latent styles in the current document shall have a uiPriority element setting of 10 by default. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
defUnhideWhenUsed (Default Hidden Until Used Setting)	Specifies the default setting for the unhideWhenUsed element (§2.7.3.20) which shall be applied to any style made available by the hosting application which is not explicitly defined in the current document. This setting shall be overridden for every style for which a latent style exception (§2.7.3.8) exists. If this element is omitted, the default unhideWhenUsed state for all latent styles in the current document shall be false. [*Example*: Consider a WordprocessingML document in which all styles which are not stored in the document shall be hidden until they are used in the document's contents. This requirement would be specified using latent styles as follows: <w:latentStyles … w:defUnhideWhenUsed="true"> … </w:latentStyles> The defUnhideWhenUsed attribute specifies that all latent styles in the current document shall have a unhideWhenUsed element setting of true by default. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LatentStyles">
   <sequence>
      <element name="lsdException" type="CT_LsdException" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="defLockedState" type="ST_OnOff"/>
   <attribute name="defUIPriority" type="ST_DecimalNumber"/>
   <attribute name="defSemiHidden" type="ST_OnOff"/>
   <attribute name="defUnhideWhenUsed" type="ST_OnOff"/>
   <attribute name="defQFormat" type="ST_OnOff"/>
   <attribute name="count" type="ST_DecimalNumber"/>
</complexType>
```

2.7.3.6 link (Linked Style Reference)

This element specifies the pairing of styles which comprise a linked style. A *linked style* is a grouping of a paragraph style and character style which is used in a user interface to allow the same set of formatting properties to be applied:

- To the contents of one or more entire paragraphs (i.e. as a paragraph style)
- To the contents of one or more runs within a paragraph (i.e. as a character style)

Each style continues to exist independently in the file format as there is both a paragraph and character style present within the styles element (§2.7.3.18), however these two styles shall be merged into one and applied appropriately based on whether they are applied to run(s) or paragraph(s), by referencing the styleId attribute of the paired linked style via this element's val attribute.

If this element is omitted, then this style is not part of a linked style pairing. If no style in the current document specifies the styleId present in the val attribute, then this element shall be ignored.

If a style with this styleId is present, then it shall be subject to the following restrictions:

- If the parent style is a table style, then this element shall be ignored.
- If the parent style is a paragraph style, then the parent style must be a character style, or this element shall be ignored.
- If the parent style is a character style, then the parent style must be a paragraph style, or this element shall be ignored.
- If the parent style is a numbering style, then this element shall be ignored.

[*Example*: Consider a linked style defined as follows in a WordprocessingML document:

```
<w:style w:type="paragraph" w:styleId="TestParagraphStyle">
  <w:link w:val="TestCharacterStyle"/>
  …
</w:style>
<w:style w:type="character" w:styleId="TestCharacterStyle">
  <w:link w:val="TestParagraphStyle"/>
  …
</w:style>
```

This pairing of a paragraph style and a character style are linked via the link element, which is used the reference the styleId of the paragraph style from the character style definition and vice versa. Because this pairing is valid based on the rules above, the resulting combination shall be used as a linked style, which appears as one style in an application, but uses the character and/or paragraph style as appropriate. *end example*]

Parent Elements
style (§2.7.3.17)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.7.3.7 locked (Style Cannot Be Applied)

This element specifies whether an application shall prevent the use of this style when this document is loaded and/or modified. If this element is set, then this style may be used to format existing content (i.e. any content which references this style shall have its properties as normal), but new instances of the style shall be prevented from being applied via all mechanisms associated with that application.

If this element is omitted, then the use of the style shall not be prevented by an application processing this document.

[*Example*: Consider a style with a primary name of `Test Style` which should be locked, and prevented from being added to any content in a given document. This requirement would be specified using the following WordprocessingML:

```
<w:style … w:styleId="TestStyle">
  <w:name w:val="Test Style"/>
  <w:locked/>
  …
</w:style>
```

The presence of the locked element specifies that this style definition shall new instances of the style shall be prevented from being applied via all mechanisms associated with that application. *end example*]

Parent Elements
style (§2.7.3.17)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.7.3.8 lsdException (Latent Style Exception)

This element specifies the properties which shall be applied a single latent style for this document. *Latent styles* refer to any set of known style definitions which have not been included in the current document.

[*Example*: Consider a WordprocessingML document which contains text specified in one of two styles: Heading1 or Normal. Based on this, the document only needs to store the formatting properties for those two styles, saving the additional overhead which would be required to save all of the styles supported by the hosting application.

However, if the documentProtection element (§2.15.1.28) specifies that the hosting application shall prevent the use of any style whose locked element (§2.7.3.7) is set to `false`, then the locking state of all styles known to that application become useful and necessary to maintain the current state of the document. Using latent styles, this information may be stored without storing any formatting properties for those styles.

For example, if all styles which are not stored in the document shall be locked except for the style with a primary name (§2.7.3.9) of `Heading 2`. This requirement would be specified using latent styles as follows:

```
<w:latentStyles … w:defLockedState="true">
  <w:lsdException w:name="Heading 2" w:locked="false"/>
</w:latentStyles>
```

The lsdException element specifies that the latent style with a primary name of `Heading 2` shall have a locked state setting of `false`. *end example*]

Parent Elements
latentStyles (§2.7.3.5)

Attributes	Description

locked (Latent Style Locking Setting)	Specifies the default setting for the locked element (§2.7.3.7) which shall be applied to the latent style with the matching style name value.
	
	If this element is omitted, the default locked state for this latent style shall be determined by the defLockedState attribute on the parent latentStyles element.
	
	[*Example*: Consider a WordprocessingML document in which all styles which are not stored in the document shall be locked except for the `TestStyle` style. This requirement would be specified using latent styles as follows:
	
	`<w:latentStyles … w:defLockedState="true">`
	`  <w:lsdException w:name="TestStyle" w:locked="false"/>`
	`</w:latentStyles>`
	
	The locked attribute on the latent style exception specifies that the `TestStyle` style shall have a locked element setting of `false` by default. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

name (Primary Style Name)	Specifies the primary name for the style which shall inherit this set of latent style property exceptions. If the current application does not know of an internal primary style with the current name, then this set of latent style exceptions may be ignored. [*Example*: Consider a WordprocessingML document in which all styles which are not stored in the document shall be locked except for the `TestStyle` style. This requirement

Attributes	Description

	would be specified using latent styles as follows:
	
	`<w:latentStyles … w:defLockedState="true">`
	`  <w:lsdException w:name="TestStyle" w:locked="false"/>`
	`</w:latentStyles>`
	
	The name attribute on the latent style exception specifies that the `TestStyle` style shall have this set of latent style properties (if the application knows of a style with this name). *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

qFormat (Latent Style Primary Style Setting)	Specifies the default setting for the qFormat element (§2.7.3.14) which shall be applied to the latent style with the matching style name value.
	
	If this element is omitted, the default qFormat state for this latent style shall be determined by the defQFormat attribute on the parent latentStyles element.
	
	[*Example*: Consider a WordprocessingML document in which all styles which are not stored in the document shall not be primary styles except for the `TestStyle` style. This requirement would be specified using latent styles as follows:
	
	`<w:latentStyles … w:defQFormat="false">`
	`  <w:lsdException w:name="TestStyle" w:qFormat="true"/>`
	`</w:latentStyles>`
	
	The qFormat attribute on the latent style exception specifies that the `TestStyle` style shall have a qFormat element setting of `true` by default. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

semiHidden (Semi hidden text override)	Specifies the default setting for the semiHidden element (§2.7.3.16) which shall be applied to the latent style with the matching style name value.
	
	If this element is omitted, the default semiHidden state for this latent style shall be determined by the defSemiHidden attribute on the parent latentStyles element.
	
	[*Example*: Consider a WordprocessingML document in which all styles which are not stored in the document shall not be semi-hidden except for the `TestStyle` style. This requirement would be specified using latent styles as follows:
	
	`<w:latentStyles … w:defSemiHidden="false">`
	`  <w:lsdException w:name="TestStyle" w:semiHidden="true"/>`
	`</w:latentStyles>`
	
	The semiHidden attribute on the latent style exception specifies that the `TestStyle` style shall have a semiHidden element setting of `true` by default. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

uiPriority (Override default sorting order)	Specifies the default setting for the uiPriority element (§2.7.3.19) which shall be applied to the latent style with the matching style name value.
	
	If this element is omitted, the default uiPriority state for this latent style shall be determined by the defUIPriority attribute on the parent latentStyles element.
	
	[*Example*: Consider a WordprocessingML document in which all styles which are not stored in the document shall have a priority value of 10 except for the TestStyle style. This requirement would be specified using latent styles as follows:
	
	`<w:latentStyles … w:defUIPriority="10">`
	`  <w:lsdException w:name="TestStyle" w:uiPriority="25"/>`
	`</w:latentStyles>`
	
	The uiPriority attribute on the latent style exception specifies that the TestStyle style shall have a uiPriority element setting of 25 by default. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

unhideWhenUsed (Unhide when used)	Specifies the default setting for the unhideWhenUsed element (§2.7.3.20) which shall be applied to the latent style with the matching style name value.
	
	If this element is omitted, the default unhideWhenUsed state for this latent style shall be determined by the defUnhideWhenUsed attribute on the parent latentStyles element.
	
	[*Example*: Consider a WordprocessingML document in which all styles are to be hidden until used except for the TestStyle style. This requirement would be specified using latent styles as follows:
	
	`<w:latentStyles … w:defUnhideWhenUsed="true">`
	`  <w:lsdException w:name="TestStyle" w:unhideWhenUsed="false"/>`
	`</w:latentStyles>`
	
	The unhideWhenUsed attribute on the latent style exception specifies that the TestStyle style shall have an unhideWhenUsed element setting of false by default. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LsdException">
    <attribute name="name" type="ST_String" use="required"/>
    <attribute name="locked" type="ST_OnOff"/>
    <attribute name="uiPriority" type="ST_DecimalNumber"/>
    <attribute name="semiHidden" type="ST_OnOff"/>
    <attribute name="unhideWhenUsed" type="ST_OnOff"/>
    <attribute name="qFormat" type="ST_OnOff"/>
</complexType>
```

2.7.3.9 name (Primary Style Name)

This element specifies the primary name for the current style in the document. This name may be used in an application's user interface as desired. The actual primary name for this style is stored in its val attribute.

If present, the alternate style names (§2.7.3.1) shall be used in the user interface in place of the built-in name specified when the appropriate value is set in the stylePaneFormatFilter element (§2.15.1.86).

If this element is omitted, then the style shall not have a primary style name.

[*Example*: Consider a style with a primary name and two alternate names, defined using the name and aliases elements, as follows:

```
<w:style w:styleId="TestStyle" … >
  <w:name w:val="GD20Complex"/>
  <w:aliases w:val="Regional Growth,Complex Growth"/>
  …
</w:style>
```

This style specifies that it has the primary name GD20Complex using the name element. *end example*]

Parent Elements
style (§2.7.3.17)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`

Attributes	Description

	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.7.3.10 next (Style For Next Paragraph)

This element specifies the style which shall automatically be applied to a new paragraph created following a paragraph with the parent paragraph style applied. [*Note*: This setting is typically used when the use of the current style is limited to one paragraph at most, and it would typically be undesirable to apply this style to following paragraphs - for example, a title style may specify that its following paragraphs shall return to regular text formatting. *end note*]

If this element is specified on a style of any type other than a paragraph style, this element shall be ignored. If no style whose styleId matches the val attribute of this element exists or that style is not a paragraph style, this element shall be ignored.

If this element is omitted, then the following paragraph shall use the same paragraph style as the current paragraph.

[*Example*: Consider a style defined as follows in a WordprocessingML document:

```
<w:style w:styleId="TestParagraphStyle" … >
  <w:name w:val="Test Paragraph Style"/>
  <w:next w:val="AnotherParagraphStyle"/>
  <w:rPr>
    <w:b/>
  </w:rPr>
  …
</w:style>
```

This style specifies via the use of the next element that the style for the next paragraph in the document shall be the paragraph style whose styleId attribute value is `AnotherParagraphStyle` (if such a paragraph style exists). *end example*]

Parent Elements
style (§2.7.3.17)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.7.3.11 personal (E-Mail Message Text Style)

This element specifies that the parent style, when in use in the context of an e-mail message, was used by default to format all message text from one or more users. [*Note*: This setting does not provide any additional semantic about the style, but may be used in the context of e-mail to automatically reformat the contents of the e-mail message while ignoring any content to which styles were deliberately applied (since this style was implicitly applied to message text without user interaction). *end note*]

If this element is specified on a style of any type other than a character style, this element shall be ignored. If no style whose styleId matches the val attribute of this element exists or that style is not a character style, this element shall be ignored.

If this element is omitted, then the current style shall not be considered a message text style in the context of e-mail messages.

[*Example*: Consider a style defined as follows in a WordprocessingML document:

```
<w:style w:styleId="EmailText" w:type="character" >
  <w:name w:val="EmailText"/>
  <w:personal w:val="true" />
  <w:rPr>
    …
  </w:rPr>
</w:style>
```

This style specifies via the use of the personal element that this style is a style used to format message text in the context of e-mail. *end example*]

Parent Elements
style (§2.7.3.17)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.7.3.12 personalCompose (E-Mail Message Composition Style)

This element specifies that the parent style, when in use in the context of an e-mail message, may be used by default to format new message text within the e-mail message. [*Note*: This setting does not provide any additional semantic about the style, but may be used in the context of e-mail to automatically format the contents of new test in the e-mail message. *end note*]

If this element is specified on a style of any type other than a character style, this element shall be ignored. If no style whose styleId matches the val attribute of this element exists or that style is not a character style, this element shall be ignored.

If this element is omitted, then the current style shall not be considered a message composition text style in the context of e-mail messages.

[*Example*: Consider a style defined as follows in a WordprocessingML document:

```
<w:style w:styleId="EmailText" w:type="character" >
  <w:name w:val="EmailText"/>
  <w:personalCompose w:val="true" />
  <w:rPr>
    …
  </w:rPr>
</w:style>
```

This style specifies via the use of the personalCompose element that this style is a style used to format new message text in the context of e-mail. *end example*]

Parent Elements
style (§2.7.3.17)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.7.3.13 personalReply (E-Mail Message Reply Style)

This element specifies that the parent style, when in use in the context of an e-mail message, may be used by default to format existing message text within the e-mail message when a new reply is generated. [*Note*: This setting does not provide any additional semantic about the style, but may be used in the context of e-mail to automatically format the contents of existing test in the e-mail message. *end note*]

If this element is specified on a style of any type other than a character style, this element shall be ignored. If no style whose styleId matches the val attribute of this element exists or that style is not a character style, this element shall be ignored.

If this element is omitted, then the current style shall not be considered a message reply text style in the context of e-mail messages.

[*Example*: Consider a style defined as follows in a WordprocessingML document:

```
<w:style w:styleId="EmailText" w:type="character" >
  <w:name w:val="EmailText"/>
  <w:personalReply w:val="true" />
  <w:rPr>
    …
  </w:rPr>
</w:style>
```

This style specifies via the use of the personalReply element that this style is a style used to format existing message text in the context of e-mail. *end example*]

Parent Elements
style (§2.7.3.17)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted.

Attributes	Description
	A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.7.3.14 qFormat (Primary Style)

This element specifies whether this style shall be treated as a primary style when this document is loaded by an application. If this element is set, then this style has been designated as being particularly important for the current document, and this information may be used by an application in any means desired. [*Note*: This setting does not imply any behavior for the style, only that the style is of particular significance for this document. *end note*]

If this element is omitted, then the style shall not be considered a primary style for this document.

[*Example*: Consider a style with a primary name of `PrimaryStyleExample` that should be treated as a primary style for the document. This requirement would be specified using the following WordprocessingML:

```
<w:style … w:styleId="PStyle">
  <w:name w:val="PrimaryStyleExample"/>
  <w:qFormat/>
  …
</w:style>
```

The qFormat element specifies that this style definition shall be treated as a primary style for this document. *end example*]

Parent Elements
style (§2.7.3.17)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element.

Attributes	Description
	A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.7.3.15 rsid (Revision Identifier for Style Definition)

This element specifies a unique four digit number which shall be used to determine the editing session in which this style definition was last modified. This value shall follow this following constraint: All document elements which specify the same rsid* values must correspond to changes made during the same editing session. An *editing session* is defined as the period of editing which takes place between any two subsequent save actions. [*Note*: This setting does not imply any behavior for the style, only that the style was last modified during one particular editing session. This information may be interpreted by an application in any manner desired. *end note*]

If this element is omitted, then no revision identifier shall be associated with the parent style definition.

[*Example*: Consider a style with a primary name of PrimaryStyleExample that is defined as follows:

```
<w:style … w:styleId="PStyle">
  <w:name w:val="PrimaryStyleExample"/>
  <w:rsid w:val="3E412D01"/>
  …
</w:style>
```

The rsid element specifies that this style definition was last edited in the editing session corresponding to the value 3E412D01. *end example*]

Parent Elements
style (§2.7.3.17)

Attributes	Description
val (Long Hexadecimal Number Value)	Specifies a number value specified as a four digit hexadecimal number), whose contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following value for an attribute of type ST_LongHexNumber: `00BE2C6C`. This value is valid, as it contains four hexadecimal digits, each an encoding of an octet of the actual decimal number value. It may therefore be interpreted as desired in the context of the parent XML element, *end example*] The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LongHexNumber">
   <attribute name="val" type="ST_LongHexNumber" use="required"/>
</complexType>
```

2.7.3.16 semiHidden (Hide Style From Main User Interface)

This element specifies whether this style shall be hidden from the main user interface when this document is loaded by an application. If this element is set, then this style may be used to format content (i.e. any content which references this style shall have its properties as normal), but the style shall be hidden from the main user interface associated with that application.

[*Note*: The interpretation of a "main" user interface shall not be dictated by this Office Open XML Standard, and may be defined by an application as appropriate.

This setting is intended to define a style property which allows styles to be seen and modified in an advanced user interface, without exposing the style in a less advanced setting, for example, the style which is used to format the contents of a comment should typically not be shown in a simple user interface (as it is uncommon to want to modify it), but would be inappropriate to hide completely using the hidden element (§2.7.3.4), as very advanced users may want to change its appearance. *end note*]

If this element is omitted, then the style shall not be required to be hidden from the main user interface.

[*Example*: Consider a style with a primary name of `Comment Style` that should not be displayed in the main user interface. This requirement would be specified using the following WordprocessingML:

```
<w:style … w:styleId="CStyle">
  <w:name w:val="Comment Style"/>
  <w:semiHidden/>
  …
</w:style>
```

The semiHidden element specifies that this style definition should not be displayed in any main user interface associated with an application which processes this document. *end example*]

Parent Elements
style (§2.7.3.17)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.7.3.17 style (Style Definition)

This element specifies the definition of a single style within a WordprocessingML document. A *style* is a predefined set of table, numbering, paragraph, and/or character properties which can be applied to regions within a document.

The style definition for any style definition can be divided into three segments:

- General style properties
- Style type
- Type-specific properties

General style properties refers to the set of properties which can be used regardless of the type of style; for example, the style name, additional aliases for the style, a style ID (used by the document content to refer to the style), if style is hidden, if style is locked, etc.

[*Example:* Consider a style called Heading 1 in a document as follows:

```
<w:style w:type="paragraph" w:styleId="Heading1">
  <w:name w:val="Heading 1"/>
  <w:basedOn w:val="Normal"/>
  <w:next w:val="Normal"/>
  <w:link w:val="Heading1Char"/>
  <w:priority w:val="1"/>
  <w:qformat/>
  <w:rsid w:val="00F303CE"/>
  …
</w:style>
```

Above the formatting information specific to this style type are a set of general style properties which define information shared by all style types. *end example*]

Style types refers to the property on a style which defines the type of style created with this style definition. WordprocessingML supports six types of style definitions by the values for the style definition's type attribute:

- Paragraph styles
- Character styles
- Linked styles (paragraph + character) [*Note*: Accomplished via the link element (§2.7.3.6). *end note*]
- Table styles
- Numbering styles
- Default paragraph + character properties

[*Example:* Consider a style called Heading 1 in a document as follows:

```
<w:style w:type="paragraph" w:styleId="Heading1">
  <w:name w:val="heading 1"/>
  <w:basedOn w:val="Normal"/>
  <w:next w:val="Normal"/>
  <w:link w:val="Heading1Char"/>
  <w:priority w:val="1"/>
  <w:qformat/>
  <w:rsid w:val="00F303CE"/>
  …
</w:style>
```

The type attribute has a value of paragraph, which indicates that the following style definition is a paragraph style. *end example*]

Type-specific properties refers to the payload of the style: its formatting information as well as any properties which apply only to that type of style.

[*Example*: Consider a table style with primary name Normal Table defined as follows:

```
<w:style w:type="table" w:default="1" w:styleId="TableNormal">
  <w:name w:val="Normal Table"/>
  …
  <w:tblPr>
    <w:tblInd w:w="0" w:type="dxa"/>
    <w:tblCellMar>
      <w:top w:w="0" w:type="dxa"/>
      <w:left w:w="108" w:type="dxa"/>
      <w:bottom w:w="0" w:type="dxa"/>
      <w:right w:w="108" w:type="dxa"/>
    </w:tblCellMar>
  </w:tblPr>
</w:style>
```

The tblPr element contains the formatting payload for this table style, which is only applicable to a table style. *end example*]

Parent Elements
styles (§2.7.3.18)

Child Elements	Subclause
aliases (Alternate Style Names)	§2.7.3.1
autoRedefine (Automatically Merge User Formatting Into Style Definition)	§2.7.3.2
basedOn (Parent Style ID)	§2.7.3.3
hidden (Hide Style From User Interface)	§2.7.3.4
link (Linked Style Reference)	§2.7.3.6
locked (Style Cannot Be Applied)	§2.7.3.7
name (Primary Style Name)	§2.7.3.9
next (Style For Next Paragraph)	§2.7.3.10
personal (E-Mail Message Text Style)	§2.7.3.11
personalCompose (E-Mail Message Composition Style)	§2.7.3.12
personalReply (E-Mail Message Reply Style)	§2.7.3.13
pPr (Style Paragraph Properties)	§2.7.7.2
qFormat (Primary Style)	§2.7.3.14
rPr (Run Properties)	§2.7.8.1
rsid (Revision Identifier for Style Definition)	§2.7.3.15
semiHidden (Hide Style From Main User Interface)	§2.7.3.16
tblPr (Style Table Properties)	§2.7.5.4

Child Elements	Subclause
tblStylePr (Style Conditional Table Formatting Properties)	§2.7.5.6
tcPr (Style Table Cell Properties)	§2.7.5.8
trPr (Style Table Row Properties)	§2.7.5.11
uiPriority (Optional User Interface Sorting Order)	§2.7.3.19
unhideWhenUsed (Remove Semi-Hidden Property When Style Is Used)	§2.7.3.20

Attributes	Description
customStyle (User-Defined Style)	Specifies that this style is a user-defined style (i.e. it is not a style which was automatically generated by an application). This setting (specifically a value of `true` or its equivalents) shall not allow the formatting associated with the style to be changed automatically by an application, but may be used to specify that if the associated style ID is known, certain user interface behaviors may be applied to its definition. [*Example*: The style's primary name may be localized to match the current user interface language. *end example*] If this attribute is omitted, then the style shall be assumed to be a built-in style. [*Example*: Consider a paragraph style defined as follows: `<w:style w:type="paragraph" w:styleId="MyStyle" w:customStyle="true"> <w:name w:val="My Paragraph Style"/> <w:rPr> <w:b/> </w:rPr> </w:style>` This paragraph style specifies that it is a user-defined style using the customStyle attribute's value of `true`. An application may therefore take action on the style if it has behaviors associated with the style ID `MyStyle`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
default (Default Style)	Specifies that this style is the default for this type of style. This property is used in conjunction with the type attribute to determine the style which is applied to objects that do not explicitly declare a style. [*Example*: The paragraph style with the default attribute set is the paragraph style applied to all paragraphs which do not explicitly reference a paragraph style using the pStyle element (§2.3.1.27). *end example*] If this attribute is not specified for any style, then no properties shall be applied to objects of the specified type. If this attribute is specified by multiple styles, then the last instance of a style with this property shall be used.

Attributes	Description
	[*Example*: Consider a paragraph style defined as follows: `<w:style w:type="paragraph" w:default="1" w:styleId="MyStyle" > <w:name w:val="My Paragraph Style"/> <w:rPr> <w:b/> </w:rPr> </w:style>` This paragraph style specifies that it is the default paragraph style, and therefore all paragraphs which do not explicitly reference a paragraph style shall have this style applied. For example, consider the following paragraphs from the same WordprocessingML document: `<w:p> <w:pPr> <w:pStyle w:val="Normal"/> </w:pPr> … </w:p> <w:p> … </w:p>` The contents of the first paragraph shall have the `Normal` paragraph style applied to them, while the contents of the second paragraph shall have the `MyStyle` paragraph style applied, since it does not explicitly reference a paragraph style and therefore inherits the default. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
styleId (Style ID)	Specifies a unique identifier for the parent style definition. This identifier shall be used in multiple contexts to uniquely reference this style definition within the document. [*Example*: The following are examples of elements which reference a style via its styleId attribute value: • To reference a style from content using elements like the pStyle element (§2.3.1.27), rStyle element (§2.3.2.27), and the tblStyle element (§2.4.59) for paragraphs, runs, and tables, respectively. • To link the paragraph and character versions of a style via the link element (§2.7.3.6) • To reference the parent style for style inheritance via the basedOn element (§2.7.3.3) *end example*]

Attributes	Description
	If multiple style definitions each declare the same value for their styleId, then the first such instance shall keep its current identifier with all other instances being reassigned in any manner desired. This reassignment shall not require references to those style definitions to be 'repaired' in the content (i.e. some content may lose its style definition information, since the document was ill-formed). If this attribute is not specified, then a style ID may be assigned in any manner desired. [*Example*: Consider a paragraph style defined as follows: `<w:style w:type="paragraph" w:styleId="MyStyle" > <w:name w:val="My Paragraph Style"/> <w:rPr> <w:b/> </w:rPr> </w:style>` This paragraph style specifies that its style identifier shall be `MyStyle` using the styleId attribute. Now consider the following paragraphs from the same WordprocessingML document: `<w:p> <w:pPr> <w:pStyle w:val="MyStyle"/> </w:pPr> … </w:p> <w:p> … </w:p>` The contents of the first paragraph shall have the bold paragraph property applied to them because their paragraph properties specify that they shall inherit the paragraph style whose styleId is `MyStyle` therefore inheriting its properties using the rules of the style hierarchy. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).
type (Style Type)	Specifies the type of style definition defined by this element. WordprocessingML supports six types of style definitions: • Paragraph styles • Character styles • Table styles • Numbering styles • Linked styles (paragraph + character)

Attributes	Description

	• Default paragraph + character properties
	
	Each of the first four types corresponds to a different value in this attribute, and therefore defines the type of the current style. [*Note*: The last two types are unique in that they are not simply a style type: a linked style is a pairing of a character and paragraph style via the link element (§2.7.3.6); and the document default properties are defined via the docDefaults element (§2.7.4.1). *end note*]
	
	If this attribute is not specified, then the default value shall be assumed to be paragraph.
	
	[*Example*: Consider a style defined as follows:
	
	`<w:style w:type="paragraph" … >`
	`  <w:name w:val="My Paragraph Style"/>`
	`  <w:rPr>`
	`  <w:b/>`
	`  </w:rPr>`
	`</w:style>`
	
	The type attribute value of paragraph specifies that this style definition creates a paragraph style. *end example*]
	
	The possible values for this attribute are defined by the ST_StyleType simple type (§2.18.90).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Style">
   <sequence>
      <element name="name" type="CT_String" minOccurs="0" maxOccurs="1"/>
      <element name="aliases" type="CT_String" minOccurs="0"/>
      <element name="basedOn" type="CT_String" minOccurs="0"/>
      <element name="next" type="CT_String" minOccurs="0"/>
      <element name="link" type="CT_String" minOccurs="0"/>
      <element name="autoRedefine" type="CT_OnOff" minOccurs="0"/>
      <element name="hidden" type="CT_OnOff" minOccurs="0"/>
      <element name="uiPriority" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="semiHidden" type="CT_OnOff" minOccurs="0"/>
      <element name="unhideWhenUsed" type="CT_OnOff" minOccurs="0"/>
      <element name="qFormat" type="CT_OnOff" minOccurs="0"/>
      <element name="locked" type="CT_OnOff" minOccurs="0"/>
      <element name="personal" type="CT_OnOff" minOccurs="0"/>
      <element name="personalCompose" type="CT_OnOff" minOccurs="0"/>
      <element name="personalReply" type="CT_OnOff" minOccurs="0"/>
      <element name="rsid" type="CT_LongHexNumber" minOccurs="0"/>
      <element name="pPr" type="CT_PPr" minOccurs="0" maxOccurs="1"/>
      <element name="rPr" type="CT_RPr" minOccurs="0" maxOccurs="1"/>
      <element name="tblPr" type="CT_TblPrBase" minOccurs="0" maxOccurs="1"/>
      <element name="trPr" type="CT_TrPr" minOccurs="0" maxOccurs="1"/>
      <element name="tcPr" type="CT_TcPr" minOccurs="0" maxOccurs="1"/>
      <element name="tblStylePr" type="CT_TblStylePr" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="type" type="ST_StyleType" use="optional"/>
   <attribute name="styleId" type="ST_String" use="optional"/>
   <attribute name="default" type="ST_OnOff" use="optional"/>
   <attribute name="customStyle" type="ST_OnOff" use="optional"/>
</complexType>
```

2.7.3.18 styles (Style Definitions)

This element specifies all of the style information stored in the WordprocessingML document: style definitions as well as latent style information.

[*Example*: The Normal paragraph style in a word processing document can have any number of formatting properties, e.g. font face = Times New Roman; font size = 12pt; paragraph justification = left). All paragraphs which reference this paragraph style would automatically inherit these properties. *end example*]

Parent Elements
Root element of WordprocessingML Style Definitions part

Child Elements	Subclause
docDefaults (Document Default Paragraph and Run Properties)	§2.7.4.1
latentStyles (Latent Style Information)	§2.7.3.5
style (Style Definition)	§2.7.3.17

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Styles">
   <sequence>
      <element name="docDefaults" type="CT_DocDefaults" minOccurs="0"/>
      <element name="latentStyles" type="CT_LatentStyles" minOccurs="0" maxOccurs="1"/>
      <element name="style" type="CT_Style" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

2.7.3.19 uiPriority (Optional User Interface Sorting Order)

This element specifies a number which may be used to sort the set of style definitions in a user interface when this document is loaded by an application and the recommended setting is specified in the stylePaneSortMethod element (§2.15.1.87). If this element is set, then this priority shall be used to sort all available styles in ascending value order.

If this element is omitted, then the style shall not have an associated priority value and shall be sorted to the end of the list of style definitions (more or less equivalent to a priority value of infinity) when the recommended sort order setting is specified.

[*Example*: Consider a style with a primary name of `Comment Style` that should have an associated priority value of ten. This requirement would be specified using the following WordprocessingML:

```
<w:style … w:styleId="CStyle">
  <w:name w:val="Comment Style"/>
  <w:uiPriority w:val="10"/>
  …
</w:style>
```

The uiPriority element specifies that this style definition should be sorted into the list of styles using a value of `10` when the styles are listed in recommended order using the stylePaneSortMethod element (§2.15.1.87). *end example*]

Parent Elements
style (§2.7.3.17)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber:

Attributes	Description
	<w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.7.3.20 unhideWhenUsed (Remove Semi-Hidden Property When Style Is Used)

This element specifies whether the semiHidden property (§2.7.3.16) shall be removed when this style is used by the content of the document. If this element is set, then an application shall ensure that even if the semiHidden element is specified on a style, that this property is removed when the document is resaved if the style is referenced by any content in the document.

If this element is omitted, then the style shall not automatically lose the semi-hidden property when it is used in the document contents.

[*Example*: Consider a style with a primary name of `Test Paragraph Style` that should not be displayed in the main user interface until it is used. This requirement would be specified using the following WordprocessingML:

```
<w:style … w:styleId="TestStyle">
  <w:name w:val="Test Paragraph Style"/>
  <w:semiHidden/>
  <w:unhideWhenUsed/>
  …
</w:style>
```

The unhideWhenUsed element specifies that this style definition should not be displayed in any main user interface associated with an application which processes this document until it is referenced by document content. If a paragraph was added to the document which referenced this style:

```
<w:p>
  <w:pPr>
    <w:pStyle w:val="TestStyle"/>
  </w:pPr>
  …
</w:p>
```

This style is now referenced by the document's contents and would have the semiHidden element removed on save. *end example*]

Parent Elements
style (§2.7.3.17)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.7.4 Document Defaults

The first formatting information which is applied to all regions of text in a WordprocessingML document when that document is displayed is the document defaults. The document defaults specify the default set of properties which shall be inherited by every paragraph and run of text within all stories of the current WordprocessingML document. If no other formatting information was referenced by that text, these properties would solely define the formatting of the resulting text.

[*Example*: Consider the following fragment from the main document part of a WordprocessingML document:

```
<w:body>
  <w:p>
    <w:r>
      <w:t>Hello world!</w:t>
    </w:r>
  </w:p>
</w:body>
```

This paragraph and run of text both specify no formatting information (i.e. the paragraph and run possess neither a pPr element nor an rPr element respectively). Therefore, the only formatting applied to this text shall be the formatting in the document defaults which is applied to all paragraphs and runs in the document.

Note that this does not imply that these properties are only applied to text with no formatting - they are rather applied to all text before all other formatting (and in this case, there is no other formatting). *end example*]

2.7.4.1 docDefaults (Document Default Paragraph and Run Properties)

This element specifies the set of default paragraph and run properties which shall be applied to every paragraph and run in the current WordprocessingML document. These properties are applied first in the style hierarchy; therefore they are superseded by any further conflicting formatting, but apply if no further formatting is present.

If this element is omitted, then the document defaults shall be application-defined by the hosting application.

[*Example*: Consider the following definition for the document defaults for a WordprocessingML document:

```
<w:docDefaults>
  <w:pPrDefault>
    <w:pPr>
      <w:jc w:val="center"/>
    </w:pPr>
  </w:pPrDefault>
  <w:rPrDefault>
    <w:rPr>
      <w:b/>
    </w::rPr>
  </w:rPrDefault>
</w:docDefaults>
```

The child elements of docDefaults specify a default paragraph property of centered text and a default run property of bold text. Applying this formatting to the following fragment from the main document part of the same document:

```
<w:body>
  <w:p>
    <w:r>
      <w:t>Hello world!</w:t>
    </w:r>
  </w:p>
</w:body>
```

This paragraph contains no formatting properties, therefore, using the style hierarchy the document default paragraph and run properties are applied as specified within the docDefaults element and the resulting

paragraph is centered as specified in the jc element (§2.3.1.13) as well as bold as specified via the b element (§2.3.2.1). *end example*]

Parent Elements
styles (§2.7.3.18)

Child Elements	Subclause
pPrDefault (Default Paragraph Properties)	§2.7.4.3
rPrDefault (Default Run Properties)	§2.7.4.5

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DocDefaults">
   <sequence>
      <element name="rPrDefault" type="CT_RPrDefault" minOccurs="0"/>
      <element name="pPrDefault" type="CT_PPrDefault" minOccurs="0"/>
   </sequence>
</complexType>
```

2.7.4.2 pPr (Paragraph Properties)

This element specifies the set of paragraph properties which comprise the default paragraph properties for the current WordprocessingML document. [*Rationale*: The reason that a pPr element is present within the pPrDefault element is to allow for easy repurposing of any set of paragraph properties within a WordprocessingML document - since the paragraph properties are always child elements of a single pPr element, that element can simply be relocated in its entirety to the desired new location without additional modifications. *end rationale*]

If this element is omitted, then the default paragraph properties for the current document are non-existent (i.e. there are no default paragraph properties, and the defaults are therefore application-defined).

[*Example*: Consider the following definition for the document defaults for a WordprocessingML document:

```
<w:docDefaults>
  <w:pPrDefault>
    <w:pPr>
      <w:jc w:val="center"/>
    </w:pPr>
  </w:pPrDefault>
  …
</w:docDefaults>
```

The pPr element as a child of the pPrDefault element contains the set of default paragraph properties for this document - in this case, a justification value of `center`. *end example*]

Parent Elements
pPrDefault (§2.7.4.3)

Child Elements	Subclause
adjustRightInd (Automatically Adjust Right Indent When Using Document Grid)	§2.3.1.1
autoSpaceDE (Automatically Adjust Spacing of Latin and East Asian Text)	§2.3.1.2
autoSpaceDN (Automatically Adjust Spacing of East Asian Text and Numbers)	§2.3.1.3
bidi (Right to Left Paragraph Layout)	§2.3.1.6
cnfStyle (Paragraph Conditional Formatting)	§2.3.1.8
contextualSpacing (Ignore Spacing Above and Below When Using Identical Styles)	§2.3.1.9
divId (Associated HTML div ID)	§2.3.1.10
framePr (Text Frame Properties)	§2.3.1.11
ind (Paragraph Indentation)	§2.3.1.12
jc (Paragraph Alignment)	§2.3.1.13
keepLines (Keep All Lines On One Page)	§2.3.1.14
keepNext (Keep Paragraph With Next Paragraph)	§2.3.1.15
kinsoku (Use East Asian Typography Rules for First and Last Character per Line)	§2.3.1.16
mirrorIndents (Use Left/Right Indents as Inside/Outside Indents)	§2.3.1.18
numPr (Numbering Definition Instance Reference)	§2.3.1.19
outlineLvl (Associated Outline Level)	§2.3.1.20
overflowPunct (Allow Punctuation to Extent Past Text Extents)	§2.3.1.21
pageBreakBefore (Start Paragraph on Next Page)	§2.3.1.23
pBdr (Paragraph Borders)	§2.3.1.24
pPrChange (Revision Information for Paragraph Properties)	§2.13.5.31
pStyle (Referenced Paragraph Style)	§2.3.1.27
rPr (Run Properties for the Paragraph Mark)	§2.3.1.29
sectPr (Section Properties)	§2.6.19
shd (Paragraph Shading)	§2.3.1.31
snapToGrid (Use Document Grid Settings for Inter-Line Paragraph Spacing)	§2.3.1.32
spacing (Spacing Between Lines and Above/Below Paragraph)	§2.3.1.33
suppressAutoHyphens (Suppress Hyphenation for Paragraph)	§2.3.1.34
suppressLineNumbers (Suppress Line Numbers for Paragraph)	§2.3.1.35
suppressOverlap (Prevent Text Frames From Overlapping)	§2.3.1.36
tabs (Set of Custom Tab Stops)	§2.3.1.38
textAlignment (Vertical Character Alignment on Line)	§2.3.1.39

Child Elements	Subclause
textboxTightWrap (Allow Surrounding Paragraphs to Tight Wrap to Text Box Contents)	§2.3.1.40
textDirection (Paragraph Text Flow Direction)	§2.3.1.41
topLinePunct (Compress Punctuation at Start of a Line)	§2.3.1.43
widowControl (Allow First/Last Line to Display on a Separate Page)	§2.3.1.44
wordWrap (Allow Line Breaking At Character Level)	§2.3.1.45

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PPr">
   <complexContent>
      <extension base="CT_PPrBase">
         <sequence>
            <element name="rPr" type="CT_ParaRPr" minOccurs="0"/>
            <element name="sectPr" type="CT_SectPr" minOccurs="0"/>
            <element name="pPrChange" type="CT_PPrChange" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.7.4.3 pPrDefault (Default Paragraph Properties)

This element specifies the presence of a set of default paragraph properties for the current document. The actual paragraph properties are stored within the pPr child element of the current element.

If this element is omitted, then the default paragraph properties for the current document are non-existent (i.e. there are no default paragraph properties in the document, and the defaults are therefore application-defined).

[*Example*: Consider the following definition for the document defaults for a WordprocessingML document:

```
<w:docDefaults>
  <w:pPrDefault>
    <w:pPr>
      <w:jc w:val="center"/>
    </w:pPr>
  </w:pPrDefault>
  …
</w:docDefaults>
```

The pPrDefault element is a container for the set of default paragraph properties for this document. *end example*]

Parent Elements
docDefaults (§2.7.4.1)

Child Elements	Subclause
pPr (Paragraph Properties)	§2.7.4.2

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PPrDefault">
   <sequence>
      <element name="pPr" type="CT_PPr" minOccurs="0"/>
   </sequence>
</complexType>
```

2.7.4.4 rPr (Run Properties)

This element specifies the set of run properties which comprise the default run properties for the current WordprocessingML document. [*Rationale*: The reason that an rPr element is present within the rPrDefault element is to allow for easy repurposing of any set of run properties within a WordprocessingML document - since the run properties are always child elements of a single rPr element, that element can simply be relocated in its entirety to the desired new location without additional modifications. *end rationale*]

If this element is omitted, then the default run properties for the current document are non-existent (i.e. there are no default run properties, and the defaults are therefore application-defined).

[*Example*: Consider the following definition for the document defaults for a WordprocessingML document:

```
<w:docDefaults>
  …
  <w:rPrDefault>
    <w:rPr>
      <w:b/>
    </w::rPr>
  </w:rPrDefault>
</w:docDefaults>
```

The rPr element as a child of the rPrDefault element contains the set of default run properties for this document - in this case, bold text. *end example*]

Parent Elements
rPrDefault (§2.7.4.5)

Child Elements	Subclause
b (Bold)	§2.3.2.1
bCs (Complex Script Bold)	§2.3.2.2
bdr (Text Border)	§2.3.2.3
caps (Display All Characters As Capital Letters)	§2.3.2.4

Child Elements	Subclause
color (Run Content Color)	§2.3.2.5
cs (Use Complex Script Formatting on Run)	§2.3.2.6
dstrike (Double Strikethrough)	§2.3.2.7
eastAsianLayout (East Asian Typography Settings)	§2.3.2.8
effect (Animated Text Effect)	§2.3.2.9
em (Emphasis Mark)	§2.3.2.10
emboss (Embossing)	§2.3.2.11
fitText (Manual Run Width)	§2.3.2.12
highlight (Text Highlighting)	§2.3.2.13
i (Italics)	§2.3.2.14
iCs (Complex Script Italics)	§2.3.2.15
imprint (Imprinting)	§2.3.2.16
kern (Font Kerning)	§2.3.2.17
lang (Languages for Run Content)	§2.3.2.18
noProof (Do Not Check Spelling or Grammar)	§2.3.2.19
oMath (Office Open XML Math)	§2.3.2.20
outline (Display Character Outline)	§2.3.2.21
position (Vertically Raised or Lowered Text)	§2.3.2.22
rFonts (Run Fonts)	§2.3.2.24
rPrChange (Revision Information for Run Properties)	§2.13.5.32
rStyle (Referenced Character Style)	§2.3.2.27
rtl (Right To Left Text)	§2.3.2.28
shadow (Shadow)	§2.3.2.29
shd (Run Shading)	§2.3.2.30
smallCaps (Small Caps)	§2.3.2.31
snapToGrid (Use Document Grid Settings For Inter-Character Spacing)	§2.3.2.32
spacing (Character Spacing Adjustment)	§2.3.2.33
specVanish (Paragraph Mark Is Always Hidden)	§2.3.2.34
strike (Single Strikethrough)	§2.3.2.35
sz (Font Size)	§2.3.2.36
szCs (Complex Script Font Size)	§2.3.2.37
u (Underline)	§2.3.2.38
vanish (Hidden Text)	§2.3.2.39
vertAlign (Subscript/Superscript Text)	§2.3.2.40

Child Elements	Subclause
w (Expanded/Compressed Text)	§2.3.2.41
webHidden (Web Hidden Text)	§2.3.2.42

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RPr">
   <sequence>
      <group ref="EG_RPrContent" minOccurs="0"/>
   </sequence>
</complexType>
```

2.7.4.5 rPrDefault (Default Run Properties)

This element specifies the presence of a set of default run properties for the current document. The actual run properties are stored within the rPr child element of the current element.

If this element is omitted, then the default run properties for the current document are non-existent (i.e. there are no default run properties in the document, and the defaults are therefore application-defined).

[*Example*: Consider the following definition for the document defaults for a WordprocessingML document:

```
<w:docDefaults>
  …
  <w:rPrDefault>
    <w:rPr>
      <w:b/>
    </w::rPr>
  </w:rPrDefault>
</w:docDefaults>
```

The rPrDefault element is a container for the set of default run properties for this document. *end example*]

Parent Elements
docDefaults (§2.7.4.1)

Child Elements	Subclause
rPr (Run Properties)	§2.7.4.4

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RPrDefault">
   <sequence>
      <element name="rPr" type="CT_RPr" minOccurs="0"/>
   </sequence>
</complexType>
```

2.7.5 Table Styles

Table styles are style definitions which apply to the contents of zero or more tables within a document. This definition may imply that the style can only define table properties (properties which apply to the table and its constituent rows and cells), however a table style can also define paragraph properties (properties which apply to the positioning and appearance of paragraphs) as well as character properties (properties which apply to runs) for all of the paragraphs and runs within the specified table in the document.

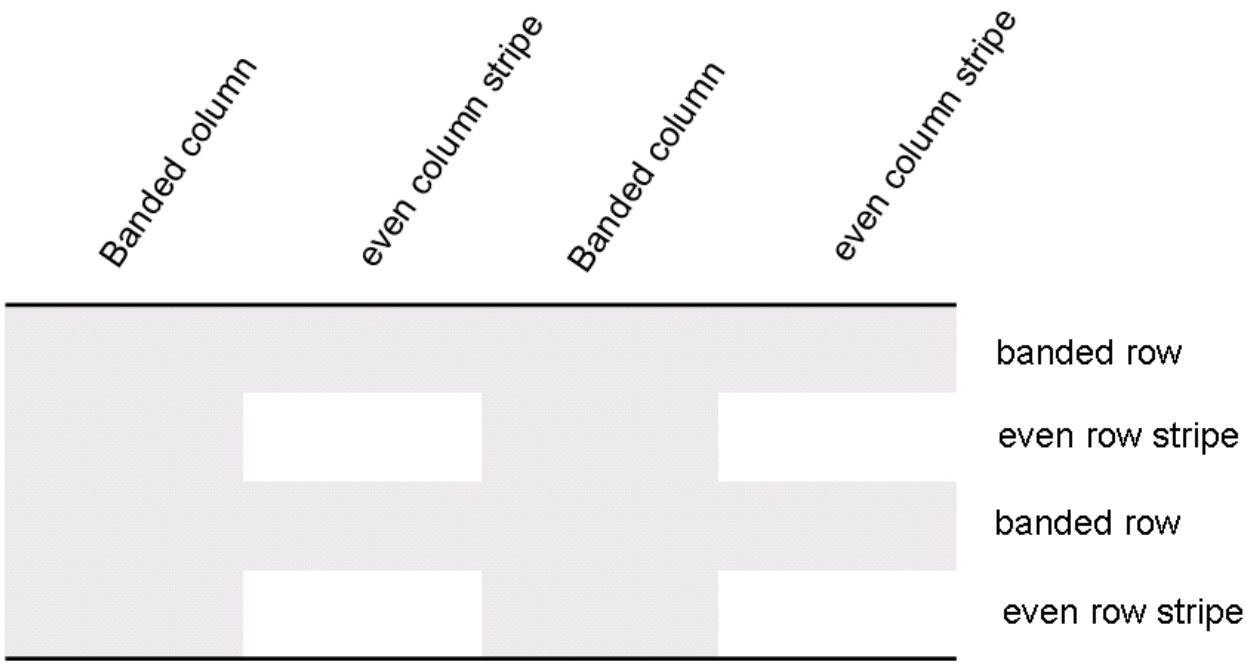
Table styles can only be referenced by tables within a document, and they must be referenced by the tblStyle element (§2.4.59) within a table's table properties.

As discussed above, table styles can specify all of the properties that can be applied to a table, as well as paragraph and character properties for the table's contents. However, unlike other style definitions, table styles allow for the definition of conditional formats for different regions of the table.

These table conditional formats are applied to different regions of the table as follows:

Top left cell	Header row	Top right cell
First column	Table body	Last column
Bottom left cell	Footer row	Bottom right cell

All rows in the table can also have conditional formatting on an alternating row/column basis as well as follows:



When specified, these conditional formats shall be applied in the following order (therefore subsequent formats override properties on previous formats):

- Whole table
- Banded columns, even column banding
- Banded rows, even row banding
- First row, last row
- First column, last column
- Top left, top right, bottom left, bottom right

[*Example*: Consider a table style `Test Table Style` defined as follows:

- All cells with 1pt table borders on all sides
- 0.1” cell margins on left and right of cells
- 0” cell margins on top and bottom of cells

As well as header row specific formatting of

- Red shading
- Bold text

```
<w:style w:type="table" w:styleId="TestTableStyle">
  <w:name w:val="Test Table Style"/>
  <w:basedOn w:val="TableNormal"/>
  <w:priority w:val="99"/>
  <w:rsid w:val="00340CC4"/>
  <w:tblPr>
    <w:tblBorders>
      <w:top w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:left w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:bottom w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:right w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:insideH w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:insideV w:val="single" w:sz="4" w:space="0" w:color="auto"/>
    </w:tblBorders>
    <w:tblCellMar>
      <w:top w:w="0" w:type="dxa"/>
      <w:left w:w="108" w:type="dxa"/>
      <w:bottom w:w="0" w:type="dxa"/>
      <w:right w:w="108" w:type="dxa"/>
    </w:tblCellMar>
  </w:tblPr>
  <w:tblStylePr w:type="firstRow">
    <w:rPr>
      <w:b/>
    </w:rPr>
    <w:tcPr>
      <w:shd w:val="clear" w:color="auto" w:fill="ED1C24"/>
    </w:tcPr>
  </w:tblStylePr>
</w:style>
```

The tblPr element holds the formatting which is applied to the entire table, and the tblStylePr element with a type attribute value of `firstRow` holds the formatting for the first table row, specifically the bold run property and red cell shading. *end example*]

An individual instance of a table defines an association with a table style using the tblStyle element in the table's properties (tblPr), as discussed above. However, individual tables can choose whether to apply the following aspects of the table's conditional formats individually:

- First row
- Last row
- First column
- Last column
- Row banding

- Column banding

The use or omission conditional formats shall be specified using the tblLook element, which contains a bitmask representing which properties are applied and omitted.

[*Example*: Consider two tables using the table style `Style2`; one which specifies that it should only use the header row and footer row conditional formatting properties from the table style, and the other which specifies that it should use the header row, footer row, and banded row conditional formatting:

```
<w:tbl>
  <w:tblPr>
    <w:tblStyle w:val="Style2"/>
    <w:tblW w:w="0" w:type="auto"/>
    <w:tblLook w:val="0660"/>
  </w:tblPr>
  …
</w:tbl>
…
<w:tbl>
  <w:tblPr>
    <w:tblStyle w:val="Style2"/>
    <w:tblW w:w="0" w:type="auto"/>
    <w:tblLook w:val="0460"/>
  </w:tblPr>
  …
</w:tbl>
```

The tables each specify the appropriate set of conditional formats using the tblLook element, as seen by the identical table styles in the tblStyle element, and different tblLook values. *end example*]

2.7.5.1 pPr (Table Style Conditional Formatting Paragraph Properties)

This element specifies the set of paragraph properties which shall be applied to all paragraphs within a table which match the conditional formatting type specified on the parent tblStylePr element. These properties are applied in the order specified via the style hierarchy.

[*Example*: Consider a table style which contains conditional formatting for its `firstRow`, defined as follows:

```
<w:style w:type="table" w:styleId="exampleTableStyle">
  …
  <w:tblStylePr w:type="firstRow">
    <w:pPr>
      <w:jc w:val="center"/>
    </w:pPr>
    …
  </w:tblStylePr>
</w:style>
```

The pPr element specified within the tblStylePr element specifies the set of paragraph properties which shall be applied to all parts of the table which meet the criteria specified by the type value of `firstRow` - all of the header rows of the table. In this example, the single paragraph property applied is an alignment value of center via the jc element (§2.3.1.13). *end example*]

Parent Elements
tblStylePr (§2.7.5.6)

Child Elements	Subclause
adjustRightInd (Automatically Adjust Right Indent When Using Document Grid)	§2.3.1.1
autoSpaceDE (Automatically Adjust Spacing of Latin and East Asian Text)	§2.3.1.2
autoSpaceDN (Automatically Adjust Spacing of East Asian Text and Numbers)	§2.3.1.3
bidi (Right to Left Paragraph Layout)	§2.3.1.6
cnfStyle (Paragraph Conditional Formatting)	§2.3.1.8
contextualSpacing (Ignore Spacing Above and Below When Using Identical Styles)	§2.3.1.9
divId (Associated HTML div ID)	§2.3.1.10
framePr (Text Frame Properties)	§2.3.1.11
ind (Paragraph Indentation)	§2.3.1.12
jc (Paragraph Alignment)	§2.3.1.13
keepLines (Keep All Lines On One Page)	§2.3.1.14
keepNext (Keep Paragraph With Next Paragraph)	§2.3.1.15
kinsoku (Use East Asian Typography Rules for First and Last Character per Line)	§2.3.1.16
mirrorIndents (Use Left/Right Indents as Inside/Outside Indents)	§2.3.1.18
numPr (Numbering Definition Instance Reference)	§2.3.1.19
outlineLvl (Associated Outline Level)	§2.3.1.20
overflowPunct (Allow Punctuation to Extent Past Text Extents)	§2.3.1.21
pageBreakBefore (Start Paragraph on Next Page)	§2.3.1.23
pBdr (Paragraph Borders)	§2.3.1.24

Child Elements	Subclause
pPrChange (Revision Information for Paragraph Properties)	§2.13.5.31
pStyle (Referenced Paragraph Style)	§2.3.1.27
rPr (Run Properties for the Paragraph Mark)	§2.3.1.29
sectPr (Section Properties)	§2.6.19
shd (Paragraph Shading)	§2.3.1.31
snapToGrid (Use Document Grid Settings for Inter-Line Paragraph Spacing)	§2.3.1.32
spacing (Spacing Between Lines and Above/Below Paragraph)	§2.3.1.33
suppressAutoHyphens (Suppress Hyphenation for Paragraph)	§2.3.1.34
suppressLineNumbers (Suppress Line Numbers for Paragraph)	§2.3.1.35
suppressOverlap (Prevent Text Frames From Overlapping)	§2.3.1.36
tabs (Set of Custom Tab Stops)	§2.3.1.38
textAlignment (Vertical Character Alignment on Line)	§2.3.1.39
textboxTightWrap (Allow Surrounding Paragraphs to Tight Wrap to Text Box Contents)	§2.3.1.40
textDirection (Paragraph Text Flow Direction)	§2.3.1.41
topLinePunct (Compress Punctuation at Start of a Line)	§2.3.1.43
widowControl (Allow First/Last Line to Display on a Separate Page)	§2.3.1.44
wordWrap (Allow Line Breaking At Character Level)	§2.3.1.45

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PPr">
   <complexContent>
      <extension base="CT_PPrBase">
         <sequence>
            <element name="rPr" type="CT_ParaRPr" minOccurs="0"/>
            <element name="sectPr" type="CT_SectPr" minOccurs="0"/>
            <element name="pPrChange" type="CT_PPrChange" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.7.5.2 rPr (Table Style Conditional Formatting Run Properties)

This element specifies the set of run properties which shall be applied to all runs within a table which match the conditional formatting type specified on the parent tblStylePr element. These properties are applied in the order specified via the style hierarchy.

[*Example*: Consider a table style which contains conditional formatting for its `firstRow`, defined as follows:

```
<w:style w:type="table" w:styleId="exampleTableStyle">
  …
  <w:tblStylePr w:type="firstRow">
    <w:rPr>
      <w:i/>
    </w:rPr>
    …
  </w:tblStylePr>
</w:style>
```

The rPr element specified within the tblStylePr element specifies the set of run properties which shall be applied to all parts of the table which meet the criteria specified by the type value of `firstRow` - all of the header rows of the table. In this example, the single run property applied is italics via the i element (§2.3.2.14). *end example*]

Parent Elements
tblStylePr (§2.7.5.6)

Child Elements	Subclause
b (Bold)	§2.3.2.1
bCs (Complex Script Bold)	§2.3.2.2
bdr (Text Border)	§2.3.2.3
caps (Display All Characters As Capital Letters)	§2.3.2.4
color (Run Content Color)	§2.3.2.5
cs (Use Complex Script Formatting on Run)	§2.3.2.6
dstrike (Double Strikethrough)	§2.3.2.7
eastAsianLayout (East Asian Typography Settings)	§2.3.2.8
effect (Animated Text Effect)	§2.3.2.9
em (Emphasis Mark)	§2.3.2.10
emboss (Embossing)	§2.3.2.11
fitText (Manual Run Width)	§2.3.2.12
highlight (Text Highlighting)	§2.3.2.13
i (Italics)	§2.3.2.14
iCs (Complex Script Italics)	§2.3.2.15
imprint (Imprinting)	§2.3.2.16
kern (Font Kerning)	§2.3.2.17
lang (Languages for Run Content)	§2.3.2.18
noProof (Do Not Check Spelling or Grammar)	§2.3.2.19

Child Elements	Subclause
oMath (Office Open XML Math)	§2.3.2.20
outline (Display Character Outline)	§2.3.2.21
position (Vertically Raised or Lowered Text)	§2.3.2.22
rFonts (Run Fonts)	§2.3.2.24
rPrChange (Revision Information for Run Properties)	§2.13.5.32
rStyle (Referenced Character Style)	§2.3.2.27
rtl (Right To Left Text)	§2.3.2.28
shadow (Shadow)	§2.3.2.29
shd (Run Shading)	§2.3.2.30
smallCaps (Small Caps)	§2.3.2.31
snapToGrid (Use Document Grid Settings For Inter-Character Spacing)	§2.3.2.32
spacing (Character Spacing Adjustment)	§2.3.2.33
specVanish (Paragraph Mark Is Always Hidden)	§2.3.2.34
strike (Single Strikethrough)	§2.3.2.35
sz (Font Size)	§2.3.2.36
szCs (Complex Script Font Size)	§2.3.2.37
u (Underline)	§2.3.2.38
vanish (Hidden Text)	§2.3.2.39
vertAlign (Subscript/Superscript Text)	§2.3.2.40
w (Expanded/Compressed Text)	§2.3.2.41
webHidden (Web Hidden Text)	§2.3.2.42

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RPr">
   <sequence>
      <group ref="EG_RPrContent" minOccurs="0"/>
   </sequence>
</complexType>
```

2.7.5.3 tblPr (Table Style Conditional Formatting Table Properties)

This element specifies the set of table properties which shall be applied to all regions within a table which match the conditional formatting type specified on the parent tblStylePr element. These properties are applied in the order specified via the style hierarchy.

If the current conditional formatting type does not consist of one or more full table rows, then table properties which cannot be applied to a single cell or column [*Example*: Table justification. *end example*] may be ignored.

[*Example*: Consider a table style which contains conditional formatting for its `firstRow`, defined as follows:

```
<w:style w:type="table" w:styleId="exampleTableStyle">
  …
  <w:tblStylePr w:type="firstRow">
    <w:tblPr>
      <w:tblCellSpacing w:w="29" w:type="dxa"/>
    </w:tblPr>
    …
  </w:tblStylePr>
</w:style>
```

The tblPr element specified within the tblStylePr element specifies the set of table properties which shall be applied to all parts of the table which meet the criteria specified by the type value of `firstRow` - all of the header rows of the table. In this example, the single table property applied is a default table cell spacing value of 0.02 inches via the tblCellSpacing element (§2.4.43). *end example*]

Parent Elements
tblStylePr (§2.7.5.6)

Child Elements	Subclause
bidiVisual (Visually Right to Left Table)	§2.4.1
jc (Table Alignment)	§2.4.23
shd (Table Shading)	§2.4.35
tblBorders (Table Borders)	§2.4.38
tblCellMar (Table Cell Margin Defaults)	§2.4.39
tblCellSpacing (Table Cell Spacing Default)	§2.4.43
tblInd (Table Indent from Leading Margin)	§2.4.48
tblLayout (Table Layout)	§2.4.49
tblLook (Table Style Conditional Formatting Settings)	§2.4.51
tblOverlap (Floating Table Allows Other Tables to Overlap)	§2.4.53
tblpPr (Floating Table Positioning)	§2.4.54
tblStyle (Referenced Table Style)	§2.4.59
tblStyleColBandSize (Number of Columns in Column Band)	§2.7.5.5
tblStyleRowBandSize (Number of Rows in Row Band)	§2.7.5.7
tblW (Preferred Table Width)	§2.4.61

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblPrBase">
   <sequence>
      <element name="tblStyle" type="CT_String" minOccurs="0"/>
      <element name="tblpPr" type="CT_TblPPr" minOccurs="0" maxOccurs="1"/>
      <element name="tblOverlap" type="CT_TblOverlap" minOccurs="0" maxOccurs="1"/>
      <element name="bidiVisual" type="CT_OnOff" minOccurs="0" maxOccurs="1"/>
      <element name="tblStyleRowBandSize" type="CT_DecimalNumber" minOccurs="0" maxOccurs="1"/>
      <element name="tblStyleColBandSize" type="CT_DecimalNumber" minOccurs="0" maxOccurs="1"/>
      <element name="tblW" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="jc" type="CT_Jc" minOccurs="0" maxOccurs="1"/>
      <element name="tblCellSpacing" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="tblInd" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="tblBorders" type="CT_TblBorders" minOccurs="0" maxOccurs="1"/>
      <element name="shd" type="CT_Shd" minOccurs="0" maxOccurs="1"/>
      <element name="tblLayout" type="CT_TblLayoutType" minOccurs="0" maxOccurs="1"/>
      <element name="tblCellMar" type="CT_TblCellMar" minOccurs="0" maxOccurs="1"/>
      <element name="tblLook" type="CT_ShortHexNumber" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

2.7.5.4 tblPr (Style Table Properties)

This element specifies the set of table properties which shall be applied to the table. These properties are not conditional and shall always be applied (although they are applied before all conditional formatting properties).

[*Example*: Consider a table style defined as follows:

```
<w:style w:type="table" w:styleId="exampleTableStyle">
  <w:tblPr>
    <w:tblCellSpacing w:w="15" w:type="dxa"/>
  </w:tblPr>
  …
  <w:tblStylePr w:type="firstRow">-
    <w:tblPr>
      <w:tblCellSpacing w:w="29" w:type="dxa"/>
    </w:tblPr>
    …
  </w:tblStylePr>
</w:style>
```

The tblPr element specified within the style element specifies the set of table properties which shall be applied to all parts of the table. In this example, the single table property applied is a default table cell spacing value of 0.01 inches via the tblCellSpacing element (§2.4.43). *end example*]

Parent Elements
style (§2.7.3.17)

Child Elements	Subclause
bidiVisual (Visually Right to Left Table)	§2.4.1
jc (Table Alignment)	§2.4.23
shd (Table Shading)	§2.4.35
tblBorders (Table Borders)	§2.4.38
tblCellMar (Table Cell Margin Defaults)	§2.4.39
tblCellSpacing (Table Cell Spacing Default)	§2.4.43
tblInd (Table Indent from Leading Margin)	§2.4.48
tblLayout (Table Layout)	§2.4.49
tblLook (Table Style Conditional Formatting Settings)	§2.4.51
tblOverlap (Floating Table Allows Other Tables to Overlap)	§2.4.53
tblpPr (Floating Table Positioning)	§2.4.54
tblStyle (Referenced Table Style)	§2.4.59
tblStyleColBandSize (Number of Columns in Column Band)	§2.7.5.5
tblStyleRowBandSize (Number of Rows in Row Band)	§2.7.5.7
tblW (Preferred Table Width)	§2.4.61

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblPrBase">
   <sequence>
      <element name="tblStyle" type="CT_String" minOccurs="0"/>
      <element name="tblpPr" type="CT_TblPPr" minOccurs="0" maxOccurs="1"/>
      <element name="tblOverlap" type="CT_TblOverlap" minOccurs="0" maxOccurs="1"/>
      <element name="bidiVisual" type="CT_OnOff" minOccurs="0" maxOccurs="1"/>
      <element name="tblStyleRowBandSize" type="CT_DecimalNumber" minOccurs="0" maxOccurs="1"/>
      <element name="tblStyleColBandSize" type="CT_DecimalNumber" minOccurs="0" maxOccurs="1"/>
      <element name="tblW" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="jc" type="CT_Jc" minOccurs="0" maxOccurs="1"/>
      <element name="tblCellSpacing" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="tblInd" type="CT_TblWidth" minOccurs="0" maxOccurs="1"/>
      <element name="tblBorders" type="CT_TblBorders" minOccurs="0" maxOccurs="1"/>
      <element name="shd" type="CT_Shd" minOccurs="0" maxOccurs="1"/>
      <element name="tblLayout" type="CT_TblLayoutType" minOccurs="0" maxOccurs="1"/>
      <element name="tblCellMar" type="CT_TblCellMar" minOccurs="0" maxOccurs="1"/>
      <element name="tblLook" type="CT_ShortHexNumber" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

2.7.5.5 tblStyleColBandSize (Number of Columns in Column Band)

This element specifies the number of columns which shall comprise each a table style column band for this table style. This element determines how many columns constitute each of the column bands for the current table, allowing column band formatting to be applied to groups of columns (rather than just single alternating columns) when the table is formatted.

If this element is omitted, then the default number of columns in a single column band shall be assumed to be 1.

[*Example*: Consider a table style defined as follows:

```
<w:style w:type="table" w:styleId="exampleTableStyle">
  <w:tblPr>
    <w:tblStyleRowBandSize w:val="3" />
    <w:tblStyleColBandSize w:val="2" />
  </w:tblPr>
  …
</w:style>
```

The tblStyleColBandSize element specifies that the width of each column band shall be 2 columns - therefore band1Vert column banding conditional formatting shall be applied to columns 1 and 2, 5 and 6, etc. in the table. *end example*]

Parent Elements
tblPr (§2.7.5.3); tblPr (§2.7.5.4); tblPr (§2.4.55); tblPr (§2.4.56)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

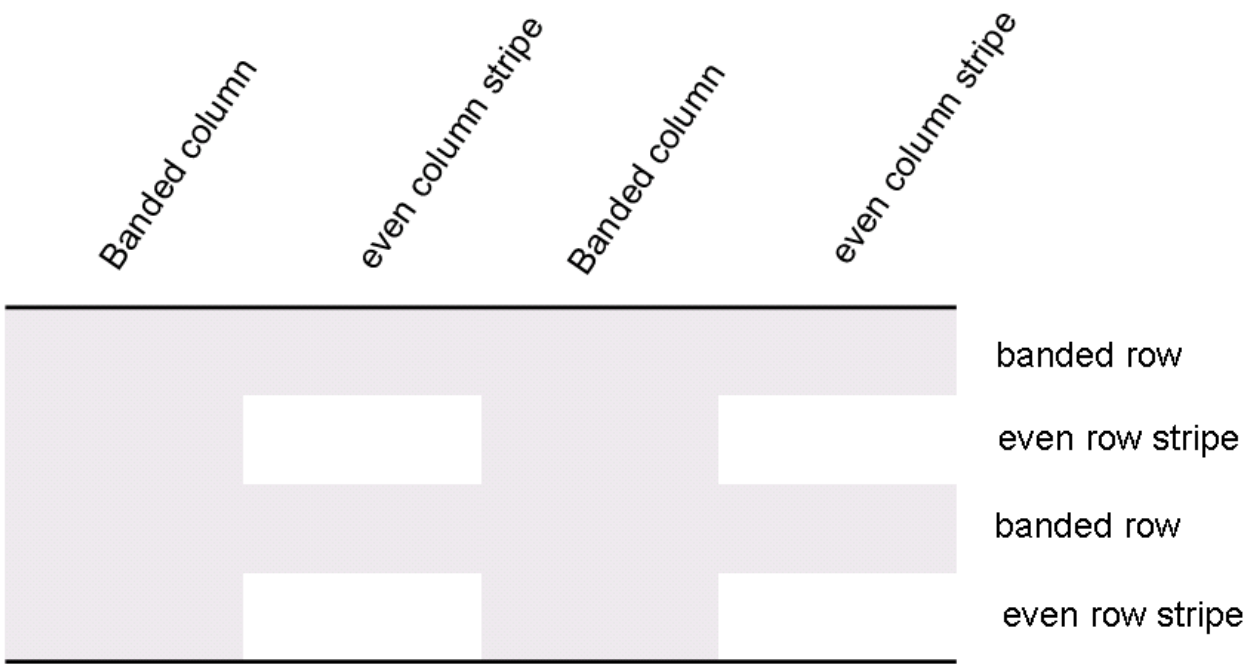
```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.7.5.6 tblStylePr (Style Conditional Table Formatting Properties)

This element specifies a set of formatting properties which shall be conditionally applied to the parts of a table which match the requirement specified on the type attribute. These table conditional formats are applied to different regions of the table as follows:

Top left cell	Header row	Top right cell
First column	Table body	Last column
Bottom left cell	Footer row	Bottom right cell

All rows in the table can also have conditional formatting on an alternating row/column basis as well as follows:



When specified, these conditional formats shall be applied in the following order (therefore subsequent formats override properties on previous formats):

- Whole table
- Banded columns, even column banding
- Banded rows, even row banding
- First row, last row
- First column, last column

- Top left, top right, bottom left, bottom right

[*Example*: Consider a table style which contains conditional formatting, defined as follows:

```
<w:style w:type="table" w:styleId="exampleTableStyle">
  …
  <w:tblStylePr w:type="firstRow">
    <w:tblPr>
      <w:tblCellSpacing w:w="29" w:type="dxa"/>
    </w:tblPr>
    …
  </w:tblStylePr>
</w:style>
```

The tblStylePr element specifies a set of table properties which shall be conditionally applied to all parts of the table which meet the criteria specified by the type attribute (in this case, all heading rows for the current table). *end example*]

Parent Elements
style (§2.7.3.17)

Child Elements	Subclause
pPr (Table Style Conditional Formatting Paragraph Properties)	§2.7.5.1
rPr (Table Style Conditional Formatting Run Properties)	§2.7.5.2
tblPr (Table Style Conditional Formatting Table Properties)	§2.7.5.3
tcPr (Table Style Conditional Formatting Table Cell Properties)	§2.7.5.9
trPr (Table Style Conditional Formatting Table Row Properties)	§2.7.5.10

Attributes	Description

type (Table Style Conditional Formatting Type)	Specifies the section of the table to which the current conditional formatting properties shall be applied.
	
	[*Example*: Consider a table style which contains conditional formatting, defined as follows:
	
	`<w:style w:type="table" …>`
	`  …`
	`  <w:tblStylePr w:type="lastRow">`
	`  …`
	`  </w:tblStylePr>`
	`</w:style>`

Attributes	Description
	The type attribute value of `lastRow` specifies that this set of conditional formatting properties shall be applied to the last row of the table only. *end example*] The possible values for this attribute are defined by the ST_TblStyleOverrideType simple type (§2.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblStylePr">
   <sequence>
      <element name="pPr" type="CT_PPr" minOccurs="0"/>
      <element name="rPr" type="CT_RPr" minOccurs="0"/>
      <element name="tblPr" type="CT_TblPrBase" minOccurs="0"/>
      <element name="trPr" type="CT_TrPr" minOccurs="0" maxOccurs="1"/>
      <element name="tcPr" type="CT_TcPr" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="type" type="ST_TblStyleOverrideType" use="required"/>
</complexType>
```

2.7.5.7 tblStyleRowBandSize (Number of Rows in Row Band)

This element specifies the number of rows which shall comprise each a table style row band for this table style. This element determines how many rows constitute each of the row bands for the current table, allowing row band formatting to be applied to groups of rows (rather than just single alternating rows) when the table is formatted.

If this element is omitted, then the default number of rows in a single row band shall be assumed to be `1`.

[*Example*: Consider a table style defined as follows:

```
<w:style w:type="table" w:styleId="exampleTableStyle">
  <w:tblPr>
    <w:tblStyleRowBandSize w:val="3" />
    <w:tblStyleColBandSize w:val="2" />
  </w:tblPr>
  …
</w:style>
```

The tblStyleRowBandSize element specifies that the width of each row band shall be `3` columns - therefore `band1Horiz` row banding conditional formatting shall be applied to row 1 through 3, 7 through 9, etc. in the table. *end example*]

Parent Elements
tblPr (§2.7.5.3); tblPr (§2.7.5.4); tblPr (§2.4.55); tblPr (§2.4.56)

Attributes	Description

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.7.5.8 tcPr (Style Table Cell Properties)

This element specifies the set of table cell properties which shall be applied to the table. These properties are not conditional and shall always be applied (although they are applied before all conditional formatting properties).

[*Example*: Consider a table style defined as follows:

```
<w:style w:type="table" w:styleId="exampleTableStyle">
  <w:tcPr>
    <w:tcFitText/>
  </w:tcPr>
</w:style>
```

The tcPr element specified within the style element specifies the set of table cell properties which shall be applied to all parts of the table. In this example, the single table cell property applied is the fit text setting via the tcFitText element (§2.4.64). *end example*]

Parent Elements
style (§2.7.3.17)

Child Elements	Subclause
cellDel (Table Cell Deletion)	§2.13.5.1

Child Elements	Subclause
cellIns (Table Cell Insertion)	§2.13.5.2
cellMerge (Vertically Merged/Split Table Cells)	§2.13.5.3
cnfStyle (Table Cell Conditional Formatting)	§2.4.7
gridSpan (Grid Columns Spanned by Current Table Cell)	§2.4.13
hideMark (Ignore End Of Cell Marker In Row Height Calculation)	§2.4.15
hMerge (Horizontally Merged Cell)	§2.4.16
noWrap (Don't Wrap Cell Content)	§2.4.28
shd (Table Cell Shading)	§2.4.33
tcBorders (Table Cell Borders)	§2.4.63
tcFitText (Fit Text Within Cell)	§2.4.64
tcMar (Single Table Cell Margins)	§2.4.65
tcPrChange (Revision Information for Table Cell Properties)	§2.13.5.38
tcW (Preferred Table Cell Width)	§2.4.68
textDirection (Table Cell Text Flow Direction)	§2.4.69
vAlign (Table Cell Vertical Alignment)	§2.4.80
vMerge (Vertically Merged Cell)	§2.4.81

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TcPr">
   <complexContent>
      <extension base="CT_TcPrInner">
         <sequence>
            <element name="tcPrChange" type="CT_TcPrChange" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.7.5.9 tcPr (Table Style Conditional Formatting Table Cell Properties)

This element specifies the set of table cell properties which shall be applied to all regions within a table which match the conditional formatting type specified on the parent tblStylePr element. These properties are applied in the order specified via the style hierarchy.

[*Example*: Consider a table style which contains conditional formatting for its `firstRow`, defined as follows:

```
<w:style w:type="table" w:styleId="exampleTableStyle">
  …
  <w:tblStylePr w:type="firstRow">
    <w:tcPr>
      <w:tcBorders>
        <w:top w:val="nil" />
        <w:left w:val="nil" />
        <w:bottom w:val="nil" />
        <w:right w:val="nil" />
        <w:insideH w:val="nil" />
        <w:insideV w:val="nil" />
      </w:tcBorders>
    </w:tcPr>
    …
  </w:tblStylePr>
</w:style>
```

The tcPr element specified within the tblStylePr element specifies the set of table cell properties which shall be applied to all parts of the table which meet the criteria specified by the type value of `firstRow` - all of the header rows of the table. In this example, the single table cell property applied is a set of table cell borders via the tcBorders element (§2.4.63). In this case, these cell borders simply reset any previous cell borders to `nil`. *end example*]

Parent Elements
tblStylePr (§2.7.5.6)

Child Elements	Subclause
cellDel (Table Cell Deletion)	§2.13.5.1
cellIns (Table Cell Insertion)	§2.13.5.2
cellMerge (Vertically Merged/Split Table Cells)	§2.13.5.3
cnfStyle (Table Cell Conditional Formatting)	§2.4.7
gridSpan (Grid Columns Spanned by Current Table Cell)	§2.4.13
hideMark (Ignore End Of Cell Marker In Row Height Calculation)	§2.4.15
hMerge (Horizontally Merged Cell)	§2.4.16
noWrap (Don't Wrap Cell Content)	§2.4.28
shd (Table Cell Shading)	§2.4.33
tcBorders (Table Cell Borders)	§2.4.63
tcFitText (Fit Text Within Cell)	§2.4.64
tcMar (Single Table Cell Margins)	§2.4.65

Child Elements	Subclause
tcPrChange (Revision Information for Table Cell Properties)	§2.13.5.38
tcW (Preferred Table Cell Width)	§2.4.68
textDirection (Table Cell Text Flow Direction)	§2.4.69
vAlign (Table Cell Vertical Alignment)	§2.4.80
vMerge (Vertically Merged Cell)	§2.4.81

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TcPr">
   <complexContent>
      <extension base="CT_TcPrInner">
         <sequence>
            <element name="tcPrChange" type="CT_TcPrChange" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.7.5.10 trPr (Table Style Conditional Formatting Table Row Properties)

This element specifies the set of table row properties which shall be applied to all rows within a table which match the conditional formatting type specified on the parent tblStylePr element. These properties are applied in the order specified via the style hierarchy.

[*Example*: Consider a table style which contains conditional formatting for its `firstRow`, defined as follows:

```
<w:style w:type="table" w:styleId="exampleTableStyle">
  …
  <w:tblStylePr w:type="firstRow">
    <w:trPr>
      <w:tblHeader/>
      <w:cantSplit/>
    </w:trPr>
    …
  </w:tblStylePr>
</w:style>
```

The trPr element specified within the tblStylePr element specifies the set of table row properties which shall be applied to all rows of the table which meet the criteria specified by the type value of `firstRow` - all of the header rows of the table. In this example, the table row properties applied are the fact that these rows shall be repeated on each page via the tblHeader element (§2.4.46) and the fact that these rows shall not be split across pages using the cantSplit element (§2.4.6). *end example*]

Parent Elements
tblStylePr (§2.7.5.6)

Child Elements	Subclause
cantSplit (Table Row Cannot Break Across Pages)	§2.4.6
cnfStyle (Table Row Conditional Formatting)	§2.4.8
del (Deleted Table Row)	§2.13.5.14
divId (Associated HTML div ID)	§2.4.9
gridAfter (Grid Columns After Last Cell)	§2.4.10
gridBefore (Grid Columns Before First Cell)	§2.4.11
hidden (Hidden Table Row Marker)	§2.4.14
ins (Inserted Table Row)	§2.13.5.16
jc (Table Row Alignment)	§2.4.22
tblCellSpacing (Table Row Cell Spacing)	§2.4.42
tblHeader (Repeat Table Row on Every New Page)	§2.4.46
trHeight (Table Row Height)	§2.4.77
trPrChange (Revision Information for Table Row Properties)	§2.13.5.39
wAfter (Preferred Width After Table Row)	§2.4.82
wBefore (Preferred Width Before Table Row)	§2.4.83

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TrPr">
   <complexContent>
      <extension base="CT_TrPrBase">
         <sequence>
            <element name="ins" type="CT_TrackChange" minOccurs="0"/>
            <element name="del" type="CT_TrackChange" minOccurs="0"/>
            <element name="trPrChange" type="CT_TrPrChange" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.7.5.11 trPr (Style Table Row Properties)

This element specifies the set of table row properties which shall be applied to the table. These properties are not conditional and shall always be applied (although they are applied before all conditional formatting properties).

[*Example*: Consider a table style defined as follows:

```
<w:style w:type="table" w:styleId="exampleTableStyle">
  <w:trPr>
    <w:jc w:val="center"/>
  </w:trPr>
</w:style>
```

The trPr element specified within the style element specifies the set of table row properties which shall be applied to all parts of the table. In this example, the single table row property applied is the alignment setting of center via the jc element (§2.4.22). *end example*]

Parent Elements
style (§2.7.3.17)

Child Elements	Subclause
cantSplit (Table Row Cannot Break Across Pages)	§2.4.6
cnfStyle (Table Row Conditional Formatting)	§2.4.8
del (Deleted Table Row)	§2.13.5.14
divId (Associated HTML div ID)	§2.4.9
gridAfter (Grid Columns After Last Cell)	§2.4.10
gridBefore (Grid Columns Before First Cell)	§2.4.11
hidden (Hidden Table Row Marker)	§2.4.14
ins (Inserted Table Row)	§2.13.5.16
jc (Table Row Alignment)	§2.4.22
tblCellSpacing (Table Row Cell Spacing)	§2.4.42
tblHeader (Repeat Table Row on Every New Page)	§2.4.46
trHeight (Table Row Height)	§2.4.77
trPrChange (Revision Information for Table Row Properties)	§2.13.5.39
wAfter (Preferred Width After Table Row)	§2.4.82
wBefore (Preferred Width Before Table Row)	§2.4.83

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TrPr">
   <complexContent>
      <extension base="CT_TrPrBase">
         <sequence>
            <element name="ins" type="CT_TrackChange" minOccurs="0"/>
            <element name="del" type="CT_TrackChange" minOccurs="0"/>
            <element name="trPrChange" type="CT_TrPrChange" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.7.6 Numbering Styles

Numbering styles are style definitions which specify common style properties for a multi-level numbering format within a document. This means that a numbering style defines only a single paragraph property: a reference to a numbering definition stored in the document's numbering part, using the numPr element.

Unlike paragraph and character styles, numbering styles are never directly referenced by content in the document – instead, an abstract numbering definition (covered in the numbering section) specifies that it is actually the underlying numbering information for a numbering style.

[*Example*: Consider a numbering style "Test Numbering Style":

```
<w:style w:type="numbering" w:styleId="TestNumberingStyle">
  <w:name w:val="Test Numbering Style" />
  <w:priority w:val="99" />
  <w:rsid w:val="0045009F" />
  <w:pPr>
    <w:numPr>
      <w:numId w:val="1" />
    </w:numPr>
  </w:pPr>
</w:style>
```

The only information specified in the numbering style definition is a reference to the numbering definition for the numbering information which is defined by this numbering style. *end example*]

2.7.7 Paragraph Styles

Paragraph styles are styles which apply to the contents of an entire paragraph as well as the paragraph mark. This definition implies that the style can define both character properties (properties which apply to text within the document) as well as paragraph properties (properties which apply to the positioning and appearance of the paragraph). Paragraph styles cannot be referenced by runs within a document; they must be referenced by the pStyle element (§2.3.1.27) within a paragraph's paragraph properties element.

A paragraph style has three defining type-specific characteristics:

- The type attribute on the style has a value of paragraph, which indicates that the following style definition is a paragraph style.
- The next element defines an editing behavior which supplies the paragraph style to be automatically applied to the next paragraph when ENTER is pressed at the end of a paragraph of this style.
- The style specifies both paragraph-level and character-level properties using the pPr and rPr elements, respectively. In this case, the run properties are the set of properties applied to each run in the paragraph.

The paragraph style is then applied to paragraphs by referencing the styleId attribute value for this style in the paragraph properties' pStyle element.

[*Example:* Consider a paragraph style titled "Test Paragraph Style" which defines; font = Algerian, font size = 20; font color = red; paragraph spacing = double; paragraph indent = 1" (first line only). The resulting style definition would be:

```
<w:style w:type="paragraph" w:styleId="TestParagraphStyle">
  <w:name w:val="Test Paragraph Style"/>
  <w:qformat/>
  <w:rsid w:val="00F85845"/>
  <w:pPr>
    <w:spacing w:line="480" w:lineRule="auto"/>
    <w:ind w:firstLine="1440"/>
  </w:pPr>
  <w:rPr>
    <w:rFonts w:ascii="Algerian" w:hAnsi="Algerian"/>
    <w:color w:val="ED1C24"/>
    <w:sz w:val="40"/>
  </w:rPr>
</w:style>
```

Notice that the character properties for the style are under the rPr element, and the paragraph properties are under the pPr element.

The document content for a paragraph of this style would be:

```
<w:p>
  <w:pPr>
    <w:pStyle w:val="TestParagraphStyle"/>
  </w:pPr>
  <w:r>
    <w:t xml:space="preserve">Here is some fancy Text</w:t>
  </w:r>
</w:p>
```

The pStyle element links the paragraph with the style definition. *end example*]

2.7.7.1 Numbering in Paragraph Styles

When a paragraph style references a numbering definition and level which shall also be applied, that reference shall be done in a way slightly different from the typical numbering reference as follows:

- When a numbering reference is created as direct formatting, that reference consists of a reference to the numbering definition instance + a numbering level

[*Example*: Consider a numbered paragraph in a WordprocessingML document whose numbering is a result of direct formatting (formatting not from a style). This numbered paragraph may be represented using the following WordprocessingML:

```
<w:p>
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="0" />
      <w:numId w:val="5" />
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>Level one</w:t>
  </w:r>
</w:p>
```

The numPr element contains two pieces of information:

- The numId element (the numbering definition instance referenced)
- The ilvl element (the level within that numbering definition)

end example]

- When numbering is done as part of a paragraph style, that reference consists of a reference to the numbering definition only. The numbering definition then in turn has a reference to the paragraph style on the level which shall be associated with this style

[*Example*: Consider a numbered paragraph in a WordprocessingML document whose numbering is a result of a paragraph style. This numbered paragraph may be represented using the following WordprocessingML:

```
<w:p>
  <w:pPr>
    <w:pStyle w:val="TestParagraphStyle"/>
  </w:pPr>
  <w:r>
    <w:t>Level one</w:t>
  </w:r>
</w:p>
```

The paragraph references the style via its styleId attribute, which itself looks like this:

```
<w:style w:styleId="TestParagraphStyle" … >
  <w:pPr>
    <w:numPr>
      <w:numId w:val="5" />
    </w:numPr>
  </w:pPr>
</w:style>
```

The numPr element contains one piece of information:

- The numId element (the numbering definition instance referenced)

Obviously, this is insufficient to apply the numbering since we need to know which level to apply, so this information is specified on the appropriate level using the pStyle element:

```
<w:abstractNum w:abstractNumId="1">
  ...
  <w:lvl w:ilvl="0">
    ...
    <w:pStyle w:val="TestParagraphStyle" />
    <w:pPr>
      <w:tabs>
        <w:tab w:val="num" w:pos="720" />
      </w:tabs>
      <w:ind w:left="720" w:hanging="360" />
    </w:pPr>
    ...
  </w:lvl>
</w:abstractNum>
```

In this case, level `0` of the underlying abstract numbering definition specifies that it is associated with paragraph style `TestParagraphStyle`, so this level of the numbering shall be applied along with the paragraph style. *end example*]

When numbering is referenced by a paragraph style, its properties shall be applied before the style's properties (the style's paragraph properties shall override the numbering level's paragraph properties).

2.7.7.2 pPr (Style Paragraph Properties)

This element specifies the set of paragraph properties which shall be applied to the paragraph.

[*Example*: Consider a paragraph style defined as follows:

```
<w:style w:type="paragraph" w:styleId="TestParaStyle">
  <w:pPr>
    <w:keepLines/>
  </w:pPr>
</w:style>
```

The pPr element specified within the style element specifies the set of paragraph properties which shall be applied to the referencing paragraph. In this example, the single paragraph property applied is the fact that the paragraph shall be displayed on a single page via the keepLines element (§2.3.1.14). *end example*]

Parent Elements
style (§2.7.3.17)

Child Elements	Subclause
adjustRightInd (Automatically Adjust Right Indent When Using Document Grid)	§2.3.1.1
autoSpaceDE (Automatically Adjust Spacing of Latin and East Asian Text)	§2.3.1.2
autoSpaceDN (Automatically Adjust Spacing of East Asian Text and Numbers)	§2.3.1.3
bidi (Right to Left Paragraph Layout)	§2.3.1.6
cnfStyle (Paragraph Conditional Formatting)	§2.3.1.8
contextualSpacing (Ignore Spacing Above and Below When Using Identical Styles)	§2.3.1.9
divId (Associated HTML div ID)	§2.3.1.10
framePr (Text Frame Properties)	§2.3.1.11
ind (Paragraph Indentation)	§2.3.1.12
jc (Paragraph Alignment)	§2.3.1.13
keepLines (Keep All Lines On One Page)	§2.3.1.14
keepNext (Keep Paragraph With Next Paragraph)	§2.3.1.15
kinsoku (Use East Asian Typography Rules for First and Last Character per Line)	§2.3.1.16
mirrorIndents (Use Left/Right Indents as Inside/Outside Indents)	§2.3.1.18
numPr (Numbering Definition Instance Reference)	§2.3.1.19
outlineLvl (Associated Outline Level)	§2.3.1.20
overflowPunct (Allow Punctuation to Extent Past Text Extents)	§2.3.1.21
pageBreakBefore (Start Paragraph on Next Page)	§2.3.1.23
pBdr (Paragraph Borders)	§2.3.1.24
pPrChange (Revision Information for Paragraph Properties)	§2.13.5.31
pStyle (Referenced Paragraph Style)	§2.3.1.27
rPr (Run Properties for the Paragraph Mark)	§2.3.1.29
sectPr (Section Properties)	§2.6.19

Child Elements	Subclause
shd (Paragraph Shading)	§2.3.1.31
snapToGrid (Use Document Grid Settings for Inter-Line Paragraph Spacing)	§2.3.1.32
spacing (Spacing Between Lines and Above/Below Paragraph)	§2.3.1.33
suppressAutoHyphens (Suppress Hyphenation for Paragraph)	§2.3.1.34
suppressLineNumbers (Suppress Line Numbers for Paragraph)	§2.3.1.35
suppressOverlap (Prevent Text Frames From Overlapping)	§2.3.1.36
tabs (Set of Custom Tab Stops)	§2.3.1.38
textAlignment (Vertical Character Alignment on Line)	§2.3.1.39
textboxTightWrap (Allow Surrounding Paragraphs to Tight Wrap to Text Box Contents)	§2.3.1.40
textDirection (Paragraph Text Flow Direction)	§2.3.1.41
topLinePunct (Compress Punctuation at Start of a Line)	§2.3.1.43
widowControl (Allow First/Last Line to Display on a Separate Page)	§2.3.1.44
wordWrap (Allow Line Breaking At Character Level)	§2.3.1.45

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PPr">
   <complexContent>
      <extension base="CT_PPrBase">
         <sequence>
            <element name="rPr" type="CT_ParaRPr" minOccurs="0"/>
            <element name="sectPr" type="CT_SectPr" minOccurs="0"/>
            <element name="pPrChange" type="CT_PPrChange" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.7.8 Run (Character) Styles

Character styles are styles which apply to the contents of one or more runs of text within a document's contents. This definition implies that the style can only define character properties (properties which apply to text within a paragraph) because it cannot be applied to paragraphs. Character styles can only be referenced by runs within a document, and they must be referenced by the rStyle element within a run's run properties element.

A character style has two defining type-specific characteristics:

- The type attribute on the style has a value of `character`, which indicates that the following style definition is a character style.
- The style specifies only character-level properties using the rPr element. In this case, the run properties are the set of properties applied to each run which is of this style.

The character style is then applied to runs by referencing the styleId attribute value for this style in the run properties' rStyle element.

[*Example:* Consider a character style titled "Test Character Style" which defines; font = Courier New, font color = yellow; underline. The resulting style definition would be:

```
<w:style w:type="character" w:styleId="TestCharacterStyle">
  <w:name w:val="Test Character Style"/>
  <w:priority w:val="99"/>
  <w:qformat/>
  <w:rsid w:val="00E77BF0"/>
  <w:rPr>
    <w:rFonts w:ascii="Courier New" w:hAnsi="Courier New"/>
    <w:color w:val="FFF200"/>
    <w:u w:val="single"/>
  </w:rPr>
</w:style>
```

Notice that the character properties applied using this style are under the rPr element. The document content for a paragraph with a run of this style would be:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">The following text is in the </w:t>
  </w:r>
  <w:r>
    <w:rPr>
      <w:rStyle w:val="TestCharacterStyle"/>
    </w:rPr>
    <w:t>character style</w:t>
  </w:r>
  <w:r>
    <w:t>.</w:t>
  </w:r>
</w:p>
```

The rStyle element in the second run links that run with the style definition, inheriting the formatting properties for that run. *end example*]

2.7.8.1 rPr (Run Properties)

This element specifies the set of run properties which shall be applied to the run.

[*Example*: Consider a character style defined as follows:

```
<w:style w:type="character" w:styleId="TestCharStyle">
  <w:rPr>
    <w:dstrike/>
  </w:rPr>
</w:style>
```

The rPr element specified within the style element specifies the set of run properties which shall be applied to the referencing run. In this example, the single run property applied is the fact that the paragraph shall be displayed with double strikethrough via the dstrike element (§2.3.2.7). *end example*]

Parent Elements
style (§2.7.3.17)

Child Elements	Subclause
b (Bold)	§2.3.2.1
bCs (Complex Script Bold)	§2.3.2.2
bdr (Text Border)	§2.3.2.3
caps (Display All Characters As Capital Letters)	§2.3.2.4
color (Run Content Color)	§2.3.2.5
cs (Use Complex Script Formatting on Run)	§2.3.2.6
dstrike (Double Strikethrough)	§2.3.2.7
eastAsianLayout (East Asian Typography Settings)	§2.3.2.8
effect (Animated Text Effect)	§2.3.2.9
em (Emphasis Mark)	§2.3.2.10
emboss (Embossing)	§2.3.2.11
fitText (Manual Run Width)	§2.3.2.12
highlight (Text Highlighting)	§2.3.2.13
i (Italics)	§2.3.2.14
iCs (Complex Script Italics)	§2.3.2.15
imprint (Imprinting)	§2.3.2.16
kern (Font Kerning)	§2.3.2.17
lang (Languages for Run Content)	§2.3.2.18
noProof (Do Not Check Spelling or Grammar)	§2.3.2.19
oMath (Office Open XML Math)	§2.3.2.20
outline (Display Character Outline)	§2.3.2.21
position (Vertically Raised or Lowered Text)	§2.3.2.22
rFonts (Run Fonts)	§2.3.2.24

Child Elements	Subclause
rPrChange (Revision Information for Run Properties)	§2.13.5.32
rStyle (Referenced Character Style)	§2.3.2.27
rtl (Right To Left Text)	§2.3.2.28
shadow (Shadow)	§2.3.2.29
shd (Run Shading)	§2.3.2.30
smallCaps (Small Caps)	§2.3.2.31
snapToGrid (Use Document Grid Settings For Inter-Character Spacing)	§2.3.2.32
spacing (Character Spacing Adjustment)	§2.3.2.33
specVanish (Paragraph Mark Is Always Hidden)	§2.3.2.34
strike (Single Strikethrough)	§2.3.2.35
sz (Font Size)	§2.3.2.36
szCs (Complex Script Font Size)	§2.3.2.37
u (Underline)	§2.3.2.38
vanish (Hidden Text)	§2.3.2.39
vertAlign (Subscript/Superscript Text)	§2.3.2.40
w (Expanded/Compressed Text)	§2.3.2.41
webHidden (Web Hidden Text)	§2.3.2.42

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RPr">
   <sequence>
      <group ref="EG_RPrContent" minOccurs="0"/>
   </sequence>
</complexType>
```

2.8 Fonts

The next component of a WordprocessingML document is storing information about the fonts used in the document. WordprocessingML stores two pieces of information about fonts:

- (optionally) Information about the font to enable font substitution. *Font substitution* is a process by which an application, when it cannot locate a specific font, attempts to locate the closest possible match to the intended appearance of the font
- (optionally) One or more embedded forms of the font for use on systems which do not have access to the font. When fonts are embedded, they are obfuscated to ensure that they are only used to view the contents of the document in which they are embedded, and for no other purpose.

[*Example*: Consider the following information stored in a document's font table part:

```
<w:fonts>
  <w:font w:name="Times New Roman">
    <w:panose1 w:val="02020603050405020304" />
    <w:charset w:val="00" />
    <w:family w:val="roman" />
    <w:pitch w:val="variable" />
    <w:sig w:usb0="20002A87" w:usb1="80000000" w:usb2="00000008"
w:usb3="00000000" w:csb0="000001FF" w:csb1="00000000" />
    <w:embedRegular r:id="rId10" w:fontKey="{302EE813-EB4A-4642-A93A-
89EF99B2457E}" />
  </w:font>
</w:fonts>
```

The font table contains information about the `Times New Roman` font; specifically, information used to locate a substitute font when it is not available and a relationship to the embedded form of the regular form of the font. *end example*]

2.8.1 Font Embedding

Within a WordprocessingML document, *font embedding* refers to a process in which the some or all of the fonts used in the current document are included in that document such that it can be guaranteed that they are available for use when the document is subsequently opened.

Embedded fonts are stored in an Embedded Font part within the package.

When a font is embedded within a WordprocessingML document, it shall be obfuscated to prevent it from being used outside of this document. This obfuscation shall be done using the following algorithm:

- Generate a GUID, which will be used and stored as the obfuscation key
- Reverse the order of the bytes in the GUID (i.e. Big Endian ordering)
- XOR the value with the first 32 bytes of the binary: once against 0-15, once against 16-31
- Store the resulting file in the document, and store the obfuscation key in the fontKey attribute

[*Example*: Consider a font to be embedded whose first 32 bytes are as follows:

00	01	00	00	00	12	01	00	00	04	00	20	44	53	49	47
A3	0E	BF	F3	00	01	36	8C	00	00	14	DC	4C	54	53	48

To obfuscate this font for storage:

- Generate a GUID (e.g. `001B70DC-AA60-4AD5-90EC-18A0948E1EAE`)
- Reverse its order (e.g. `AE1E8E94-A018-EC90-D54A-60AADC701B00`)
- XOR the GUID with the first and second 16 bytes

The resulting 32 bytes would be:

AE	1F	8E	94	A0	0A	ED	90	D5	4E	60	8A	98	23	52	47
0D	10	31	67	A0	19	DA	1C	D5	4A	74	76	90	24	48	48

end example]

To retrieve an obfuscated font for viewing the content of this document only, repeat the procedure above to retrieve the original font.

2.8.2 Elements

The following elements comprise the content of the font table:

2.8.2.1 altName (Alternate Names for Font)

This element specifies a set of alternative names which may be used to locate the font specified by the parent element. This set of alternative names is stored in a comma-delimited list, with all adjacent commas ignored (i.e. a value of `Name A, Name B` is equivalent to `Name A,,,,,,,,, Name B`).

When an application cannot locate a font using the primary name stored on the font attribute of the font element (§2.8.2.10), it should use each alternate name in term to attempt to locate the font, and use the first font for which is locates a match.

If this element is omitted, then no alternate names are present for the parent font.

[*Example*: Consider the following information stored for a single font:

```
<w:font w:name="SimSun">
  <w:altName w:val="Arial Unicode MS" />
  …
</w:font>
```

The altName element specifies that when no font with a name of `SimSun` (the primary font name) can be located, that applications should attempt to locate a font with the name `Arial Unicode MS` before doing substitution based on the font metrics. *end example*]

Parent Elements
font (§2.8.2.10)

Attributes	Description
val (String Value)	Specifies that its contents will contain a string. The contents of this string are interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML fragment:

Attributes	Description
	<w:pPr> <w:pStyle w:val="heading1" /> </w:pPr> The value of the val attribute is the ID of the associated paragraph style's styleId. However, consider the following fragment: <w:sdtPr> <w:alias w:val="SDT Title Example" /> ... </w:sdtPr> In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.8.2.2 charset (Character Set Supported By Font)

This element specifies the character set which is supported by the parent font. This information may be used as defined in font substitution logic to locate an appropriate substitute font when this font is not available. This information is determined by querying the font when present and shall not be modified when the font is not available.

The value of this element shall be interpreted as follows:

Value	Description
0x00	Specifies the ANSI character set.
0x01	Specifies the default character set.
0x02	Specifies the Symbol character set.
0x4D	Specifies a Macintosh (Standard Roman) character set.
0x80	Specifies the JIS character set.
0x81	Specifies the Hangul character set.
0x82	Specifies a Johab character set.
0x86	Specifies the GB-2312 character set.
0x88	Specifies the Chinese Big Five character set.

Value	Description
0xA1	Specifies a Greek character set.
0xA2	Specifies a Turkish character set.
0xA3	Specifies a Vietnamese character set.
0xB1	Specifies a Hebrew character set.
0xB2	Specifies an Arabic character set.
0xBA	Specifies a Baltic character set.
0xCC	Specifies a Russian character set.
0xDE	Specifies a Thai character set.
0xEE	Specifies an Eastern European character set.
0xFF	Specifies an OEM character set not defined by this Office Open XML Standard.
Any other value	Application-defined, may be ignored.

If this element is not present, then the character set for this font shall be assumed to be the ANSI character set.

[*Example*: Consider the following information stored for a single font:

```
<w:font w:name="SimSun">
  <w:charset w:val="86" />
  …
</w:font>
```

The charset element specifies via its val attribute value of 86 that this font uses the GB-2312 character set. *end example*]

Parent Elements
font (§2.8.2.10)

Attributes	Description
val (Value)	Specifies a value specified as single octet (two digit) hexadecimal number whose contents are interpreted based on the context of the parent XML element. [*Example*: Consider the following value for an attribute of type ST_UCharHexNumber: `<w:… w:val="BE"/>` This value is valid, as it contains two hexadecimal digits, an encoding of an octet of the actual decimal number value. *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple

Attributes	Description
	type (§2.18.106).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_UcharHexNumber">
   <attribute name="val" type="ST_UcharHexNumber" use="required"/>
</complexType>
```

2.8.2.3 embedBold (Bold Style Font Style Embedding)

This element specifies information about the embedded font storage for the bold form of a font, when it is embedded. This form is used when bold is applied to a text run.

If this element is omitted, then no bold form of the font is stored in the document. The relationship targeted by the id attribute must be of the embedded font type, or the document shall be considered to be invalid.

[*Example*: Consider a WordprocessingML document in which the Arial font has been embedded in the file. This status would be specified using the following WordprocessingML:

```
<w:font w:name="Arial">
  …
  <w:embedBold r:id="rId10" />
</w:font>
```

The embedBold element specifies that the embedded font targeted with the relationship with ID `rId10` may be used to retrieve the bold form of the embedded Arial font. *end example*]

Parent Elements
font (§2.8.2.10)

Attributes	Description

fontKey (Embedded Font Obfuscation Key)	Specifies the key which was used to obfuscate this embedded font. This key may be used to retrieve the embedded font for the purposes of viewing this WordprocessingML document only, using the algorithm described in §2.8.1.
	
	If this attribute is omitted, then no key is provided for this font.
	
	[*Example*: Consider a WordprocessingML document in which the Arial font has been embedded in the file. This status would be specified using the following WordprocessingML:
	
	`<w:font w:name="Arial">`
	`  …`
	`  <w:embedRegular r:id="rId10" w:fontKey="{302EE813-EB4A-4642-A93A-89EF99B2457E}" />`

Attributes	Description
	</w:font> The fontKey attribute has a value of {302EE813-EB4A-4642-A93A-89EF99B2457E}, therefore the embedded Arial font targeted with the relationship with ID rId10 may be retrieved if needed by using this key and the algorithm above. *end example*] The possible values for this attribute are defined by the ST_Guid simple type (§2.18.39).
id (Relationship to Part) Namespace: .../officeDocument/2006/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the type required by the parent element: • http://schemas.openxmlformats.org/officeDocument/2006/relationships/footer for the footerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/header for the headerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://schemas.openxmlformats.org/officeDocument/2006/relationships/printerSettings for the printerSettings element [*Example*: Consider an XML element which has the following id attribute: <… r:id="rId10" /> The markup specifies the associated relationship part with relationship ID rId1 contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
subsetted (Embedded Font Is Subsetted)	Specifies that the embedded font targeted by the id attribute has been subsetted. *Subsetting* is a mechanism by which only the glyphs used in the contents of this WordprocessingML document are stored in an embedded font, in order to prevent the file from becoming unnecessarily large from the use of a small number of glyphs from a large embedded font. If this attribute is omitted, then the embedded font target by the id attribute shall not be handled as though it is subsetted. [*Example*: Consider a WordprocessingML document in which the Arial font has been embedded in the file after subsetting. This status would be specified using the following WordprocessingML: <w:font w:name="Arial"> … <w:embedRegular r:id="rId10" w:subsetted="true" /> </w:font>

Attributes	Description
	The subsetted attribute has a value of `true`, therefore the embedded Arial font targeted with the relationship with ID `rId10` shall be treated as a subsetted font. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FontRel">
   <complexContent>
      <extension base="CT_Rel">
         <attribute name="fontKey" type="ST_Guid"/>
         <attribute name="subsetted" type="ST_OnOff"/>
      </extension>
   </complexContent>
</complexType>
```

2.8.2.4 embedBoldItalic (Bold Italic Font Style Embedding)

This element specifies information about the embedded font storage for the bold italic form of a font, when it is embedded. This form is used when bold and italics are applied to a text run.

If this element is omitted, then no bold italic form of the font is stored in the document.

[*Example*: Consider a WordprocessingML document in which the Arial font has been embedded in the file. This status would be specified using the following WordprocessingML:

```
<w:font w:name="Arial">
  …
  <w:embedBoldItalic r:id="rId11" />
</w:font>
```

The embedBoldItalic element specifies that the embedded font targeted with the relationship with ID `rId11` may be used to retrieve the bold italic form of the embedded Arial font. *end example*]

Parent Elements
font (§2.8.2.10)

Attributes	Description
fontKey (Embedded Font Obfuscation Key)	Specifies the key which was used to obfuscate this embedded font. This key may be used to retrieve the embedded font for the purposes of viewing this WordprocessingML document only, using the algorithm described in §2.8.1. If this attribute is omitted, then no key is provided for this font. [*Example*: Consider a WordprocessingML document in which the Arial font has been

Attributes	Description

	embedded in the file. This status would be specified using the following WordprocessingML:
	
	`<w:font w:name="Arial">`
	`  …`
	`  <w:embedRegular r:id="rId10" w:fontKey="{302EE813-EB4A-4642-A93A-89EF99B2457E}" />`
	`</w:font>`
	
	The fontKey attribute has a value of `{302EE813-EB4A-4642-A93A-89EF99B2457E}`, therefore the embedded Arial font targeted with the relationship with ID `rId10` may be retrieved if needed by using this key and the algorithm above. *end example*]
	
	The possible values for this attribute are defined by the ST_Guid simple type (§2.18.39).

id (Relationship to Part) Namespace: …/officeDocument/2006/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the type required by the parent element: • http://schemas.openxmlformats.org/officeDocument/2006/relationships/footer for the footerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/header for the headerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://schemas.openxmlformats.org/officeDocument/2006/relationships/printerSettings for the printerSettings element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
subsetted (Embedded Font Is Subsetted)	Specifies that the embedded font targeted by the id attribute has been subsetted. *Subsetting* is a mechanism by which only the glyphs used in the contents of this WordprocessingML document are stored in an embedded font, in order to prevent the file from becoming unnecessarily large from the use of a small number of glyphs from a large embedded font. If this attribute is omitted, then the embedded font target by the id attribute shall not be handled as though it is subsetted. [*Example*: Consider a WordprocessingML document in which the Arial font has been

Attributes	Description

	embedded in the file after subsetting. This status would be specified using the following WordprocessingML:
	
	`<w:font w:name="Arial">`
	`…`
	`<w:embedRegular r:id="rId10" w:subsetted="true" />`
	`</w:font>`
	
	The subsetted attribute has a value of `true`, therefore the embedded Arial font targeted with the relationship with ID `rId10` shall be treated as a subsetted font. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FontRel">
   <complexContent>
      <extension base="CT_Rel">
         <attribute name="fontKey" type="ST_Guid"/>
         <attribute name="subsetted" type="ST_OnOff"/>
      </extension>
   </complexContent>
</complexType>
```

2.8.2.5 embedItalic (Italic Font Style Embedding)

This element specifies information about the embedded font storage for the italic form of a font, when it is embedded. This form is used when italics are applied to a text run.

If this element is omitted, then no italic form of the font is stored in the document.

[*Example*: Consider a WordprocessingML document in which the Arial font has been embedded in the file. This status would be specified using the following WordprocessingML:

```
<w:font w:name="Arial">
  …
  <w:embedItalic r:id="rId12" />
</w:font>
```

The embedItalic element specifies that the embedded font targeted with the relationship with ID `rId12` may be used to retrieve the italic form of the embedded Arial font. *end example*]

Parent Elements
font (§2.8.2.10)

Attributes	Description

Attributes	Description

fontKey (Embedded Font Obfuscation Key)	Specifies the key which was used to obfuscate this embedded font. This key may be used to retrieve the embedded font for the purposes of viewing this WordprocessingML document only, using the algorithm described in §2.8.1.
	
	If this attribute is omitted, then no key is provided for this font.
	
	[*Example*: Consider a WordprocessingML document in which the Arial font has been embedded in the file. This status would be specified using the following WordprocessingML:
	
	`<w:font w:name="Arial">`
	`  …`
	`  <w:embedRegular r:id="rId10" w:fontKey="{302EE813-EB4A-4642-A93A-89EF99B2457E}" />`
	`</w:font>`
	
	The fontKey attribute has a value of `{302EE813-EB4A-4642-A93A-89EF99B2457E}`, therefore the embedded Arial font targeted with the relationship with ID `rId10` may be retrieved if needed by using this key and the algorithm above. *end example*]
	
	The possible values for this attribute are defined by the ST_Guid simple type (§2.18.39).

id (Relationship to Part) Namespace: .../officeDocument/2006/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the type required by the parent element: • http://schemas.openxmlformats.org/officeDocument/2006/relationships/footer for the footerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/header for the headerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://schemas.openxmlformats.org/officeDocument/2006/relationships/printerSettings for the printerSettings element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
subsetted (Embedded Font Is Subsetted)	Specifies that the embedded font targeted by the id attribute has been subsetted. *Subsetting* is a mechanism by which only the glyphs used in the contents of this WordprocessingML document are stored in an embedded font, in order to prevent the

Attributes	Description

	file from becoming unnecessarily large from the use of a small number of glyphs from a large embedded font.
	
	If this attribute is omitted, then the embedded font target by the id attribute shall not be handled as though it is subsetted.
	
	[*Example*: Consider a WordprocessingML document in which the Arial font has been embedded in the file after subsetting. This status would be specified using the following WordprocessingML:
	
	`<w:font w:name="Arial">`
	`…`
	`<w:embedRegular r:id="rId10" w:subsetted="true" />`
	`</w:font>`
	
	The subsetted attribute has a value of `true`, therefore the embedded Arial font targeted with the relationship with ID `rId10` shall be treated as a subsetted font. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FontRel">
   <complexContent>
      <extension base="CT_Rel">
         <attribute name="fontKey" type="ST_Guid"/>
         <attribute name="subsetted" type="ST_OnOff"/>
      </extension>
   </complexContent>
</complexType>
```

2.8.2.6 embedRegular (Regular Font Style Embedding)

This element specifies information about the embedded font storage for the regular form of a font, when it is embedded. This form is used when neither bold nor italics is applied to a text run.

If this element is omitted, then no regular form of the font is stored in the document.

[*Example*: Consider a WordprocessingML document in which the Arial font has been embedded in the file. This status would be specified using the following WordprocessingML:

```
<w:font w:name="Arial">
  …
  <w:embedRegular r:id="rId13" />
</w:font>
```

The embedRegular element specifies that the embedded font targeted with the relationship with ID `rId13` may be used to retrieve the regular form of the embedded Arial font. *end example*]

Parent Elements
font (§2.8.2.10)

Attributes	Description

fontKey (Embedded Font Obfuscation Key)	Specifies the key which was used to obfuscate this embedded font. This key may be used to retrieve the embedded font for the purposes of viewing this WordprocessingML document only, using the algorithm described in §2.8.1.
	
	If this attribute is omitted, then no key is provided for this font.
	
	[*Example*: Consider a WordprocessingML document in which the Arial font has been embedded in the file. This status would be specified using the following WordprocessingML:
	
	`<w:font w:name="Arial">`
	`  …`
	`  <w:embedRegular r:id="rId10" w:fontKey="{302EE813-EB4A-4642-A93A-89EF99B2457E}" />`
	`</w:font>`
	
	The fontKey attribute has a value of `{302EE813-EB4A-4642-A93A-89EF99B2457E}`, therefore the embedded Arial font targeted with the relationship with ID `rId10` may be retrieved if needed by using this key and the algorithm above. *end example*]
	
	The possible values for this attribute are defined by the ST_Guid simple type (§2.18.39).

id (Relationship to Part) Namespace: .../officeDocument/2006/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the type required by the parent element: • http://schemas.openxmlformats.org/officeDocument/2006/relationships/footer for the footerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/header for the headerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://schemas.openxmlformats.org/officeDocument/2006/relationships/printerSettings for the printerSettings element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

subsetted (Embedded Font Is Subsetted)	Specifies that the embedded font targeted by the id attribute has been subsetted. *Subsetting* is a mechanism by which only the glyphs used in the contents of this WordprocessingML document are stored in an embedded font, in order to prevent the file from becoming unnecessarily large from the use of a small number of glyphs from a large embedded font.
	
	If this attribute is omitted, then the embedded font target by the id attribute shall not be handled as though it is subsetted.
	
	[*Example*: Consider a WordprocessingML document in which the Arial font has been embedded in the file after subsetting. This status would be specified using the following WordprocessingML:
	
	`<w:font w:name="Arial">`
	`…`
	`<w:embedRegular r:id="rId10" w:subsetted="true" />`
	`</w:font>`
	
	The subsetted attribute has a value of `true`, therefore the embedded Arial font targeted with the relationship with ID `rId10` shall be treated as a subsetted font. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FontRel">
   <complexContent>
      <extension base="CT_Rel">
         <attribute name="fontKey" type="ST_Guid"/>
         <attribute name="subsetted" type="ST_OnOff"/>
      </extension>
   </complexContent>
</complexType>
```

2.8.2.7 embedSystemFonts (Embed Common System Fonts)

This element specifies that applications shall embed common system fonts when they are in use and font embedding is enabled for this document using the embedTrueTypeFonts element (§2.8.2.8). *Common system fonts* refer to a set of fonts which are typically always present on a machine, and are not defined by this Office Open XML Standard.

If this element is omitted, then the set of fonts defined as common system fonts should not be embedded in the current document when font embedded is turned on. If the embedTrueTypeFonts element is omitted or `false`, then this setting has no effect.

[*Example*: Consider a WordprocessingML document that specifies that it shall embed fonts, including common system fonts. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:embedTrueTypeFonts w:val="true" />
<w:embedSystemFonts w:val="true"/>
```

The embedSystemFonts element's val attribute has a value of `true` specifying that common system fonts should be included in this document when they are used. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.8.2.8 embedTrueTypeFonts (Embed TrueType Fonts)

This element specifies that applications shall embed the fonts in use in this document when it is saved. These fonts shall be embedded subject to the algorithm specified in §2.8.1.

If this element is omitted, then fonts in use should not be embedded in the current document.

[*Example*: Consider a WordprocessingML document that specifies that it shall embed fonts, including common system fonts. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:embedTrueTypeFonts w:val="true" />
<w:embedSystemFonts w:val="true"/>
```

The embedTrueType element's val attribute has a value of `true` specifying that fonts should be embedded in this document when they are used. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.8.2.9 family (Font Family)

This element specifies the font family of the current font. This information may be used as defined in font substitution logic to locate an appropriate substitute font when this font is not available. This information is determined by querying the font when present and shall not be modified when the font is not available.

If this element is omitted, then its value shall be assumed to be `auto`.

[*Example*: Consider the following information stored for a single font:

```
<w:font w:name="Calibri">
  <w:family w:val="swiss" />
  …
</w:font>
```

The family element specifies via its val attribute value of swiss that this font is part of the Swiss family. *end example*]

Parent Elements
font (§2.8.2.10)

Attributes	Description

val (Font Family Value)	Specifies the font family for the parent font.
	
	[*Example*: Consider the following information stored for a single font:
	
	`<w:font w:name="Times New Roman">`
	`  <w:family w:val="roman" />`
	`  …`
	`</w:font>`
	
	The val attribute value of swiss that this font is part of the Roman family. *end example*]
	
	The possible values for this attribute are defined by the ST_FontFamily simple type (§2.18.34).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FontFamily">
   <attribute name="val" type="ST_FontFamily" use="required"/>
</complexType>
```

2.8.2.10 font (Properties for a Single Font)

This element specifies the properties for one of the fonts used in this document. A font element shall be written out for each font face used in the document, and includes:

- The name of the font as used in the document's stories
- (optionally) Font metrics allowing other applications to locate appropriate substitute fonts as needed
- (optionally) Embedded forms of the font

[*Example*: Consider the following information stored for a single font:

```
<w:font w:name="Times New Roman">
  <w:panose1 w:val="02020603050405020304" />
  <w:charset w:val="00" />
  <w:family w:val="roman" />
  <w:pitch w:val="variable" />
  <w:sig w:usb0="20002A87" w:usb1="80000000" w:usb2="00000008" w:usb3="00000000"
w:csb0="000001FF" w:csb1="00000000" />
</w:font>
```

The font element contains information about the `Times New Roman` font; specifically, information used to locate a substitute font if it is not available. *end example*]

Parent Elements
fonts (§2.8.2.11)

Child Elements	Subclause
altName (Alternate Names for Font)	§2.8.2.1
charset (Character Set Supported By Font)	§2.8.2.2
embedBold (Bold Style Font Style Embedding)	§2.8.2.3
embedBoldItalic (Bold Italic Font Style Embedding)	§2.8.2.4
embedItalic (Italic Font Style Embedding)	§2.8.2.5
embedRegular (Regular Font Style Embedding)	§2.8.2.6
family (Font Family)	§2.8.2.9
notTrueType (Raster or Vector Font)	§2.8.2.12
panose1 (Pansose-1 Typeface Classification Number)	§2.8.2.13
pitch (Font Pitch)	§2.8.2.14
sig (Supported Unicode Subranges and Code Pages)	§2.8.2.16

Attributes	Description

name (Primary Font Name)	Specifies the primary name of the current font. This name shall be used to link the information stored in this element with uses of this value in the rFonts element (§2.3.2.24) in document content.
	
	[*Example*: Consider the following information stored for a single font:
	
	`<w:font w:name="Times New Roman">`
	`  …`
	`</w:font>`
	
	The name attribute specifies that the information contained in this element shall be used to look up information about all uses of the `Times New Roman` font in the document contents. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Font">
   <sequence>
      <element name="altName" type="CT_String" minOccurs="0" maxOccurs="1"/>
      <element name="panose1" type="CT_Panose" minOccurs="0" maxOccurs="1"/>
      <element name="charset" type="CT_UcharHexNumber" minOccurs="0" maxOccurs="1"/>
      <element name="family" type="CT_FontFamily" minOccurs="0" maxOccurs="1"/>
      <element name="notTrueType" type="CT_OnOff" minOccurs="0" maxOccurs="1"/>
      <element name="pitch" type="CT_Pitch" minOccurs="0" maxOccurs="1"/>
      <element name="sig" type="CT_FontSig" minOccurs="0" maxOccurs="1"/>
      <element name="embedRegular" type="CT_FontRel" minOccurs="0" maxOccurs="1"/>
      <element name="embedBold" type="CT_FontRel" minOccurs="0" maxOccurs="1"/>
      <element name="embedItalic" type="CT_FontRel" minOccurs="0" maxOccurs="1"/>
      <element name="embedBoldItalic" type="CT_FontRel" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="name" type="ST_String" use="required"/>
</complexType>
```

2.8.2.11 fonts (Font Table Root Element)

This element specifies the root element for a font table part within a WordprocessingML document, and specifies information about the fonts used in this document, each contained within a child font element.

[*Example*: Consider the following information stored in a font table part:

```
<w:fonts>
  <w:font w:name="Times New Roman">
    …
  </w:font>
  <w:font w:name="Arial">
    …
  </w:font>
</w:fonts>
```

The fonts element contains information about all fonts used in the document - in this example, the `Times New Roman` and `Arial` fonts. *end example*]

Parent Elements
Root element of WordprocessingML Font Table part

Child Elements	Subclause
font (Properties for a Single Font)	§2.8.2.10

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FontsList">
   <sequence>
      <element name="font" type="CT_Font" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

2.8.2.12 notTrueType (Raster or Vector Font)

This element specifies that this font is not a TrueType or OpenType font, but is rather a raster or vector font. This information may be used as defined in font substitution logic to locate an appropriate substitute font when this font is not available. This information is determined by querying the font when present and shall not be modified when the font is not available.

If this element is omitted, then the font shall be assumed to be a TrueType or OpenType font.

[*Example*: Consider the following information stored for a single font:

```
<w:font w:name="JonsFont">
  <w:notTrueType w:val="true" />
  …
</w:font>
```

The notTrueType element specifies via its val attribute value of `true` that this font is a raster or vector font. *end example*]

Parent Elements
font (§2.8.2.10)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.8.2.13 panose1 (Pansose-1 Typeface Classification Number)

This element specifies the Panose-1 classification number for the current font using the PANOSE Classification Guide, Version 1.2. This information may be used as defined in font substitution logic to locate an appropriate substitute font when this font is not available. This information is determined by querying the font when present and shall not be modified when the font is not available.

If this element is omitted, then no Panose-1 information is available.

[*Example*: Consider the following information stored for a single font:

```
<w:font w:name="Times New Roman">
  <w:panose1 w:val="02020603050405020304" />
  …
</w:font>
```

The panose1 element specifies its Panose-1 number via its val attribute value of `02020603050405020304`. *end example*]

Parent Elements
font (§2.8.2.10)

Attributes	Description
val (Value)	Specifies the Panose-1 classification number for the font, stored as a series of two digit hexadecimal encodings of each digits of the Panose number. [*Example*: Consider the following information stored for a single font: `<w:panose1 w:val="020F0603050405020304" />` The val attribute specifies that the digits in the Panose-1 number are: 2,15,6,3,5,2,3,4. *end example*] The possible values for this attribute are defined by the ST_Panose simple type (§2.18.72).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Panose">
   <attribute name="val" type="ST_Panose" use="required"/>
</complexType>
```

2.8.2.14 pitch (Font Pitch)

This element specifies the font pitch of the current font. This information may be used as defined in font substitution logic to locate an appropriate substitute font when this font is not available. This information is determined by querying the font when present and shall not be modified when the font is not available.

If this element is omitted, then its value shall be assumed to be `default`.

[*Example*: Consider the following information stored for a single font:

```
<w:font w:name="Courier New">
  <w:pitch w:val="fixed" />
  …
</w:font>
```

The pitch element specifies via its val attribute value of `fixed` that this is a fixed width font. *end example*]

Parent Elements
font (§2.8.2.10)

Attributes	Description
val (Value)	Specifies the font pitch for the font. [*Example*: Consider the following information stored for a single font: `<w:pitch w:val="variable" />` The val attribute value of `variable` specifies that this is a variable width font. *end example*] The possible values for this attribute are defined by the ST_Pitch simple type (§2.18.73).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Pitch">
   <attribute name="val" type="ST_Pitch" use="required"/>
</complexType>
```

2.8.2.15 saveSubsetFonts (Subset Fonts When Embedding)

This element specifies that applications shall subset fonts when font embedding is enabled for this document using the embedTrueTypeFonts element (§2.8.2.8). *Subsetting* is a mechanism by which only the glyphs used in the contents of this WordprocessingML document are stored in an embedded font, in order to prevent the file from becoming unnecessarily large from the use of a small number of glyphs from a large embedded font.

If this element is omitted, then the set of fonts should not be subsetted in the current document when font embedded is turned on. If the embedTrueTypeFonts element is omitted or `false`, then this setting has no effect.

[*Example*: Consider a WordprocessingML document that specifies that it shall subset embedded fonts. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:embedTrueTypeFonts w:val="true" />
<w:saveSubsetFonts w:val="true"/>
```

The embedSystemFonts element's val attribute has a value of `true` specifying fonts should be subsetted in this document when they are embedded. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.8.2.16 sig (Supported Unicode Subranges and Code Pages)

This element specifies information identifying the code pages and Unicode subranges for which the parent font provides glyphs. This information may be used as defined in font substitution logic to locate an appropriate substitute font when this font is not available. This information is determined by querying the font when present and shall not be modified when the font is not available.

When storing Unicode subrange information, the appropriate bit in the bitfield shall only be set if the entire subrange is supported by that font.

If this element is omitted, then no supported code page/Unicode subrange information is available.

[*Example*: Consider the following information stored for a single font:

```
<w:font w:name="Times New Roman">
  <w:sig w:usb0="20002A87" w:usb1="80000000" w:usb2="00000008" w:usb3="00000000"
w:csb0="000001FF" w:csb1="00000000" />
  …
</w:font>
```

The sig element specifies the supported code pages and Unicode sub ranges via its attributes. For example, the code pages supported are:

- Latin 1
- Latin 2: Eastern Europe
- Cyrillic
- Greek
- Turkish
- Baltic

end example]

Parent Elements
font (§2.8.2.10)

Attributes	Description
csb0 (Lower 32 Bits of Code Page Bit Field)	Specifies a four digit hexadecimal encoding of the first 32 bits of the 64-bit code-page bit field that identifies which specific character sets or code pages are supported by the parent font. Each bit in this 32 bits represents the following code page:

Bit	Description
0	Latin 1
1	Latin 2: Eastern Europe
2	Cyrillic
3	Greek
4	Turkish
5	Hebrew
6	Arabic

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>7</td><td>Windows Baltic</td></tr>
<tr><td>8 to 16</td><td>Reserved for Alternate ANSI</td></tr>
<tr><td>17</td><td>Thai</td></tr>
<tr><td>18</td><td>JIS/Japan</td></tr>
<tr><td>19</td><td>Chinese (Simplified)</td></tr>
<tr><td>20</td><td>Korean Wansung</td></tr>
<tr><td>21</td><td>Chinese (Traditional)</td></tr>
<tr><td>22 to 29</td><td>Reserved for Alternate ANSI and OEM</td></tr>
<tr><td>30</td><td>Macintosh Character Set (Standard Roman)</td></tr>
<tr><td>31</td><td>Symbol Character Set</td></tr>
</table>
[Example: Consider font information specified as follows:
<pre>
<w:font w:name="Lucida Console">
  <w:sig w:csb0="0000001F" … />
  …
</w:font>
</pre>
The csb0 attribute value of 0000001F specifies that the following code pages are supported by this font:

Latin 1
Latin 2: Eastern Europe
Cyrillic
Greek
Turkish

end example]
The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).
</td></tr>
<tr><td>csb1 (Upper 32 Bits of Code Page Bit Field)</td><td>
Specifies a four digit hexadecimal encoding of the upper 32 bits of the 64-bit code-page bit field that identifies which specific character sets or code pages are supported by the parent font.
Each bit in this 32 bits represents the following code page:
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0 to 15</td><td>Reserved for OEM</td></tr>
<tr><td>16</td><td>IBM Greek</td></tr>
<tr><td>17</td><td>MS-DOS Russian</td></tr>
<tr><td>18</td><td>MS-DOS Nordic</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>19</td><td>Arabic</td></tr>
<tr><td>20</td><td>MS-DOS Canadian French</td></tr>
<tr><td>21</td><td>Hebrew</td></tr>
<tr><td>22</td><td>MS-DOS Icelandic</td></tr>
<tr><td>23</td><td>MS-DOS Portuguese</td></tr>
<tr><td>24</td><td>IBM Turkish</td></tr>
<tr><td>25</td><td>IBM Cyrillic</td></tr>
<tr><td>26</td><td>Latin 2</td></tr>
<tr><td>27</td><td>MS-DOS Baltic</td></tr>
<tr><td>28</td><td>Greek (former 437G)</td></tr>
<tr><td>29</td><td>Arabic (AMSO 708)</td></tr>
<tr><td>30</td><td>WE/Latin 1</td></tr>
<tr><td>31</td><td>US</td></tr>
</table>
[Example: Consider font information specified as follows:
<pre>
<w:font w:name="Lucida Console">
  <w:sig w:csb1="00000000" … />
  …
</w:font>
</pre>
The csb1 attribute value of 00000000 specifies that none of the specified code pages are supported by this font. end example]
The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).
</td></tr>
<tr><td>usb0 (First 32 Bits of Unicode Subset Bitfield)</td><td>
Specifies the first 32 bits of the 128-bit Unicode subset bit field (USB). Subranges are ordered in accordance with the ISO 10646 standard.
[Example: Consider font information specified as follows:
<pre>
<w:font w:name="Times New Roman">
  <w:sig w:usb0="20002A87" … />
  …
</w:font>
</pre>
The usb0 attribute value of 20002A87 specifies that the first 32 bits of the bitfield are 00100000000000000010101010000111, which corresponds to:

Basic Latin
Latin-1 Supplement
Latin Extended-A

</td></tr>
</table>

Attributes	Description
	• Basic Greek • Cyrillic • Basic Hebrew • Basic Arabic • Latin Extended Additional *end example*] The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).

usb1 (Second 32 Bits of Unicode Subset Bitfield)	Specifies the second 32 bits of the 128-bit Unicode subset bit field (USB). Subranges are ordered in accordance with the ISO 10646 standard.
	
	[*Example*: Consider font information specified as follows:
	
	`<w:font w:name="Times New Roman">`
	`  <w:sig w:usb1="80000000" … />`
	`  …`
	`</w:font>`
	
	The usb0 attribute value of `80000000` specifies that the first 32 bits of the bitfield are 10000000000000000000000000000000, which corresponds to:
	• Arabic Presentation Forms-A
	
	end example]
	
	The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).

usb2 (Third 32 Bits of Unicode Subset Bitfield)	Specifies the third 32 bits of the 128-bit Unicode subset bit field (USB). Subranges are ordered in accordance with the ISO 10646 standard.
	
	[*Example*: Consider font information specified as follows:
	
	`<w:font w:name="Times New Roman">`
	`  <w:sig w:usb2="00000008" … />`
	`  …`
	`</w:font>`
	
	The usb0 attribute value of `80000000` specifies that the first 32 bits of the bitfield are 00000000000000000000000000001000, which corresponds to:
	• Arabic Presentation Forms-B
	
	end example]
	
	The possible values for this attribute are defined by the ST_LongHexNumber simple type

Attributes	Description
	(§2.18.57).

usb3 (Fourth 32 Bits of Unicode Subset Bitfield)	Specifies the fourth 32 bits of the 128-bit Unicode subset bit field (USB). Subranges are ordered in accordance with the ISO 10646 standard.
	
	[*Example*: Consider font information specified as follows:
	
	`<w:font w:name="Times New Roman">`
	`  <w:sig w:usb3="00000000" … />`
	`  …`
	`</w:font>`
	
	The usb3 attribute value of `00000000` specifies that the first 32 bits of the bitfield are 00000000000000000000000000000000, which corresponds to no subranges. *end example*]
	
	The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FontSig">
   <attribute name="usb0" use="required" type="ST_LongHexNumber"/>
   <attribute name="usb1" use="required" type="ST_LongHexNumber"/>
   <attribute name="usb2" use="required" type="ST_LongHexNumber"/>
   <attribute name="usb3" use="required" type="ST_LongHexNumber"/>
   <attribute name="csb0" use="required" type="ST_LongHexNumber"/>
   <attribute name="csb1" use="required" type="ST_LongHexNumber"/>
</complexType>
```

2.9 Numbering

Numbering refers to symbols - Arabic numerals, Roman numerals, symbol characters ("bullets"), text strings, etc. - in WordprocessingML that are used to label individual paragraphs of text.

[*Example*: The following two paragraphs each contain numbering as defined by WordprocessingML: the first uses an Arabic numeral, the second a symbol character:

9. This is a paragraph with numbering information.

- This is also a paragraph with numbering information.

end example]

The basis for all numbering in WordprocessingML is specified via two structures:

- abstract numbering definitions
- numbering definition instances

Abstract numbering definitions define the appearance and behavior of a specific set of numbered paragraphs in a document. Because this construct is abstract, they are not be directly referenced by document content, but rather they must be inherited by a *numbering definition instance*, which itself is referenced by document content.

[*Example*: Consider the following example of an abstract numbering definition in a WordprocessingML document:

```
<w:abstractNum w:abstractNumId="4">
  <w:nsid w:val="FFFFFF7F" />
  <w:multiLevelType w:val="singleLevel" />
  <w:lvl w:ilvl="0">
    <w:start w:val="1" />
    <w:lvlText w:val="%1." />
    <w:lvlJc w:val="left" />
    <w:pPr>
      <w:tabs>
        <w:tab w:val="num" w:pos="720" />
      </w:tabs>
      <w:ind w:left="720" w:hanging="360" />
    </w:pPr>
  </w:lvl>
</w:abstractNum>
```

This abstractNum element defines an abstract numbering definition which defines a set of numbering properties. It is inherited by any numbering definition instance which inherits from an abstractNumId equal to 4:

```
<w:num w:numId="2">
    <w:abstractNumId w:val="4" />
</w:num>
```

This num element defines an numbering definition instance which may define overrides to the abstract numbering definition (in this case it does not), and is used by any paragraphs with a numId equal to 2:

```
<w:p>
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="0" />
      <w:numId w:val="2" />
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>Level one</w:t>
  </w:r>
</w:p>
```

The resulting paragraph will inherit the properties of level `0` in the numbering definition instance of `2` which is simply a instance of the abstract numbering definition of `4`. *end example*]

2.9.1 abstractNum (Abstract Numbering Definition)

This element specifies a set of properties which shall dictate the appearance and behavior of a set of numbered paragraphs in a WordprocessingML document. These properties are collectively called an *abstract numbering definition*, and are the basis for all numbering information in a WordprocessingML document.

Although an abstract numbering definition contains a complete set of numbering, it shall not be directly referenced by content (hence the use of *abstract*). Instead, these properties shall be inherited by a numbering definition instance using the num element (§2.9.16), which can then itself be referenced by content.

[*Example*: Consider the following example of an abstractNum in a WordprocessingML document:

```
<w:abstractNum w:abstractNumId="4">
  <w:nsid w:val="FFFFFF7F" />
  <w:multiLevelType w:val="singleLevel" />
  <w:lvl w:ilvl="0">
    <w:start w:val="1" />
    <w:lvlText w:val="%1." />
    <w:lvlJc w:val="left" />
    <w:pPr>
      <w:tabs>
        <w:tab w:val="num" w:pos="720" />
      </w:tabs>
      <w:ind w:left="720" w:hanging="360" />
    </w:pPr>
  </w:lvl>
</w:abstractNum>
```

This abstractNum element defines an abstract numbering definition which shall be inherited by any numbering definition instance which inherits from abstract numbering definition with an abstractNumId equal to 4. *end example*]

Parent Elements
numbering (§2.9.17)

Child Elements	Subclause
lvl (Numbering Level Definition)	§2.9.7
multiLevelType (Abstract Numbering Definition Type)	§2.9.13
name (Abstract Numbering Definition Name)	§2.9.14
nsid (Abstract Numbering Definition Identifier)	§2.9.15
numStyleLink (Numbering Style Reference)	§2.9.22
styleLink (Numbering Style Definition)	§2.9.29
tmpl (Numbering Template Code)	§2.9.31

Attributes	Description

abstractNumId (Abstract Numbering Definition ID)	Specifies a unique number which shall be used as the identifier for this abstract numbering definition. This unique number shall be referenced by any numbering definition instance in order to inherit the properties specified by this abstract numbering definition.
	
	[*Example*: Consider the WordprocessingML for an abstract numbering definition with an abstractNumId attribute of 4:
	
	`<w:abstractNum w:abstractNumId="4">`
	`  <w:nsid w:val="FFFFFF7F" />`
	`  <w:multiLevelType w:val="singleLevel" />`
	`  <w:lvl w:ilvl="0">`
	`  <w:start w:val="1" />`
	`  <w:lvlText w:val="%1." />`
	`  <w:lvlJc w:val="left" />`
	`  <w:pPr>`
	`  <w:tabs>`
	`  <w:tab w:val="num" w:pos="720" />`
	`  </w:tabs>`
	`  <w:ind w:left="720"/>`
	`  </w:pPr>`
	`  </w:lvl>`
	`</w:abstractNum>`
	
	The abstractNumId attribute serves as a unique identifier for the abstract numbering

Attributes	Description

	definition, allowing numbering definition instances (§2.9.16) with a abstractNumId element with a matching attribute value to inherit the abstract numbering definition properties, for example:
	
	`<w:numbering>`
	`...`
	`<w:num w:numId="2">`
	`<w:abstractNumId w:val="0" />`
	`</w:num>`
	`<w:num w:numId="3">`
	`<w:abstractNumId w:val="1" />`
	`</w:num>`
	`<w:num w:numId="4">`
	`<w:abstractNumId w:val="4" />`
	`</w:num>`
	`<w:num w:numId="5">`
	`<w:abstractNumId w:val="4" />`
	`</w:num>`
	`</w:numbering>`
	
	In this case, the final two numbering definition instances both inherit from the abstract numbering definition with a abstractNumId of 4. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AbstractNum">
   <sequence>
      <element name="nsid" type="CT_LongHexNumber" minOccurs="0"/>
      <element name="multiLevelType" type="CT_MultiLevelType" minOccurs="0"/>
      <element name="tmpl" type="CT_LongHexNumber" minOccurs="0"/>
      <element name="name" type="CT_String" minOccurs="0"/>
      <element name="styleLink" type="CT_String" minOccurs="0"/>
      <element name="numStyleLink" type="CT_String" minOccurs="0"/>
      <element name="lvl" type="CT_Lvl" minOccurs="0" maxOccurs="9"/>
   </sequence>
   <attribute name="abstractNumId" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.9.2 abstractNumId (Abstract Numbering Definition Reference)

This element specifies the abstract numbering definition information whose properties shall be inherited by the parent numbering definition instance.

[*Example*: Consider the WordprocessingML for a document with two numbering definition instances, each referencing a different abstract numbering definition:

```
<w:numbering>
  <w:abstractNum w:abstractNumId="0">
  <w:abstractNum w:abstractNumId="1">
  ...
  <w:num w:numId="1">
    <w:abstractNumId w:val="0" />
  </w:num>
  <w:num w:numId="2">
    <w:abstractNumId w:val="1" />
  </w:num>
  ...
</w:numbering>
```

The two numbering definition instances reference the abstract numbering definitions with abstractNumId attribute values of 0 and 1 respectively, via their abstractNumId elements. *end example*]

Parent Elements
num (§2.9.16)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.9.3 ilvl (Numbering Level Reference)

This element specifies the numbering level of the numbering definition instance which shall be applied to the parent paragraph.

This numbering level is specified on either the abstract numbering definition's lvl element (§2.9.7), and may be overridden by a numbering definition instance level override's lvl element (§2.9.6).

[*Example*: Consider the following numbered paragraphs in a WordprocessingML document:

1. Level one
    a. Level two

These numbered paragraphs may be represented using the following WordprocessingML:

```
<w:p>
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="0" />
      <w:numId w:val="5" />
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>Level one</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="1" />
      <w:numId w:val="5" />
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>Level two</w:t>
  </w:r>
</w:p>
```

The WordprocessingML above specifies that the first numbered paragraph references the numbering level of 0, within the numbering definition of the num element (§2.9.16) with a numId attribute equal to 5.

The second numbered paragraph references the numbering of 1, within the same numbering definition instance. The WordprocessingML referenced by the ilvl elements above is given below:

```
<w:num w:numId="5">
  <w:abstractNumId w:val="0" />
</w:num>
...
```

```
<w:abstractNum w:abstractNumId="0">
  <w:nsid w:val="FFFFFF7F" />
  <w:multiLevelType w:val="singleLevel" />
  <w:lvl w:ilvl="0">
    ...
  </w:lvl>
  <w:lvl w:ilvl="1">
    ...
  </w:lvl>
</w:abstractNum>
```

In this case, the resulting paragraphs would inherit the properties of the abstract numbering definition levels with ilvl attributes of `0` and `1`, respectively. *end example*]

Parent Elements
numPr (§2.3.1.19)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.9.4 isLgl (Display All Levels Using Arabic Numerals)

This element specifies whether or not all previous levels displayed for a given numbering level's text shall be displayed using the `decimal` number format, regardless of the actual number format of that level in the list. [*Note*: This numbering style is often referred to as the legal numbering style. *end note*]

If this element is present, then all numbering levels present in the lvlTxt element (§2.9.12) shall be converted to their decimal equivalents when they are displayed in this level in the numbering format. If this element is omitted, then each level is displayed using the numFmt (§2.9.18) of that level.

[*Example*: Consider the numbering set below. In this set of blank numbered paragraphs, three numbering levels have been used and the third has the isLgl property applied, resulting in the following:

```
A
    A.a
    A.b
          1.2.1
          1.2.2
B
    B.a
          2.1.1
```

As shown above, each number in the third level in the list has been converted to its decimal equivalent. The WordprocessingML necessary to turn on the legal numbering rule for the third numbering level is given below:

```
<w:lvl w:ilvl="2">
  ...
  <w:isLgl />
  ...
</w:lvl>
```

end example]

Parent Elements
lvl (§2.9.6); lvl (§2.9.7)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.9.5 legacy (Legacy Numbering Level Properties)

This element specifies that a given numbering level is from an earlier word processing application which did not support the full richness of the numbering properties supported by WordprocessingML.

These properties shall be used to render any numbered paragraph which references this numbering level if the legacy attribute is set. [*Note*: Using this element in generated WordprocessingML documents is not recommended, as updated numbering structures in WordprocessingML should be used in its place. This element is provided solely to save and roundtrip the numbering properties of legacy word processing products in WordprocessingML such that they are recreated if the document is resaved in an older word processor format. *end note*]

[*Example*: Consider the following WordprocessingML numbering level:

```
<w:lvl w:ilvl="0">
  ...
  <w:legacy w:legacySpace="820" w:legacyIndent="960" />
  <w:lvlJc w:val="left" />
  <w:pPr>
    <w:ind w:left="360" w:hanging="360" />
  </w:pPr>
</w:lvl>
```

This level has the legacy element present, therefore the legacy numbering level properties shall be used to format all paragraphs which reference this level. *end example*]

Parent Elements
lvl (§2.9.6); lvl (§2.9.7)

Attributes	Description
legacy (Use Legacy Numbering Properties)	Specifies whether the legacy numbering properties present for this numbering level shall be used to format the numbering for any paragraph which references it. A value of on, 1, or true for this attribute value specifies that the legacy numbering properties shall be applied. This is the default value for this attribute, and is implied when

Attributes	Description
	the attribute is omitted. A value of `off`, `0`, or `false` for this attribute value specifies that the legacy numbering properties shall not be used, and shall be explicitly turned off. [*Example*: For example, consider the set of legacy numbering properties from a document: `<w:legacy w:legacy="off" w:legacySpace="820" w:legacyIndent="960" />` This set of legacy properties are explicitly not used when processing the numbering level via the fact that the legacy attribute is turned off for this example. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
legacyIndent (Legacy Indent)	Specifies the indentation which shall be applied to a legacy numbering symbol from the text margin of the document. This value is specified in twentieths of a point. If this attribute is not present, then no indentation shall be applied with respect to the margin. [*Example*: For example, consider the set of legacy numbering properties from a document: `<w:legacy w:legacySpace="820" w:legacyIndent="960" />` This set of legacy properties specify that there shall be exactly 960 twentieths of a point (⅔ of an inch) between the text margin and the start of the numbering on the paragraph. *end example*] The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§2.18.88).
legacySpace (Legacy Spacing)	Specifies the indentation which shall be applied between a legacy numbering symbol and the accompanying text of the associated paragraph in the document. This value is specified in twentieths of a point. If this attribute is not present, then no indentation shall be applied with respect to the paragraph text. [*Example*: For example, consider the set of legacy numbering properties from a document: `<w:legacy w:legacySpace="820" w:legacyIndent="960" />` This set of legacy properties specify that there shall be exactly 860 twentieths of a point

Attributes	Description
	between the end of the numbering on the paragraph and the associated paragraph text. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LvlLegacy">
   <attribute name="legacy" type="ST_OnOff" use="optional"/>
   <attribute name="legacySpace" type="ST_TwipsMeasure" use="optional"/>
   <attribute name="legacyIndent" type="ST_SignedTwipsMeasure" use="optional"/>
</complexType>
```

2.9.6 lvl (Numbering Level Override Definition)

This element specifies the appearance and behavior of a specific numbering level within a given numbering level definition override defined using the lvlOverride element (§2.9.9).

A numbering level override definition is identical to a numbering level definition, except for the fact that it is defined as part of a numbering definition instance using the num element (§2.9.16) rather than as part of an abstract numbering definition using the abstractNum element (§2.9.1).

[*Example*: Consider a numbering definition instance which inherits its information from the abstract numbering definition with abstractNumId of 4, but should use a different set of properties for level 0 of the numbering definition. The resulting WordprocessingML would look like:

```
<w:num w:numId="6">
  <w:abstractNumId w:val="4" />
  <w:lvlOverride w:ilvl="0">
    <w:lvl w:ilvl="0">
      <w:start w:val="4" />
      <w:lvlText w:val="%1)" />
      <w:lvlJc w:val="left" />
      <w:pPr>
        <w:ind w:left="360" w:hanging="360" />
      </w:pPr>
    </w:lvl>
  </w:lvlOverride>
</w:num>
```

This numbering definition instance overrides level 0 of the list with the specified numbering level override definition, replacing those in the abstract numbering level definition. *end example*]

[*Note*: The ability to set level overrides optimizes use of numbering in WordprocessingML as it prevents writing out redundant abstract numbering definitions if numbering sets only slightly differ.

Consider using WordprocessingML to create two numbered sets that only differ only in the appearance and style of the first numbering level. Both could use the same abstract numbering definition as long as each references a different numbering definition instance with one of the numbering definition instances leveraging a level override for the first numbering level. Below is WordprocessingML that illustrates this:

```
<w:num w:numId="5">
  <w:abstractNumId w:val="4" />
</w:num>
<w:num w:numId="6">
  <w:abstractNumId w:val="4" />
  <w:lvlOverride w:ilvl="0">
    <w:lvl w:ilvl="0">
      <w:start w:val="4" />
      <w:lvlText w:val="%1)" />
      <w:lvlJc w:val="left" />
      <w:pPr>
        <w:ind w:left="360" w:hanging="360" />
      </w:pPr>
    </w:lvl>
  </w:lvlOverride>
</w:num>
```

end note]

Parent Elements
lvlOverride (§2.9.9)

Child Elements	Subclause
isLgl (Display All Levels Using Arabic Numerals)	§2.9.4
legacy (Legacy Numbering Level Properties)	§2.9.5
lvlJc (Justification)	§2.9.8
lvlPicBulletId (Picture Numbering Symbol Definition Reference)	§2.9.10
lvlRestart (Restart Numbering Level Symbol)	§2.9.11
lvlText (Numbering Level Text)	§2.9.12
numFmt (Numbering Format)	§2.9.18
pPr (Numbering Level Associated Paragraph Properties)	§2.9.24
pStyle (Paragraph Style's Associated Numbering Level)	§2.9.25
rPr (Numbering Symbol Run Properties)	§2.9.26
start (Starting Value)	§2.9.27
suff (Content Between Numbering Symbol and Paragraph Text)	§2.9.30

Attributes	Description

ilvl (Numbering Level)	Specifies the numbering level definition that is to be defined by this set of numbering properties.
	
	This override is a zero-based index of the number of list levels in the document. [*Example*: A value of 2 is the 3rd list level in the document. *end example*]
	
	[*Example*: Consider the following WordprocessingML for a numbering definition instance:
	
	`<w:num w:numId="6">`
	`  <w:abstractNumId w:val="4" />`
	`  <w:lvlOverride w:ilvl="0">`
	`  …`
	`</w:num>`
	
	In this example, the first numbering level definition (with an ilvl of 0) within the referenced abstract numbering definition will be overridden. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

tentative (Tentative Numbering)	Specifies that a given numbering level was been saved by a producer but was not used in the parent document. This means that this numbering level may be redefined by a future consumer without changing the actual content of the document.
	
	A value of on, 1, or true for this attribute value specifies that the numbering level is not used in the current document's contents.
	
	A value of off, 0, or false for this attribute value specifies that the numbering level is used in the parent document and cannot be redefined without changing its contents. This is the default value for this attribute, and is implied when this attribute is omitted.
	
	[*Example*: Consider the following WordprocessingML numbering level:
	
	`<w:lvl w:ilvl="0" w:tentative="on" >`
	`  ...`
	`</w:lvl>`
	
	This level has the tentative attribute set to on, therefore the contents of this numbering level have not been used in the document and may be redefined by a consumer as desired. *end example*]
	
	If this attribute is equal to on, 1, or true, the WordprocessingML for a given document will contain the numbering level information associated with this numbering level, but the 'tentative' numbering level(s) shall not be represented in any of the hosting application's user interface pertaining to numbering levels.

Attributes	Description
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

tplc (Template Code)	Specifies a unique hexadecimal value which may be used to specify a location within an application's user interface in which this numbering level shall be displayed. The method by which this value is interpreted shall be application-defined.
	
	If this attribute is omitted, then this numbering may be displayed in any location chosen by the consumer.
	
	[*Example*: Consider the following abstract numbering definition:
	
	`<w:abstractNum w:abstractNumId="1" w:tplc="04090019">`
	`…`
	`</w:abstractNum>`
	
	In this example the abstractNum element with attribute abstractNumId equal to 1, would appear in the area within a consumer's application user interface specified by the template code 04090019. *end example*]
	
	The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Lvl">
   <sequence>
      <element name="start" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="numFmt" type="CT_NumFmt" minOccurs="0"/>
      <element name="lvlRestart" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="pStyle" type="CT_String" minOccurs="0"/>
      <element name="isLgl" type="CT_OnOff" minOccurs="0"/>
      <element name="suff" type="CT_LevelSuffix" minOccurs="0"/>
      <element name="lvlText" type="CT_LevelText" minOccurs="0"/>
      <element name="lvlPicBulletId" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="legacy" type="CT_LvlLegacy" minOccurs="0"/>
      <element name="lvlJc" type="CT_Jc" minOccurs="0"/>
      <element name="pPr" type="CT_PPr" minOccurs="0"/>
      <element name="rPr" type="CT_RPr" minOccurs="0"/>
   </sequence>
   <attribute name="ilvl" type="ST_DecimalNumber" use="required"/>
   <attribute name="tplc" type="ST_LongHexNumber" use="optional"/>
   <attribute name="tentative" type="ST_OnOff" use="optional"/>
</complexType>
```

2.9.7 lvl (Numbering Level Definition)

This element specifies the appearance and behavior of a numbering level within a given abstract numbering definition. A numbering level contains a set of properties for the display of the numbering for a given numbering level within an abstract numbering definition.

A numbering level definition is identical to a numbering level override definition, except for the fact that it is defined as part of a numbering definition instance using the abstractNum element (§2.9.1) rather than as part of an abstract numbering definition using the num element (§2.9.16).

[*Example*: Consider the WordprocessingML below:

```
<w:abstractNum w:abstractNumId="4">
  <w:nsid w:val="1DE04504" />
  <w:multiLevelType w:val="hybridMultilevel" />
  <w:lvl w:ilvl="0" w:tplc="0409000F">
    ...
  </w:lvl>
  <w:lvl w:ilvl="1" w:tplc="04090019">
    ...
  </w:lvl>
  <w:lvl w:ilvl="2" w:tplc="04090019">
    ...
  </w:lvl>
  <w:lvl w:ilvl="3" w:tplc="0409000F">
    ...
  </w:lvl>
  ...
</w:abstractNum>
```

This example shows that any paragraph whose numbering properties use the ilvl elements with the attribute val set equal to 0, 1, 2, or 3 will have the appearance and behavior of their first four numbered levels specified by the lvl elements given above (assuming that no level overrides have been specified). *end example*]

Parent Elements
abstractNum (§2.9.1)

Child Elements	Subclause
isLgl (Display All Levels Using Arabic Numerals)	§2.9.4
legacy (Legacy Numbering Level Properties)	§2.9.5
lvlJc (Justification)	§2.9.8
lvlPicBulletId (Picture Numbering Symbol Definition Reference)	§2.9.10

Child Elements	Subclause
lvlRestart (Restart Numbering Level Symbol)	§2.9.11
lvlText (Numbering Level Text)	§2.9.12
numFmt (Numbering Format)	§2.9.18
pPr (Numbering Level Associated Paragraph Properties)	§2.9.24
pStyle (Paragraph Style's Associated Numbering Level)	§2.9.25
rPr (Numbering Symbol Run Properties)	§2.9.26
start (Starting Value)	§2.9.27
suff (Content Between Numbering Symbol and Paragraph Text)	§2.9.30

Attributes	Description

ilvl (Numbering Level)	Specifies the numbering level definition that is to be defined by this set of numbering properties.
	
	This override is a zero-based index of the number of list levels in the document. [*Example*: A value of 2 is the 3rd list level in the document. *end example*]
	
	[*Example*: Consider the following WordprocessingML for a numbering definition instance:
	
	`<w:num w:numId="6">`
	`  <w:abstractNumId w:val="4" />`
	`  <w:lvlOverride w:ilvl="0">`
	`  …`
	`</w:num>`
	
	In this example, the first numbering level definition (with an ilvl of `0`) within the referenced abstract numbering definition will be overridden. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

tentative (Tentative Numbering)	Specifies that a given numbering level was been saved by a producer but was not used in the parent document. This means that this numbering level may be redefined by a future consumer without changing the actual content of the document. A value of on, `1`, or `true` for this attribute value specifies that the numbering level is not used in the current document's contents. A value of `off`, `0`, or `false` for this attribute value specifies that the numbering level is used in the parent document and cannot be redefined without changing its contents. This is the default value for this attribute, and is implied when this attribute is omitted. [*Example*: Consider the following WordprocessingML numbering level:

Attributes	Description
	<w:lvl w:ilvl="0" w:tentative="on" > ... </w:lvl> This level has the tentative attribute set to on, therefore the contents of this numbering level have not been used in the document and may be redefined by a consumer as desired. *end example*] If this attribute is equal to on, 1, or true, the WordprocessingML for a given document will contain the numbering level information associated with this numbering level, but the 'tentative' numbering level(s) shall not be represented in any of the hosting application's user interface pertaining to numbering levels. The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
tplc (Template Code)	Specifies a unique hexadecimal value which may be used to specify a location within an application's user interface in which this numbering level shall be displayed. The method by which this value is interpreted shall be application-defined. If this attribute is omitted, then this numbering may be displayed in any location chosen by the consumer. [*Example*: Consider the following abstract numbering definition: <w:abstractNum w:abstractNumId="1" w:tplc="04090019"> … </w:abstractNum> In this example the abstractNum element with attribute abstractNumId equal to 1, would appear in the area within a consumer's application user interface specified by the template code 04090019.*end example*] The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Lvl">
   <sequence>
      <element name="start" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="numFmt" type="CT_NumFmt" minOccurs="0"/>
      <element name="lvlRestart" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="pStyle" type="CT_String" minOccurs="0"/>
      <element name="isLgl" type="CT_OnOff" minOccurs="0"/>
      <element name="suff" type="CT_LevelSuffix" minOccurs="0"/>
      <element name="lvlText" type="CT_LevelText" minOccurs="0"/>
      <element name="lvlPicBulletId" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="legacy" type="CT_LvlLegacy" minOccurs="0"/>
      <element name="lvlJc" type="CT_Jc" minOccurs="0"/>
      <element name="pPr" type="CT_PPr" minOccurs="0"/>
      <element name="rPr" type="CT_RPr" minOccurs="0"/>
   </sequence>
   <attribute name="ilvl" type="ST_DecimalNumber" use="required"/>
   <attribute name="tplc" type="ST_LongHexNumber" use="optional"/>
   <attribute name="tentative" type="ST_OnOff" use="optional"/>
</complexType>
```

2.9.8 lvlJc (Justification)

This element specifies the type of justification used on a numbering level's text within a given numbering level. This justification is applied relative to the text margin of the parent numbered paragraph in the document.

If omitted, the paragraph shall have left justification relative to the text margin in left-to-right paragraphs, and right justification relative to the text margin in right-to-left paragraphs.

[*Example*: Consider the numbering level defined below:

```
<w:lvl w:ilvl="8" w:tplc="756C1446" w:tentative="1">
  <w:start w:val="1" />
  <w:nfc w:val="23" />
  <w:lvlText w:val="•" />
  <w:lvlJc w:val="left" />
  …
</w:lvl>
```

In this numbering level, the given numbering symbol will be `left` justified with respect to the text margin, therefore the numbering will extend left from the text margin towards the text (assuming a left-to-right paragraph). *end example*]

A numbering level's text is the numeral, symbol, character, graphic, etc. used to create a numbered paragraph as defined by the lvlText element (§2.9.12).

[*Example:* Consider the numbered paragraphs below:

1) Example one

a. Example two

- Example three

The numbering symbol in these three numbered paragraphs are "1", "a", and "•", respectively. *end example*]

Parent Elements
lvl (§2.9.6); lvl (§2.9.7)

Attributes	Description

val (Alignment Type)	Specifies the justification which should be applied to the parent object within a document.
	
	The possible values (see below) for this attribute are always specified relative to the page, and do not change semantic from right-to-left and left-to-right documents.
	
	[*Example*: Consider the following WordprocessingML fragment for a paragraph in a document:
	
	`<w:pPr>`
	`  <w:jc w:val="right" />`
	`</w:pPr>`
	
	This paragraph is now right justified on the page, regardless of the paragraph or section settings. *end example*]
	
	The possible values for this attribute are defined by the ST_Jc simple type (§2.18.50).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Jc">
   <attribute name="val" type="ST_Jc" use="required"/>
</complexType>
```

2.9.9 lvlOverride (Numbering Level Definition Override)

This element specifies an optional override which shall be applied in place of zero or more levels from the abstract numbering definition for a given numbering definition instance. Each instance of this element is used to override the appearance and behavior of a given numbering level definition within the given abstract numbering definition.

[*Example*: Consider a numbering definition instance which inherits its information from the abstract numbering definition with abstractNumId of 4, but wishes to use a different set of properties for level 0 and level 1 of the numbering definition. The resulting WordprocessingML would look like:

```
<w:num w:numId="6">
  <w:abstractNumId w:val="4" />
  <w:lvlOverride w:ilvl="0">
    <w:lvl w:ilvl="0">
      <w:start w:val="4" />
      <w:lvlText w:val="%1)" />
      <w:lvlJc w:val="left" />
      <w:pPr>
        <w:ind w:left="360" w:hanging="360" />
      </w:pPr>
    </w:lvl>
  </w:lvlOverride>
  <w:lvlOverride w:ilvl="1">
    <w:lvl w:ilvl="1">
      <w:start w:val="5" />
      <w:lvlText w:val="%Test)" />
      <w:lvlJc w:val="left" />
      <w:pPr>
        <w:ind w:left="360" w:hanging="360" />
      </w:pPr>
    </w:lvl>
  </w:lvlOverride>
</w:num>
```

end example]

[*Note*: The ability to set level overrides optimizes use of numbering in WordprocessingML as it prevents writing out redundant abstract numbering definitions if numbering sets only slightly differ.

Consider using WordprocessingML to create two numbered sets that only differ only in the appearance and style of the first numbering level. Both could use the same abstract numbering definition as long as each references a different numbering definition instance with one of the numbering definition instances leveraging a level override for the first numbering level. Below is WordprocessingML that illustrates this:

```
<w:num w:numId="5">
  <w:abstractNumId w:val="4" />
</w:num>
```

```
<w:num w:numId="6">
  <w:abstractNumId w:val="4" />
  <w:lvlOverride w:ilvl="0">
    <w:lvl w:ilvl="0">
      <w:start w:val="4" />
      <w:lvlText w:val="%1)" />
      <w:lvlJc w:val="left" />
      <w:pPr>
        <w:ind w:left="360" w:hanging="360" />
      </w:pPr>
    </w:lvl>
  </w:lvlOverride>
</w:num>
```

end note]

Parent Elements
num (§2.9.16)

Child Elements	Subclause
lvl (Numbering Level Override Definition)	§2.9.6
startOverride (Numbering Level Starting Value Override)	§2.9.28

Attributes	Description

ilvl (Numbering Level ID)	Specifies the numbering level of a given abstract numbering definition to be overridden.
	
	If this number conflicts with the ilvl of the child lvl element, then the latter shall be ignored.
	
	[*Example*: Consider a numbering definition instance which inherits its information from the abstract numbering definition with abstractNumId of 4, but wishes to use a different set of properties for level 0 of the numbering definition. The resulting WordprocessingML would look like:
	
	`<w:num w:numId="6">`
	`  <w:abstractNumId w:val="4" />`
	`  <w:lvlOverride w:ilvl="0">`
	`  <w:lvl w:ilvl="0">`
	`  <w:start w:val="4" />`
	`  <w:lvlText w:val="%1)" />`
	`  <w:lvlJc w:val="left" />`
	`  <w:pPr>`
	`  <w:ind w:left="360" />`

Attributes	Description
	</w:pPr> </w:lvl> </w:lvlOverride> </w:num> This level overrides level 0 of the abstract numbering definition's level properties with the specified set of numbering properties, replacing those in the abstract numbering definition. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NumLvl">
   <sequence>
      <element name="startOverride" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="lvl" type="CT_Lvl" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="ilvl" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.9.10 lvlPicBulletId (Picture Numbering Symbol Definition Reference)

This element specifies a picture which shall be used as a numbering symbol for a given numbering level by referring to a picture numbering symbol definition's numPictBullet element (§2.9.21). This reference is made through this element's val attribute.

The picture shall be added to the numbering level by replacing each character in the lvlText with on instance of this image

[*Example*: Consider the WordprocessingML below illustrating how the lvlPicBulletId references a picture numbering symbol definition thought its val attribute:

```
<w:numPicBullet w:numPicBulletId="1">
   <w:pict>
    <v:shape id="_x0000_i1031" type="#_x0000_t75"
style="width:3in;height:276.75pt" o:bullet="t">
      <v:imagedata r:id="rId2" o:title="testpic" />
    </v:shape>
  </w:pict>
</w:numPicBullet>
…
<w:abstractNum w:abstractNumId="7">
  <w:nsid w:val="71A06359" />
  <w:multiLevelType w:val="hybridMultilevel" />
  <w:tmpl w:val="10643FE6" />
```

```
    <w:lvl w:ilvl="0" w:tplc="B7663E56">
      <w:start w:val="1" />
      <w:nfc w:val="23" />
      <w:lvlText w:val="AA" />
      <w:lvlPicBulletId w:val="1" />
    </w:lvl>
  </w:abstractNum>
```

The resulting numbering shall consist of two instances of the image specified using the numPicBullet element. *end example*]

Parent Elements
lvl (§2.9.6); lvl (§2.9.7)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.9.11 lvlRestart (Restart Numbering Level Symbol)

This element specifies a one-based index which determines when a numbering level should restart to its start value (§2.9.27). A numbering level restarts when an instance of the specified numbering level, which shall be higher (earlier than the this level) is used in the given document's contents.

If this element is omitted, the numbering level shall restart each time the previous numbering level is used. If the specified level is higher than the current level, then this element shall be ignored. As well, a value of 0 shall specify that this level shall never restart.

[*Example*: Consider a set of numbered paragraphs in a WordprocessingML document where numbering level with ilvl of 2 is set to never restart:

```
<w:lvl w:ilvl="0">
  <w:start w:val="1" />
  <w:lvlText w:val="%1)" />
  <w:lvlJc w:val="left" />
   <w:pPr>
    <w:ind w:left="360" w:hanging="360" />
  </w:pPr>
   <w:rPr>
    <w:rFonts w:hint="default" />
  </w:rPr>
</w:lvl>
 <w:lvl w:ilvl="1">
  <w:start w:val="1" />
  <w:numFmt w:val="upperLetter" />
  <w:lvlText w:val="%2)" />
  <w:lvlJc w:val="left" />
   <w:pPr>
    <w:ind w:left="720" w:hanging="360" />
  </w:pPr>
   <w:rPr>
    <w:rFonts w:hint="default" />
  </w:rPr>
</w:lvl>
<w:lvl w:ilvl="2">
  <w:start w:val="1" />
  <w:numFmt w:val="lowerRoman" />
  <w:lvlRestart w:val="0">
  <w:lvlText w:val="%3)" />
  <w:lvlJc w:val="left" />
   <w:pPr>
    <w:ind w:left="1080" w:hanging="360" />
  </w:pPr>
   <w:rPr>
    <w:rFonts w:hint="default" />
  </w:rPr>
</w:lvl>
```

Since the lvlRestart element is omitted in numbering level 1 (a,b,…), the numbering level restarts after numbering level 0 (1,2,…) is used. Numbering level two (i, ii, iii …) never restarts as lvlRestart has a val equal to 0. An example of the resulting content would be as follows:

1) Level one

    a) Level two

        i) Level three

        ii) Level three

2) Level one

    a) Level two

        iii) Level three

        iv) Level four

The resulting set of paragraphs has level two restarting to its start value after each level one (after `2)`, the next level two is again `a)`), but level three never restarts and continues at `iii)` even after the use of a level two and one. *end example*]

Parent Elements
lvl (§2.9.6); lvl (§2.9.7)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.9.12 lvlText (Numbering Level Text)

This element specifies the textual content which shall be displayed when displaying a paragraph with the given numbering level.

All text in this element's val attribute shall be taken as literal text to be repeated in each instance of this numbering level, except for any use of the percent symbol (%) followed by a number, which shall be used to indicate the one-based index of the number to be used at this level. Any number of a level higher than this level shall be ignored.

When the % syntax is used, the number shall be incremented for each subsequent paragraph of that level (sequential or not), until the restart level is seen between two subsequent paragraphs of this level.

[*Example*: Consider the following WordprocessingML for a numbering level:

```
<w:lvl w:ilvl="1">
  …
  <w:lvlText w:val="StringA %2 StringB %1 StringC %3"/>
  …
</w:lvl>
```

This specifies that three strings (`StringA`, `StringB`, `StringC`) shall be used as string literals in the numbering for level two (ilvl of 1) along with the numbering symbol used for level one and level zero. Although level two is also referenced here, it is ignored as it is a higher level than the current numbering level.

Therefore, assuming the numbering symbol used by numbering level zero is an Arabic numeral, and the numbering symbol used by numbering level one is a Roman numeral, a set of numbered paragraphs using this WordprocessingML numbering set shall be output as:

```
1
  StringA I StringB 1 StringC
  StringA II StringB 1 StringC
  StringA III StringB 1 StringC
2
  StringA I StringB 2 StringC
  StringA II StringB 2 StringC
```

with the %1 and %2 values corresponding to the current numbering symbol value for numbering level zero and one, respectively. *end example*]

Parent Elements
lvl (§2.9.6); lvl (§2.9.7)

Attributes	Description

Attributes	Description

null (Level Text Is Null Character)	Specifies that a null character shall be used as the numbering symbol for a given numbering level.
	
	If the val attribute contains any content, then this attribute shall be ignored.
	
	If this attribute is omitted, then the null string shall not be used in place of the empty string. [*Note*: A null character is different from an empty string. *end note*]
	
	[*Example*: Consider the WordprocessingML below:
	
	`<w:lvl w:ilvl="1">`
	`…`
	`<w:lvlText w:null="on" />`
	`…`
	`</w:lvl>`
	
	This level text consists of a single null character, and not the empty string, as the null attribute is set. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

val (Level Text)	Specifies the actual text to be used for the numbering level when it is referenced in the document's content.
	
	If this attribute is not specified, then the empty string shall be used as the level's text.
	
	[*Example*: Consider the WordprocessingML below:
	
	`<w:lvl w:ilvl="1">`
	`…`
	`<w:lvlText w:val="test" />`
	`…`
	`</w:lvl>`
	
	Here the val attribute specifies that the literal string `test` is to be surfaced as the text for the given numbering level, regardless of its position. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LevelText">
   <attribute name="val" type="ST_String" use="optional"/>
   <attribute name="null" type="ST_OnOff" use="optional"/>
</complexType>
```

2.9.13 multiLevelType (Abstract Numbering Definition Type)

This element specifies the type of numbering defined by a given abstract numbering type. This information shall only be used by a consumer to determine user interface behaviors for this numbering definition, and shall not be used to limit the behavior of the list (i.e. a list with multiple levels marked as `singleLevel` shall not be prevented from using levels 2 through 9).

If this element is omitted, then the list shall be assumed to be of any type desired by the consumer.

[*Example*: Consider the WordprocessingML below:

```
<w:abstractNum w:abstractNumId="8">
  …
  <w:multiLevelType w:val="singleLevel" />
  …
</w:abstractNum>
```

This abstract numbering definition is specified to be of the `singleLevel` type by the multiLevelType element. *end example*]

Parent Elements
abstractNum (§2.9.1)

Attributes	Description

val (Abstract Numbering Definition Type)	Specifies the specific type of numbering enabled by a given abstract numbering definition.
	
	[*Example*: Consider the WordprocessingML below:
	
	`<w:abstractNum w:abstractNumId="8">`
	`  …`
	`  <w:multiLevelType w:val="multiLevel" />`
	`  …`
	`</w:abstractNum>`
	
	This abstract numbering definition is specified to be of the `multiLevel` type, which may be used by consumers to place this numbering correctly within a user interface. *end example*]
	
	The possible values for this attribute are defined by the ST_MultiLevelType simple type (§2.18.65).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MultiLevelType">
   <attribute name="val" type="ST_MultiLevelType" use="required"/>
</complexType>
```

2.9.14 name (Abstract Numbering Definition Name)

This element specifies the name of a given abstract numbering definition. This name may be surfaced in order to provide a user friendly alias for a given numbering definition, but shall not influence the behavior of the list - two identical definitions with different name elements shall behave identically.

If this element is omitted, then this abstract numbering definition shall have no name.

[*Example*: Consider the WordprocessingML below:

```
<w:abstractNum w:abstractNumId="4">
  <w:nsid w:val="5C294B5B" />
  <w:multiLevelType w:val="multilevel" />
  <w:tmpl w:val="6F8A81B0" />
  <w:name w:val="Example Name" />
  …
</w:abstractNum>
```

In this example, the given abstract numbering definition is named `Example Name` by use of the name element. *end example*]

Parent Elements
abstractNum (§2.9.1)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent

Attributes	Description
	structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.9.15 nsid (Abstract Numbering Definition Identifier)

This element associates a unique hexadecimal ID to the parent abstract numbering definition. This number shall be identical for two abstract numbering definitions that are based from the same initial numbering definition - if a document is repurposed and the underlying numbering definition is changed, it shall maintain its original nsid.

If this element is omitted, then the list shall have no nsid and one may be added by a producer arbitrarily.

[*Note*: This element may be used to determine the abstract numbering definition to be applied to a numbered paragraph copied from one document and pasted into another. Consider a case in which a given numbered paragraph associated with a abstract numbering definition with nsid FFFFFF23, is pasted among numbered paragraphs associated with a completely different appearance and an abstract numbering definition with an nsid of FFFFFF23. Here, because of the distinction enabled by the identical nsid values, the hosting application would not have to arbitrarily keep the pasted numbered paragraph associated with its original abstract numbering definition, as it may use the information provided by the abstract numbering definition's identical nsid values to know that those two numbering sets are identical, and merge the paragraphs into the target numbering format. *end note*]

[*Example*: Consider the WordprocessingML for an abstract numbering definition below:

```
<w:abstractNum w:abstractNumId="3">
  <w:nsid w:val="FFFFFF89" />
  <w:multiLevelType w:val="singleLevel" />
  <w:tmpl w:val="D9842532" />
  …
</w:abstractNum>
```

In this example, the given abstract numbering definition is associated with the unique hexadecimal ID FFFFFF89. *end example*]

Parent Elements
abstractNum (§2.9.1)

Attributes	Description
val (Long Hexadecimal Number Value)	Specifies a number value specified as a four digit hexadecimal number), whose contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following value for an attribute of type ST_LongHexNumber: `00BE2C6C`. This value is valid, as it contains four hexadecimal digits, each an encoding of an octet of the actual decimal number value. It may therefore be interpreted as desired in the context of the parent XML element, *end example*] The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LongHexNumber">
   <attribute name="val" type="ST_LongHexNumber" use="required"/>
</complexType>
```

2.9.16 num (Numbering Definition Instance)

This element specifies a unique instance of numbering information that can be referenced by zero or more paragraphs within the parent WordprocessingML document.

This instance requires the referencing of a base abstract numbering definition through the abstractNumId child element (§2.9.2). This element also can be used to specify a set of optional overrides applied to zero or more levels from the abstract numbering definition inherited by this instance second though the optional lvlOverride child elements (§2.9.9).

[*Example*: Consider the WordprocessingML for a document with four numbering definition instances, two of which reference the same underlying abstract numbering definition:

```
<w:numbering>
  ...
  <w:num w:numId="2">
    <w:abstractNumId w:val="0" />
  </w:num>
  <w:num w:numId="3">
    <w:abstractNumId w:val="1" />
  </w:num>
  <w:num w:numId="4">
    <w:abstractNumId w:val="4" />
  </w:num>
  <w:num w:numId="5">
    <w:abstractNumId w:val="4" />
  </w:num>
</w:numbering>
```

As shown above, the first two numbering definition instances reference abstractNumId values of 0 and 1 respectively, and the last two both reference the abstract numbering definition with an abstractNumId of 4. *end example*]

[*Example*: Consider a numbering definition instance which inherits its information from the abstract numbering definition with abstractNumId of 4, but wishes to use a different set of properties for level 0 of the numbering definition. The resulting WordprocessingML would look like:

```
<w:num w:numId="6">
  <w:abstractNumId w:val="4" />
  <w:lvlOverride w:ilvl="0">
    <w:lvl w:ilvl="0">
      <w:start w:val="4" />
      <w:lvlText w:val="%1)" />
      <w:lvlJc w:val="left" />
      <w:pPr>
        <w:ind w:left="360" w:hanging="360" />
      </w:pPr>
    </w:lvl>
  </w:lvlOverride>
</w:num>
```

The lvlOverride element specifies an override for level 0 of the abstract numbering definition. *end example*]

Parent Elements
numbering (§2.9.17)

Child Elements	Subclause
abstractNumId (Abstract Numbering Definition Reference)	§2.9.2
lvlOverride (Numbering Level Definition Override)	§2.9.9

Attributes	Description

numId (Numbering Definition Instance ID)	Specifies a unique ID which any numbered paragraph which wishes to inherit these numbering properties shall reference using the numPr element (§2.3.1.19).
	
	[*Example*: Consider the WordprocessingML below for an example numbered paragraph:
	
	`<w:p>`
	`  <w:pPr>`
	`  <w:numPr>`
	`  <w:ilvl w:val="0" />`
	`  <w:numId w:val="5" />`
	`  </w:numPr>`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	
	This paragraph references a numbering definition instance with a numId attribute of 5:
	
	`<w:num w:numId="5">`
	`  <w:abstractNumId w:val="4" />`
	`</w:num>`
	
	The numbering definition instance with a numId attribute of 5 correlates with the numbered paragraph with the numbering definition instance referent element with a val of 5, so the numbered paragraph inherits its properties. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Num">
   <sequence>
      <element name="abstractNumId" type="CT_DecimalNumber" minOccurs="1"/>
      <element name="lvlOverride" type="CT_NumLvl" minOccurs="0" maxOccurs="9"/>
   </sequence>
   <attribute name="numId" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.9.17 numbering (Numbering Definitions)

This element specifies the formatting, display, and functionality of numbering - Arabic numerals, Roman numerals, symbol characters ("bullets"), text strings, etc. - in WordprocessingML documents, which are used to label individual paragraphs of text.

[*Example*: The following two paragraphs each contain numbering as defined by WordprocessingML: the first uses an Arabic numeral, the second a symbol character:

10. This is a paragraph with numbering information.

- This is also a paragraph with numbering information.

end example]

Parent Elements
Root element of WordprocessingML Numbering Definitions part

Child Elements	Subclause
abstractNum (Abstract Numbering Definition)	§2.9.1
num (Numbering Definition Instance)	§2.9.16
numIdMacAtCleanup (Last Reviewed Abstract Numbering Definition)	§2.9.20
numPicBullet (Picture Numbering Symbol Definition)	§2.9.21

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Numbering">
   <sequence>
      <element name="numPicBullet" type="CT_NumPicBullet" minOccurs="0" maxOccurs="unbounded"/>
      <element name="abstractNum" type="CT_AbstractNum" minOccurs="0" maxOccurs="unbounded"/>
      <element name="num" type="CT_Num" minOccurs="0" maxOccurs="unbounded"/>
      <element name="numIdMacAtCleanup" type="CT_DecimalNumber" minOccurs="0"/>
   </sequence>
</complexType>
```

2.9.18 numFmt (Numbering Format)

This element specifies the number format which shall be used to display all numbering at this level in the numbering definition. This information is used to replace the level text string %x, where x is a particular one-based level index, with the appropriate value. This value shall be calculated by counting the number of paragraphs at this level since the last restart using the numbering system defined in the val attribute.

If omitted, the level shall be assumed to be of type `decimal`.

[*Example*: Consider the following WordprocessingML fragment for a numbering level in a numbering definition:

```
<w:lvl w:ilvl="2">
```

```
    <w:start w:val="1" />
    <w:numFmt w:val="lowerRoman" />
    <w:lvlRestart w:val="0">
    <w:lvlText w:val="%3)" />
    <w:lvlJc w:val="left" />
    <w:pPr>
      <w:ind w:left="1080" w:hanging="360" />
    </w:pPr>
    <w:rPr>
      <w:rFonts w:hint="default" />
    </w:rPr>
  </w:lvl>
```

A numFmt value of `lowerLetter` indicates that a consumer shall use lowercase letters for all numbering of this level: a,b,c… *end example*]

Parent Elements
lvl (§2.9.6); lvl (§2.9.7)

Attributes	Description
val (Numbering Format Type)	Specifies the number format that shall be used for all numbering in the parent object. [*Example*: A value of `lowerLetter` indicates that a consumer shall use lowercase letters for each number in this grouping: a,b,c… *end example*] The possible values for this attribute are defined by the ST_NumberFormat simple type (§2.18.66).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NumFmt">
   <attribute name="val" type="ST_NumberFormat" use="required"/>
</complexType>
```

2.9.19 numId (Numbering Definition Instance Reference)

This element specifies the numbering definition instance which shall be used for the given parent numbered paragraph in the WordprocessingML document.

A value of `0` for the val attribute shall never be used to point to a numbering definition instance, and shall instead only be used to designate the removal of numbering properties at a particular level in the style hierarchy (typically via direct formatting).

[*Example*: Consider the WordprocessingML below for an example numbered paragraph:

```
<w:p>
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="0" />
      <w:numId w:val="5" />
    </w:numPr>
  </w:pPr>
  …
</w:p>
```

This paragraph references a numbering definition instance with a numId attribute of 5, as follows:

```
<w:num w:numId="5">
  <w:abstractNumId w:val="4" />
</w:num>
```

The numbering definition instance reference specifies the given numbering definition instance to be applied to the given paragraph, which itself inherits its properties from abstract numbering definition with abstractNumId of 4. *end example*]

Parent Elements
numPr (§2.3.1.19)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.9.20 numIdMacAtCleanup (Last Reviewed Abstract Numbering Definition)

This element specifies to a consumer the progress in the last attempt made by the application to remove unused abstract numbering definitions from a given document. If a legacy document is opened by a consumer, it may choose to remove abstract numbering definition which are 'orphaned' (have no associated numbering definition instances). This element is used by those consumers to indicate their progress (if not complete) in reviewing existing abstract numbering definitions. [*Note*: Removing unused abstract numbering definition from a document will reduce the file size, but is not required. *end note*]

If omitted, then all abstract numbering definitions shall be considered reviewed.

[*Example*: Consider a document with 32 abstract numbering definitions, with abstractNumId values ranging from 0 to 85. If an application has only reviewed those abstract numbering definitions with abstractNumId values lower than 25 at save time, it would indicate that state as follows:

```
<w:numIdMacAtCleanup w:val="25"/>
```

This value specifies that all abstract numbering definitions with an abstractNumId value higher than 25 have not yet been reviewed. *end example*]

Parent Elements
numbering (§2.9.17)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
    <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.9.21 numPicBullet (Picture Numbering Symbol Definition)

This element specifies the appearance and behavior of a specific picture to be used as the numbering symbol within a numbering level definition in a document, and is the basis for all picture numbering symbol information in a WordprocessingML document.

This element is not used directly within abstract numbering definitions but rather is referenced through its numPicBulletId attribute by the lvlPicBulletId element (§2.9.10) used within numbering level definitions.

[*Example*: Consider the WordprocessingML fragment below which illustrates how a numPicBullet definition is referenced by a picture numbering symbol definition reference through its numPicBulletId attribute:

```
<w:numPicBullet w:numPicBulletId="1">
   <w:pict>
     <v:shape id="_x0000_i1031" type="#_x0000_t75"
style="width:3in;height:276.75pt" o:bullet="t">
      <v:imagedata r:id="rId2" o:title="testpic" />
    </v:shape>
  </w:pict>
</w:numPicBullet>
...
<w:abstractNum w:abstractNumId="7">
  <w:nsid w:val="71A06359" />
  <w:multiLevelType w:val="hybridMultilevel" />
  <w:tmpl w:val="10643FE6" />
  <w:lvl w:ilvl="0" w:tplc="B7663E56">
    <w:start w:val="1" />
    <w:nfc w:val="23" />
    <w:lvlText w:val="" />
    <w:lvlPicBulletId w:val="1" />
  </w:lvl>
</w:abstractNum>
```

The lvlPicBulletId element references a numPicBullet element, which defines the size and appearance of all picture bullets of this type the document. It is important to note that this picture bullet may be referenced by multiple levels of various numbering definitions. *end example*]

Parent Elements
numbering (§2.9.17)

Child Elements	Subclause
pict (Picture Numbering Symbol Properties)	§2.9.23

Attributes	Description

numPicBulletId (Picture Numbering Symbol ID)	Specifies a unique ID for this picture bullet definition which shall be used to reference this picture bullet from a numbering level definition.
	
	[*Example*: Consider the WordprocessingML fragment below which illustrates how a numPicBullet definition is referenced by a picture numbering symbol definition reference through its numPicBulletId attribute:
	
	`<w:numPicBullet w:numPicBulletId="1">`
	`  …`
	`</w:numPicBullet>`
	`...`
	`<w:abstractNum w:abstractNumId="7">`
	`  <w:lvl w:ilvl="0" w:tplc="B7663E56">`
	`  …`
	`  <w:lvlPicBulletId w:val="1" />`
	`  </w:lvl>`
	`</w:abstractNum>`
	
	The lvlPicBulletId element references the ID in the numPicBulletId attribute directly. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NumPicBullet">
   <sequence>
      <element name="pict" type="CT_Picture"/>
   </sequence>
   <attribute name="numPicBulletId" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.9.22 numStyleLink (Numbering Style Reference)

This element specifies an abstract numbering does not contain the actual numbering properties for its type, but rather serves as a reference to a numbering style stored in the document, which shall be applied when this abstract numbering definition is referenced, and itself points at the actual underlying abstract numbering definition to be used.

The numbering style that is to be applied when this abstract numbering definition is referenced is identified by the string contained in numStyleLink's val attribute.

[*Example*: Consider the abstract numbering definition below:

```
<w:abstractNum w:abstractNumId="0">
  <w:nsid w:val="38901FA4" />
  <w:multiLevelType w:val="multilevel" />
  <w:numStyleLink w:val="TestNumberingStyle" />
</w:abstractNum>
```

This abstract numbering definition references the numbering style with a styleId attribute equal to `TestNumberingStyle`, as follows below:

```
<w:style w:type="numbering" w:styleId="TestNumberingStyle">
  …
</w:style>
```

Therefore, this numbering style shall be applied whenever the base abstract numbering definition is inherited by a numbered paragraph. *end example*]

Parent Elements
abstractNum (§2.9.1)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.9.23 pict (Picture Numbering Symbol Properties)

This element specifies the properties for a picture which shall be used as a picture numbering symbol in a given document, using the VML syntax.

[*Example*: Consider the WordprocessingML below illustrating the usage of the pict element in a document containing a single picture numbering symbol:

```
<w:numPicBullet w:numPicBulletId="0">
  <w:pict>
    <v:shapetype id="_x0000_t75" coordsize="21600,21600" o:spt="75"
o:preferrelative="t" path="m@4@5l@4@11@9@11@9@5xe" filled="f" stroked="f">
      <v:stroke joinstyle="miter" />
      <v:formulas>
        <v:f eqn="if lineDrawn pixelLineWidth 0" />
        <v:f eqn="sum @0 1 0" />
        <v:f eqn="sum 0 0 @1" />
        <v:f eqn="prod @2 1 2" />
        <v:f eqn="prod @3 21600 pixelWidth" />
        <v:f eqn="prod @3 21600 pixelHeight" />
        <v:f eqn="sum @0 0 1" />
        <v:f eqn="prod @6 1 2" />
        <v:f eqn="prod @7 21600 pixelWidth" />
        <v:f eqn="sum @8 21600 0" />
        <v:f eqn="prod @7 21600 pixelHeight" />
        <v:f eqn="sum @10 21600 0" />
      </v:formulas>
      <v:path o:extrusionok="f" gradientshapeok="t" o:connecttype="rect" />
      <o:lock v:ext="edit" aspectratio="t" />
    </v:shapetype>
    <v:shape id="_x0000_i1029" type="#_x0000_t75"
style="width:11.25pt;height:11.25pt" o:bullet="t">
      <v:imagedata r:id="rId1" o:title="sample picture" />
    </v:shape>
  </w:pict>
```

end example]

Parent Elements
numPicBullet (§2.9.21)

Child Elements	Subclause
Any element from the urn:schemas-microsoft-com:vml namespace	§6.1
Any element from the urn:schemas-microsoft-com:office:office namespace	§6.2
control (Floating Embedded Control)	§2.3.3.2
movie (Embedded Video)	§2.3.3.17

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Picture">
   <complexContent>
      <extension base="CT_PictureBase">
         <sequence maxOccurs="1">
            <element name="movie" type="CT_Rel" minOccurs="0"/>
            <element name="control" type="CT_Control" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.9.24 pPr (Numbering Level Associated Paragraph Properties)

This element specifies the paragraph properties which shall be applied as part of a given numbering level within the parent numbering definition. These paragraph properties are applied to any numbered paragraph that references the given numbering definition and numbering level.

Paragraph properties specified on the numbered paragraph itself override the paragraph properties specified by pPr elements within a numbering lvl element (§2.9.6, §2.9.7).

[*Example:* Consider the WordprocessingML below which specifies numbering level paragraph properties:

```
<w:abstractNum w:abstractNumId="1">
  ...
  <w:lvl w:ilvl="0">
    ...
    <w:pPr>
      <w:tabs>
        <w:tab w:val="num" w:pos="720" />
      </w:tabs>
      <w:ind w:left="720" w:hanging="360" />
    </w:pPr>
  </w:lvl>
</w:abstractNum>
```

Each of the paragraph properties specified inside the pPr element are applied to any numbered paragraph which inherits this numbering level definition as part of the numbering properties in the order defined by the style hierarchy. *end example*]

Parent Elements
lvl (§2.9.6); lvl (§2.9.7)

Child Elements	Subclause
adjustRightInd (Automatically Adjust Right Indent When Using Document Grid)	§2.3.1.1
autoSpaceDE (Automatically Adjust Spacing of Latin and East Asian Text)	§2.3.1.2
autoSpaceDN (Automatically Adjust Spacing of East Asian Text and Numbers)	§2.3.1.3
bidi (Right to Left Paragraph Layout)	§2.3.1.6
cnfStyle (Paragraph Conditional Formatting)	§2.3.1.8
contextualSpacing (Ignore Spacing Above and Below When Using Identical Styles)	§2.3.1.9
divId (Associated HTML div ID)	§2.3.1.10
framePr (Text Frame Properties)	§2.3.1.11
ind (Paragraph Indentation)	§2.3.1.12
jc (Paragraph Alignment)	§2.3.1.13
keepLines (Keep All Lines On One Page)	§2.3.1.14
keepNext (Keep Paragraph With Next Paragraph)	§2.3.1.15
kinsoku (Use East Asian Typography Rules for First and Last Character per Line)	§2.3.1.16
mirrorIndents (Use Left/Right Indents as Inside/Outside Indents)	§2.3.1.18
numPr (Numbering Definition Instance Reference)	§2.3.1.19
outlineLvl (Associated Outline Level)	§2.3.1.20
overflowPunct (Allow Punctuation to Extent Past Text Extents)	§2.3.1.21
pageBreakBefore (Start Paragraph on Next Page)	§2.3.1.23
pBdr (Paragraph Borders)	§2.3.1.24
pPrChange (Revision Information for Paragraph Properties)	§2.13.5.31
pStyle (Referenced Paragraph Style)	§2.3.1.27
rPr (Run Properties for the Paragraph Mark)	§2.3.1.29
sectPr (Section Properties)	§2.6.19
shd (Paragraph Shading)	§2.3.1.31
snapToGrid (Use Document Grid Settings for Inter-Line Paragraph Spacing)	§2.3.1.32
spacing (Spacing Between Lines and Above/Below Paragraph)	§2.3.1.33
suppressAutoHyphens (Suppress Hyphenation for Paragraph)	§2.3.1.34
suppressLineNumbers (Suppress Line Numbers for Paragraph)	§2.3.1.35

Child Elements	Subclause
suppressOverlap (Prevent Text Frames From Overlapping)	§2.3.1.36
tabs (Set of Custom Tab Stops)	§2.3.1.38
textAlignment (Vertical Character Alignment on Line)	§2.3.1.39
textboxTightWrap (Allow Surrounding Paragraphs to Tight Wrap to Text Box Contents)	§2.3.1.40
textDirection (Paragraph Text Flow Direction)	§2.3.1.41
topLinePunct (Compress Punctuation at Start of a Line)	§2.3.1.43
widowControl (Allow First/Last Line to Display on a Separate Page)	§2.3.1.44
wordWrap (Allow Line Breaking At Character Level)	§2.3.1.45

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PPr">
   <complexContent>
      <extension base="CT_PPrBase">
         <sequence>
            <element name="rPr" type="CT_ParaRPr" minOccurs="0"/>
            <element name="sectPr" type="CT_SectPr" minOccurs="0"/>
            <element name="pPrChange" type="CT_PPrChange" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.9.25 pStyle (Paragraph Style's Associated Numbering Level)

This element specifies the name of a paragraph style which shall automatically this numbering level when applied to the contents of the document. When a paragraph style is defined to include a numbering definition, any numbering level defined by the numPr element (§2.3.1.19) shall be ignored, and instead this element shall specify the numbering level associated with that paragraph style.

If this element references a style which does not exist, or is not a paragraph style, then it may be ignored.

[*Example*: Consider the WordprocessingML below which specifies that the paragraph style with styleId `example`, when applied to paragraphs in the document, shall also apply the first numbering level of the abstract numbering definition with an abstractNumId equal to `1`, as follows:

```
<w:abstractNum w:abstractNumId="1">
  ...
  <w:lvl w:ilvl="0">
    ...
    <w:pStyle w:val="example" />
    <w:pPr>
      <w:tabs>
        <w:tab w:val="num" w:pos="720" />
      </w:tabs>
```

```
        <w:ind w:left="720" w:hanging="360" />
      </w:pPr>
      ...
    </w:lvl>
  </w:abstractNum>
```

The style definition for the paragraph style would only include the numId of the numbering definition instance, and not its level:

```
  <w:style w:styleId="example" w:type="paragraph">
    …
    <w:pPr>
      <w:numPr>
        <w:numId w:val="0" />
      </w:numPr>
    </w:pPr>
  </w:style>
```

end example]

Parent Elements
lvl (§2.9.6); lvl (§2.9.7)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent

Attributes	Description
	structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.9.26 rPr (Numbering Symbol Run Properties)

This element specifies the run properties which shall be applied to the numbering level's text specified in the lvlText element (§2.9.12) when it is applied to paragraphs in this document.

These run properties are applied to all numbering level text used by a given abstract numbering definition and numbering level. It should be noted that run properties specified on a numbered paragraph itself, or on text runs within a numbered paragraph, are separate from the run properties specified by rPr elements within a numbering level, as the latter affects only the numbering text itself, not the remainder of runs in the numbered paragraph.

[*Example*: Consider the WordprocessingML below which uses the rPr element to specify that the numbering symbol used within a given numbering level should be bold and of a 16 point font size:

```
<w:lvl w:ilvl="1">
  …
  <w:rPr>
    <w:b />
    <w:sz w:val="32" />
  </w:rPr>
</w:lvl>
```

The resulting paragraph will use its regular paragraph formatting, but the numbering level text itself shall be specifically formatted as bold in 16 point font. *end example*]

Parent Elements
lvl (§2.9.6); lvl (§2.9.7)

Child Elements	Subclause
b (Bold)	§2.3.2.1
bCs (Complex Script Bold)	§2.3.2.2
bdr (Text Border)	§2.3.2.3

Child Elements	Subclause
caps (Display All Characters As Capital Letters)	§2.3.2.4
color (Run Content Color)	§2.3.2.5
cs (Use Complex Script Formatting on Run)	§2.3.2.6
dstrike (Double Strikethrough)	§2.3.2.7
eastAsianLayout (East Asian Typography Settings)	§2.3.2.8
effect (Animated Text Effect)	§2.3.2.9
em (Emphasis Mark)	§2.3.2.10
emboss (Embossing)	§2.3.2.11
fitText (Manual Run Width)	§2.3.2.12
highlight (Text Highlighting)	§2.3.2.13
i (Italics)	§2.3.2.14
iCs (Complex Script Italics)	§2.3.2.15
imprint (Imprinting)	§2.3.2.16
kern (Font Kerning)	§2.3.2.17
lang (Languages for Run Content)	§2.3.2.18
noProof (Do Not Check Spelling or Grammar)	§2.3.2.19
oMath (Office Open XML Math)	§2.3.2.20
outline (Display Character Outline)	§2.3.2.21
position (Vertically Raised or Lowered Text)	§2.3.2.22
rFonts (Run Fonts)	§2.3.2.24
rPrChange (Revision Information for Run Properties)	§2.13.5.32
rStyle (Referenced Character Style)	§2.3.2.27
rtl (Right To Left Text)	§2.3.2.28
shadow (Shadow)	§2.3.2.29
shd (Run Shading)	§2.3.2.30
smallCaps (Small Caps)	§2.3.2.31
snapToGrid (Use Document Grid Settings For Inter-Character Spacing)	§2.3.2.32
spacing (Character Spacing Adjustment)	§2.3.2.33
specVanish (Paragraph Mark Is Always Hidden)	§2.3.2.34
strike (Single Strikethrough)	§2.3.2.35
sz (Font Size)	§2.3.2.36
szCs (Complex Script Font Size)	§2.3.2.37
u (Underline)	§2.3.2.38
vanish (Hidden Text)	§2.3.2.39

Child Elements	Subclause
vertAlign (Subscript/Superscript Text)	§2.3.2.40
w (Expanded/Compressed Text)	§2.3.2.41
webHidden (Web Hidden Text)	§2.3.2.42

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RPr">
   <sequence>
      <group ref="EG_RPrContent" minOccurs="0"/>
   </sequence>
</complexType>
```

2.9.27 start (Starting Value)

This element specifies the starting value for the numbering used by the parent numbering level within a given numbering level definition. This value is used when this level initially starts in a document, as well as whenever it is restarted via the properties set in the lvlRestart element (§2.9.11).

If this element is omitted, then the starting value shall be zero (0).

[*Example*: Consider the following WordprocessingML fragment for an abstract numbering definition:

```
<w:abstractNum w:abstractNumId="1">
  …
  <w:lvl w:ilvl="0">
    <w:start w:val="2" />
    <w:numFmt w:val="upperLetter"/>
    …
  </w:lvl>
</w:abstractNum>
```

In this example, since upper case Western letters (upperLetter) are being used as numbering symbols for this numbering level, the first instance of a numbering paragraph associated with this abstract numbering definition and numbering level would have the numbering symbol B, the second letter in the number format.

Subsequent numbered paragraphs with this abstract numbering definition and at this level would have their numbering symbols incremented from B (the starting value for this numbering level). *end example*]

Parent Elements
lvl (§2.9.6); lvl (§2.9.7)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number.

Attributes	Description
	The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.9.28 startOverride (Numbering Level Starting Value Override)

This element specifies the number which the specified level override shall begin with. This value is used when this level initially starts in a document, as well as whenever it is restarted via the properties set in the lvlRestart element (§2.9.11).

If they disagree, this value shall override the starting number of the child lvl element (§2.9.6).

[*Example*: Consider the WordprocessingML fragment for a numbering level definition in a WordprocessingML document:

```
<w:abstractNum w:abstractNumId="1">
  …
  <w:lvl w:ilvl="0">
    …
    <w:numFmt w:val="upperRoman" />
    <w:start w:val="2" />
    …
  </w:lvl>
</w:abstractNum>
```

In this example, since upper case Roman numerals (upperRoman) are being used as numbering symbols for this numbering level, the first instance of a numbering paragraph associated with this abstract numbering definition and numbering level would have the numbering symbol II, the second letter in the number format.

Subsequent numbered paragraphs with this abstract numbering definition and at this level would have their numbering symbols incremented from II (the starting value for this numbering level). *end example*]

Parent Elements
lvlOverride (§2.9.9)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.9.29 styleLink (Numbering Style Definition)

This element specifies that the parent abstract numbering definition is the base numbering definition for the specified numbering style referenced in its val attribute.

If this element is omitted, or it references a style which does not exist, then this numbering definition shall not be the underlying properties for a numbering style.

[*Note*: Numbering styles are never directly referenced by paragraphs or runs in the document – instead, an abstract numbering definition specifies that it contains the underlying numbering information for a numbering style, and one or more numbering definition instances reference a numbering definition which inherits from it. The numbering style itself is just a friendly name on an abstract numbering definition. *end note*]

[E*xample*: Consider the WordprocessingML fragment below, representing an abstract numbering definition which defines the properties for a numbering style:

```
<w:numbering>
  …
  <w:abstractNum w:abstractNumId="5">
    …
    <w:styleLink w:val="ExampleNumberingStyle" />
```

```
    …
  </w:abstractNum>
</w:numbering>
…
<w:styles>
  …
  <w:style w:type="numbering" w:styleId="ExampleNumberingStyle">
    <w:name w:val="ExampleNumberingStyle" />
    …
    <w:pPr>
      <w:numPr>
        <w:numId w:val="6" />
      </w:numPr>
    </w:pPr>
  </w:style>
  …
</w:styles>
```

The styleLink element specifies that the abstract numbering definition defines the properties for a numbering style whose styleId matches its val attribute, and is defined in the styles element of the WordprocessingML. *end example*]

Parent Elements
abstractNum (§2.9.1)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`

Attributes	Description
	</w:sdtPr> In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.9.30 suff (Content Between Numbering Symbol and Paragraph Text)

This element specifies the content which shall be added between a given numbering level's text and the text of every numbered paragraph which references that numbering level.

If this element is omitted, then its value shall be assumed to be `tab`.

[*Example:* Consider the numbered paragraph below:

1. Test

In this example, a space exists between the numbering symbol `1.` and the numbered paragraph text `Test`. The space would be specified in WordprocessingML as follows:

```
<w:lvl w:ilvl="0">
  …
  <w:suff w:val="space" />
  …
</w:lvl>
```

The suff element with an attribute value of `space` specifies that the character between the numbering's level text and the paragraph text shall be a space. *end example*]

Parent Elements
lvl (§2.9.6); lvl (§2.9.7)

Attributes	Description
val (Character Type Between Numbering and Text)	Specifies the character which shall follow the list number. [*Example:* Consider a numbered for which a tab exists between the numbering symbol and the numbered paragraph's text. The tab would be specified in WordprocessingML as

Attributes	Description

	follows:
	
	`<w:lvl w:ilvl="0">`
	`…`
	`<w:suff w:val="tab" />`
	`…`
	`</w:lvl>`
	
	The val attribute with a value of tab specifies that the character between the numbering's level text and the paragraph text shall be a tab. This tab will follow normal tab stop rules. *end example*]
	
	The possible values for this attribute are defined by the ST_LevelSuffix simple type (§2.18.53).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LevelSuffix">
   <attribute name="val" type="ST_LevelSuffix" use="required"/>
</complexType>
```

2.9.31 tmpl (Numbering Template Code)

This element specifies a unique hexadecimal code which may be used to determine a location within application user interface in which this abstract numbering definition shall be displayed.

If this element is omitted, then this abstract numbering definition may be displayed in any location chosen by the consumer.

[*Example*: Consider the following abstract numbering definition:

```
<w:abstractNum w:abstractNumId="1">
  …
  <w:tmpl w:val="CA48B6BA" />
  …
</w:abstractNum>
```

In this example the abstractNum element with attribute abstractNumId equal to 1, would appear in the area within a consumer's application user interface specified by the template code CA48B6BA. *end example*]

Parent Elements
abstractNum (§2.9.1)

Attributes	Description
val (Long	Specifies a number value specified as a four digit hexadecimal number), whose contents

Attributes	Description
Hexadecimal Number Value)	of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following value for an attribute of type ST_LongHexNumber: `00BE2C6C`. This value is valid, as it contains four hexadecimal digits, each an encoding of an octet of the actual decimal number value. It may therefore be interpreted as desired in the context of the parent XML element, *end example*] The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).

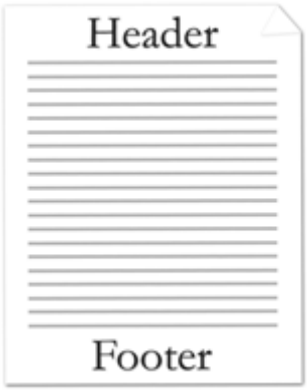
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LongHexNumber">
   <attribute name="val" type="ST_LongHexNumber" use="required"/>
</complexType>
```

2.10 Headers and Footers

Headers and footers refer to text, graphics or data (such as page number, date, document title, and so on) that can appear at the top or bottom of each page in a WordprocessingML document.

A header appears in the top margin (above the main document content on the page), while a footer appears in the bottom margin of a document page (below the main document content on the page).



Since WordprocessingML is a flow-based format, headers and footers are applied by specifying the headers and footers for all pages in a particular section of a document.

Within each section of a document there may be up to three different types of headers and footers:

- First page header/footer
- Odd page header/footer
- Even page header/footer

First page headers and footers specify a unique header or footer which shall appear on the first page of a section. Odd page headers and footers specify a unique header and footer which shall appear on all odd numbered pages for a given section. Even page headers and footers specify a unique header and footer which shall appear on all even numbered pages in a given section.

2.10.1 evenAndOddHeaders (Different Even/Odd Page Headers and Footers)

This element specifies whether sections in this document shall have different headers and footers for even and odd pages (an odd page header/footer and an even page header/footer).

If the val attribute is set to `true`, then each section in the document shall use an odd page header for all odd numbered pages in the section, and an even page header for all even numbered pages in the section (counting each page in the section starting from one, regardless of the page numbering settings for the section). If the val attribute is set to `false`, then all pages in a section shall use the odd page header.

This setting does not affect the presence of a first page header on each section, which is specified using the titlePg element (§2.10.6). If a first page header is specified, then all subsequent pages shall have this setting applied, including the first page in the odd/even page count.

If this element is set to `false` and an even page header is specified , then it shall be ignored and only the odd page header shall be displayed. Conversely, if this element is set to true and either header type is omitted for a given section, then a blank header shall be created as needed (another header type shall not be used in its place).

If this element is omitted, then its value shall be assumed to be `false`.

[*Example*: Consider a document which shall have a different even and odd page header for each section in its contents. This requirement must be specified using the following WordprocessingML:

```
<w:settings>
  …
  <w:evenAndOddHeaders />
  …
</w:setttings>
```

Since the evenAndOddHeaders property is set (and its default value is `true`), this document will now have different headers and footers for even and odd pages. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element.

Attributes	Description
	A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.10.2 footerReference (Footer Reference)

This element specifies a single footer which shall be associated with the current section in the document. This footer shall be referenced via the id attribute, which specifies an explicit relationship to the appropriate Footer part in the WordprocessingML package.

If the relationship type of the relationship specified by this element is not http://schemas.openxmlformats.org/officeDocument/2006/footer, is not present, or does not have a TargetMode attribute value of Internal, then the document shall be considered non-conformant.

Within each section of a document there may be up to three different types of footers:

- First page footer
- Odd page footer
- Even page footer

The footer type specified by the current footerReference is specified via the type attribute.

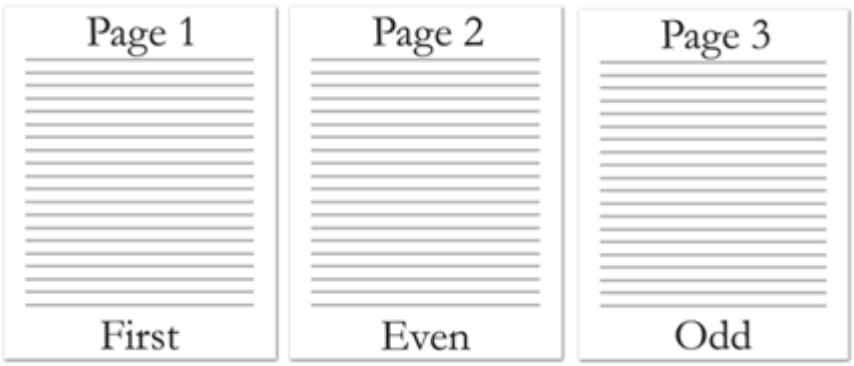
If any type of footer is omitted for a given section, then the following rules shall apply.

- If no footerReference for the first page footer is specified and the titlePg element is specified, then the first page footer shall be inherited from the previous section or, if this is the first section in the document, a new blank footer shall be created. If the titlePg element is not specified, then no first page footer shall be shown, and the odd page footer shall be used in its place.
- If no footerReference for the even page footer is specified and the evenAndOddHeaders element is specified, then the even page footer shall be inherited from the previous section or, if this is the first

section in the document, a new blank footer shall be created. If the evenAndOddHeaders element is not specified, then no even page footer shall be shown. and the odd page footer shall be used in its place.

- If no footerReference for the odd page footer is specified then the even page footer shall be inherited from the previous section or, if this is the first section in the document, a new blank footer shall be created.

[*Example*: Consider a three page document with different first, odd, and even page footers defined as follows:



This document defines three footers, each of have a relationship from the document part with a unique relationship ID, as shown in the following packaging markup:

```
<Relationships
xmlns=http://schemas.openxmlformats.org/package/2006/relationships>
  …
  <Relationship Id="rId6"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/footer
" Target="footer1.xml" />
  <Relationship Id="rId7"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/footer
" Target="footer2.xml" />
  <Relationship Id="rId10"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/footer
" Target="footer3.xml" />
  …
</Relationships>
```

These relationships are then referenced in the section's properties using the following WordprocessingML:

```
<w:sectPr>
  …
  <w:footerReference r:id="rId6" w:type="first" />
  <w:footerReference r:id="rId7" w:type="default" />
  <w:footerReference r:id="rId10" w:type="even" />
```

```
    …
  </w:sectPr>
```

The resulting section shall use the footer part with relationship id `rId6` for the first page, the footer part with relationship id `rId10` for all subsequent even pages, and the footer part with relationship id `rId7` for all subsequent odd pages. *end example*]

Parent Elements
sectPr (§2.6.18); sectPr (§2.6.19)

Attributes	Description
id (Relationship to Part) Namespace: …/officeDocument/2006/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the type required by the parent element: • http://schemas.openxmlformats.org/officeDocument/2006/relationships/footer for the footerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/header for the headerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://schemas.openxmlformats.org/officeDocument/2006/relationships/printerSettings for the printerSettings element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

type (Header or Footer Type)	Specifies the type of header or footer specified by the target relationship ID. This type determines the page(s) on which the current header or footer shall be displayed.
	
	If any section contains more than a single header or footer of each type, then the document shall be considered non-conformant.
	
	[*Example*: Consider a document with the following WordprocessingML:
	
	`<w:sectPr>`
	`  …`
	`  <w:footerReference r:id="rId6" w:type="first" />`
	`  <w:footerReference r:id="rId7" w:type="first" />`

Attributes	Description
	<w:footerReference r:id="rId10" w:type="even" /> … </w:sectPr> The resulting section has two footers of type `first`, and therefore is invalid. *end example*] [*Example*: Consider a WordprocessingML section which specifies the following header reference: <w:headerReference r:id="rId10" w:type="first" /> The resulting section shall use the specified header part for the `first` page. *end example*] The possible values for this attribute are defined by the ST_HdrFtr simple type (§2.18.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HdrFtrRef">
   <complexContent>
      <extension base="CT_Rel">
         <attribute name="type" type="ST_HdrFtr" use="required"/>
      </extension>
   </complexContent>
</complexType>
```

2.10.3 ftr (Footer)

This element specifies the content for a single footer for use within one or more sections of a WordprocessingML document.

Within the ftr element, the content of the element is similar to the content of the body (§2.2.2) element, and contains what is referred to as *block-level markup* - markup which can exist as a sibling element to paragraphs in a WordprocessingML document.

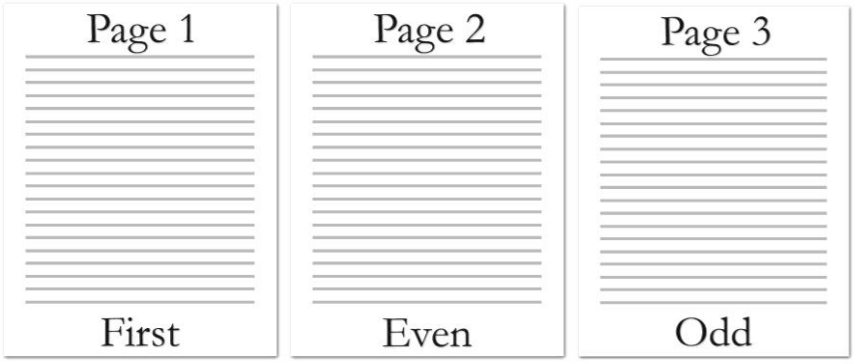
[*Example*: Consider the following simple one page document with one footer:

This document defines one footer with the text `footer`. The footer contents are stored in a unique footer part. The resulting footer is represented by the following WordprocessingML:

```
<w:ftr>
  <w:p>
    <w:r>
      <w:t>Footer</w:t>
    </w:r>
  </w:p>
</w:ftr>
```

Since footers are containers of block level contents, all block level elements can be used within them. In this particular example, the content is a single paragraph. *end example*]

[*Example*: Consider a more complex three page document with different first, odd, and even page footers defined:



This document defines three footers stored in three different footer parts. The resulting footers are represented by the following WordprocessingML:

First page footer part:

```
<w:ftr>
  <w:p>
    <w:r>
      <w:t>First</w:t>
    </w:r>
  </w:p>
</w:ftr>
```

Even page footer part:

```
<w:ftr>
  <w:p>
    <w:r>
      <w:t>Even</w:t>
    </w:r>
  </w:p>
</w:ftr>
```

Odd page footer part:

```
<w:ftr>
  <w:p>
    <w:r>
      <w:t>Odd</w:t>
    </w:r>
  </w:p>
</w:ftr>
```

end example]

Parent Elements
Root element of WordprocessingML Footer part

Child Elements	Subclause
altChunk (Anchor for Imported External Content)	§2.17.3.1
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Block-Level Custom XML Element)	§2.5.1.6
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5

Child Elements	Subclause
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
del (Deleted Run Content)	§2.13.5.12
ins (Inserted Run Content)	§2.13.5.20
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
p (Paragraph)	§2.3.1.22
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
sdt (Block-Level Structured Document Tag)	§2.5.2.30
tbl (Table)	§2.4.36

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HdrFtr">
    <group ref="EG_BlockLevelElts" minOccurs="1" maxOccurs="unbounded"/>
</complexType>
```

2.10.4 hdr (Header)

This element specifies the content for a single header for use within one or more sections of a WordprocessingML document.

Within the hdr element, the content of the element is similar to the content of the body (§2.2.2) element, and contains what is referred to as *block-level markup* - markup which can exist as a sibling element to paragraphs in a WordprocessingML document.

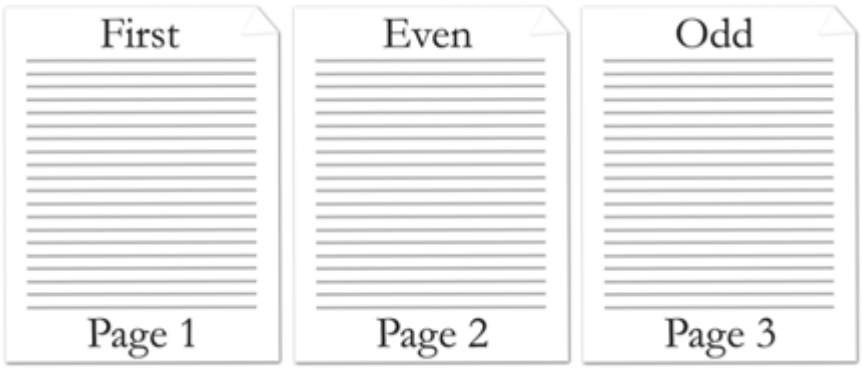
[*Example*: Consider the following simple one page document with one header:

This document defines one header with the text `Header`. The header's contents is stored in a unique Header part. The resulting header is represented by the following WordprocessingML:

```
<w:hdr>
  <w:p>
    <w:r>
      <w:t>Header</w:t>
    </w:r>
  </w:p>
</w:hdr>
```

Since headers are containers of block level contents, all block level elements can be used within them. In this particular example, the content is a single paragraph. *end example*]

[*Example*: Consider a more complex three page document with different first, odd, and even page headers defined:



This document defines three headers stored in three different header parts. The resulting headers are represented by the following WordprocessingML:

First page header part:

```
<w:hdr>
  <w:p>
    <w:r>
      <w:t>First</w:t>
    </w:r>
  </w:p>
</w:hdr>
```

Even page header part:

```
<w:hdr>
  <w:p>
    <w:r>
      <w:t>Even</w:t>
    </w:r>
  </w:p>
</w:hdr>
```

Odd page header part:

```
<w:hdr>
  <w:p>
    <w:r>
      <w:t>Odd</w:t>
    </w:r>
  </w:p>
</w:hdr>
```

end example]

Parent Elements
Root element of WordprocessingML Header part

Child Elements	Subclause
altChunk (Anchor for Imported External Content)	§2.17.3.1
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Block-Level Custom XML Element)	§2.5.1.6
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5

Child Elements	Subclause
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
del (Deleted Run Content)	§2.13.5.12
ins (Inserted Run Content)	§2.13.5.20
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
p (Paragraph)	§2.3.1.22
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
sdt (Block-Level Structured Document Tag)	§2.5.2.30
tbl (Table)	§2.4.36

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HdrFtr">
    <group ref="EG_BlockLevelElts" minOccurs="1" maxOccurs="unbounded"/>
</complexType>
```

2.10.5 headerReference (Header Reference)

This element specifies a single header which shall be associated with the current section in the document. This header shall be referenced via the id attribute, which specifies an explicit relationship to the appropriate Header part in the WordprocessingML package.

If the relationship type of the relationship specified by this element is not http://schemas.openxmlformats.org/officeDocument/2006/header, is not present, or does not have a TargetMode attribute value of `Internal`, then the document shall be considered non-conformant.

Within each section of a document there may be up to three different types of headers:

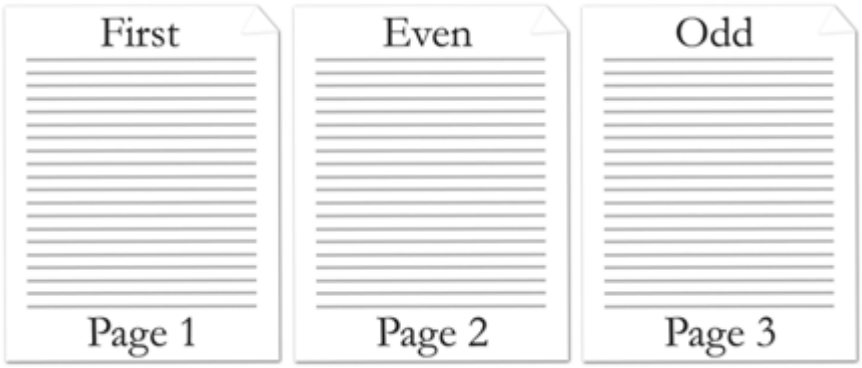
- First page header
- Odd page header
- Even page header

The header type specified by the current headerReference is specified via the type attribute.

If any type of header is omitted for a given section, then the following rules shall apply.

- If no headerReference for the first page header is specified and the titlePg element is specified, then the first page header shall be inherited from the previous section or, if this is the first section in the document, a new blank header shall be created. If the titlePg element is not specified, then no first page header shall be shown, and the odd page header shall be used in its place.
- If no headerReference for the even page header is specified and the evenAndOddHeaders element is specified, then the even page header shall be inherited from the previous section or, if this is the first section in the document, a new blank header shall be created. If the evenAndOddHeaders element is not specified, then no even page header shall be shown, and the odd page header shall be used in its place.
- If no headerReference for the odd page header is specified then the even page header shall be inherited from the previous section or, if this is the first section in the document, a new blank header shall be created.

[*Example*: Consider a three page document with different first, odd, and even page header defined as follows:



This document defines three headers, each of have a relationship from the document part with a unique relationship ID, as shown in the following packaging markup:

```
<Relationships
xmlns=http://schemas.openxmlformats.org/package/2006/relationships>
  …
  <Relationship Id="rId2"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/header
" Target="header1.xml" />
```

```
  <Relationship Id="rId3"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/header
" Target="header2.xml" />
  <Relationship Id="rId5"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/header
" Target="header3.xml" />
  …
</Relationships>
```

These relationships are then referenced in the section's properties using the following WordprocessingML:

```
<w:sectPr>
  …
  <w:headerReference r:id="rId3" w:type="first" />
  <w:headerReference r:id="rId5" w:type="default" />
  <w:headerReference r:id="rId2" w:type="even" />
  …
</w:sectPr>
```

The resulting section shall use the header part with relationship id `rId3` for the first page, the header part with relationship id `rId2` for all subsequent even pages, and the header part with relationship id `rId5` for all subsequent odd pages. *end example*]

Parent Elements
sectPr (§2.6.18); sectPr (§2.6.19)

Attributes	Description
id (Relationship to Part) Namespace: .../officeDocument/2006/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the type required by the parent element: • http://schemas.openxmlformats.org/officeDocument/2006/relationships/footer for the footerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/header for the headerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://schemas.openxmlformats.org/officeDocument/2006/relationships/printerSettings for the printerSettings element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />`

Attributes	Description
	The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

type (Header or Footer Type)	Specifies the type of header or footer specified by the target relationship ID. This type determines the page(s) on which the current header or footer shall be displayed.
	
	If any section contains more than a single header or footer of each type, then the document shall be considered non-conformant.
	
	[*Example*: Consider a document with the following WordprocessingML:
	
	`<w:sectPr>`
	`…`
	`<w:footerReference r:id="rId6" w:type="first" />`
	`<w:footerReference r:id="rId7" w:type="first" />`
	`<w:footerReference r:id="rId10" w:type="even" />`
	`…`
	`</w:sectPr>`
	
	The resulting section has two footers of type `first`, and therefore is invalid. *end example*]
	
	[*Example*: Consider a WordprocessingML section which specifies the following header reference:
	
	`<w:headerReference r:id="rId10" w:type="first" />`
	
	The resulting section shall use the specified header part for the `first` page. *end example*]
	
	The possible values for this attribute are defined by the ST_HdrFtr simple type (§2.18.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HdrFtrRef">
   <complexContent>
      <extension base="CT_Rel">
         <attribute name="type" type="ST_HdrFtr" use="required"/>
      </extension>
   </complexContent>
</complexType>
```

2.10.6 titlePg (Different First Page Headers and Footers)

This element specifies whether the parent section in this document shall have a different header and footer for its first page.

If the val attribute is set to `true`, then the parent section in the document shall use a first page header for the first page in the section. If the val attribute is set to `false`, then the first page in the parent section shall use the odd page header.

This setting does not affect the presence of even and odd page header on all sections, which is specified using the evenAndOddHeaders element (§2.10.1).

If this element is set to `false` and a first page header is specified , then it shall be ignored and only the odd page header shall be displayed. Conversely, if this element is set to `true` and the first page header type is omitted for the given section, then a blank header shall be created as needed (another header type shall not be used in its place).

If this element is omitted, then its value shall be assumed to be `false`.

[*Example*: Consider a section which shall have a different first page header. This requirement is specified using the following WordprocessingML:

```
<w:sectPr>
  …
  <w:titlePg />
  …
</w:sectPr>
```

Since the titlePg property is present (and its default attribute value is `true`), this document will now have a different header and footer for its first page. *end example*]

Parent Elements
sectPr (§2.6.17); sectPr (§2.6.18); sectPr (§2.6.19)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property:

Attributes	Description
	`<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.11 Footnotes and Endnotes

Footnotes and *endnotes* are separate text storied used in documents and books to show the source of borrowed material or to enter explanatory or supplementary information which does not interrupt the normal reading flow of the document.

Footnotes are typically located at the bottom of a page or beneath text being referenced, and *endnotes* are typically placed at the end of a document or at the end of a section. If document has been divided up into one or more sections, each section of a document may contain endnotes.

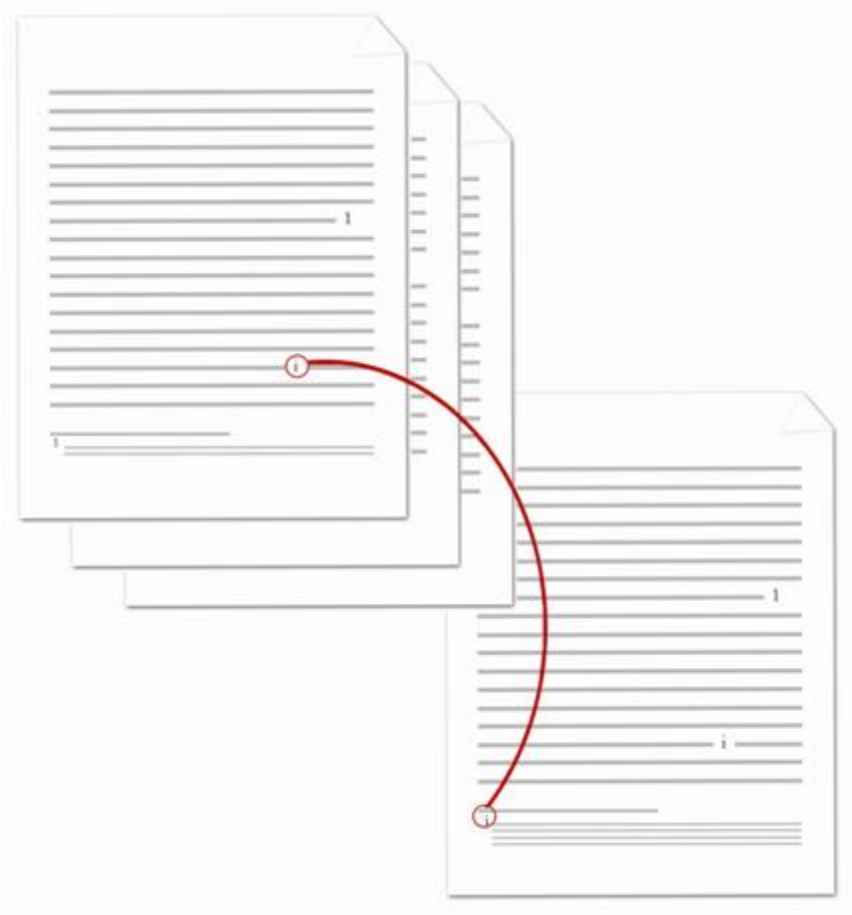
Both footnotes and endnotes consist of two parts:

- A note reference mark in the body text to indicate that additional information is in a footnote or endnote, with a numbering system used for each to tell readers whether to look for the note at the end of the page or the end of the document or section.
- The actual footnote or endnote story content.

[*Example:* Example of a footnote applied to text in a document:

The note reference mark follows the noted text and specifies that there is associated footnote information, and the footnote itself is at the bottom of the current page. *end example*]

[*Example:* Consider the following example of an endnote applied to text in a document:



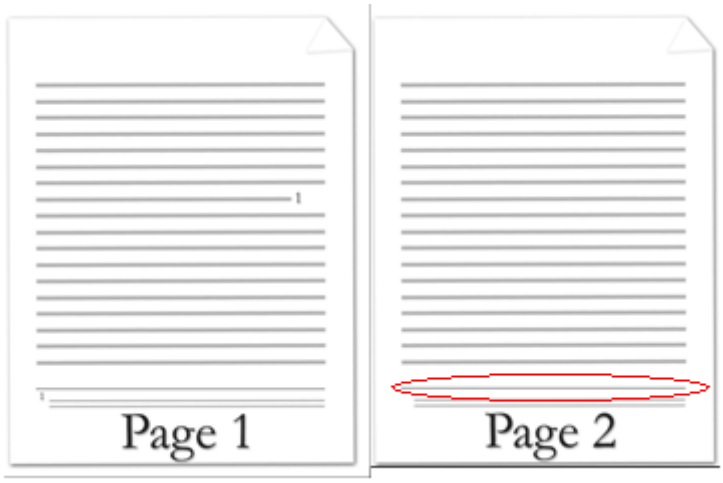
The note reference mark follows the noted text and specifies that there is associated endnote information, and the endnote itself is at the end of the current section. *end example*]

2.11.1 continuationSeparator (Continuation Separator Mark)

This element specifies the presence of a continuation separator mark within the current run. A continuation separator mark is a horizontal line which spans the width of the main story's text extents.

[*Note*: The continuation separator mark is typically used within the context of continuation separator footnotes or endnotes. These footnote and endnote types define the footnote/endnote used to separate the contents of the main document story from continuation of footnotes or endnotes which began on a previous page. *end note*]

[*Example*: Consider the following two pages in a document, where some text is referenced by a footnote that extends to the next page (with the continuation separator circled in red):



The line separating the document text from the footnote that is continued on the next page is represented by the following WordprocessingML:

```
<w:footnote w:type="continuationSeparator" w:id="1">
  <w:p>
    <w:r>
      <w:continuationSeparator />
    </w:r>
  </w:p>
</w:footnote>
```

In this example, the footnote has a content which consists of a single continuationSeparator, which is displayed as a horizontal line across the text extents. *end example*]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

2.11.2 endnote (Endnote Content)

This element specifies the content of a single endnote within a WordprocessingML document. Each endnote shall be represented by a single endnote element, which may contain any valid *block-level content*.

[*Example*: Consider a document with a single endnote, identified by a endnote element, defined in the endnotes part:

```
<w:endnotes>
  <w:endnote w:id="2">
    <w:p>
      <w:pPr>
        <w:pStyle w:val="EndnoteText" />
      </w:pPr>
      <w:r>
        <w:rPr>
          <w:rStyle w:val="EndnoteReference" />
        </w:rPr>
        <w:endnoteRef />
      </w:r>
      <w:r>
        <w:t xml:space="preserve">This is an endnote</w:t>
      </w:r>
    </w:p>
  </w:endnote>
</w:endnotes>
```

This endnote contains an endnote reference mark, as well as the endnote text `This is an endnote`. *end example*]

Parent Elements
endnotes (§2.11.8)

Child Elements	Subclause
altChunk (Anchor for Imported External Content)	§2.17.3.1
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Block-Level Custom XML Element)	§2.5.1.6
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10

Child Elements	Subclause
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
del (Deleted Run Content)	§2.13.5.12
ins (Inserted Run Content)	§2.13.5.20
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
p (Paragraph)	§2.3.1.22
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
sdt (Block-Level Structured Document Tag)	§2.5.2.30
tbl (Table)	§2.4.36

Attributes	Description

id (Footnote/Endnote ID)	Specifies a unique ID which shall be used to match the contents of a footnote or endnote to the associated footnote/endnote reference mark in the document using the footnoteRef or endnoteRef element, as appropriate.
	
	If this attribute is omitted, then this footnote or endnote shall have no ID. If more than one footnote shares the same ID, then this document shall be considered non-conformant. If more than one endnote shares the same ID, then this document shall be considered non-conformant.
	
	[*Example*: Consider the following footnote as defined in the footnotes part:
	
	`<w:footnotes>`
	`  <w:footnote w:type="normal" w:id="0">`
	`  …`
	`  </w:footnote>`
	`  …`
	`</w:footnotes>`
	
	The contents of this footnote are associated with the footnoteReference with a matching ID, as follows:

Attributes	Description
	<w:p> <w:r> <w:footnoteReference w:id="0" /> </w:r> </w:p> The resulting paragraph will have a footnote reference mark which references the footnote number value of the footnote with an id of 0. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
type (Footnote/Endnote Type)	Specifies the type of footnote or endnote contained within the current footnote or endnote content definition. If this attribute is omitted, then it shall be considered to be of type normal. If a footnote or endnote is not of type normal, then it shall not be referenced by a footnoteReference or endnoteReference element within the main document story. [*Example*: Consider the following example of a footnote defined in a WordprocessingML document as follows: <w:footnote w:type="continuationSeparator" w:id="1"> <w:p> <w:r> <w:continuationSeparator /> </w:r> </w:p> </w:footnote> In this example, the footnote is of type continuationSeparator and shall be used by a consumer to separate continued footnotes from the main document contents (see simple type for full details). *end example*] The possible values for this attribute are defined by the ST_FtnEdn simple type (§2.18.37).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FtnEdn">
   <sequence>
      <group ref="EG_BlockLevelElts" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="type" type="ST_FtnEdn" use="optional"/>
   <attribute name="id" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.11.3 endnote (Special Endnote List)

This element specifies the ID for all endnotes which are located in the current document that are not of type `normal`. Each other type of endnote shall be referenced in this list, or it shall not be loaded. If an endnote is not listed beneath this element, and it is required by the document content, then the document shall be considered non-conformant.

[*Example*: Consider a document that has three endnotes represented by the following WordprocessingML:

```
<w:endnotes ...>
  <w:endnote w:type="separator" w:id="0">
    ...
  </w:endnote>
  <w:endnote w:type="continuationSeparator" w:id="1">
    ...
  </w:endnote>
  <w:endnote w:id="2">
    ...
  </w:endnote>
</w:endnotes>
```

Each of the endnotes which are not of type `normal` must be specified in the endnotePr element, as follows:

```
<w:endnotePr>
  <w:endnote w:id="0" />
  <w:endnote w:id="1" />
</w:endnotePr>
```

This indicates to the consumer that the endnotes with an id attribute value of `0` and `1` are special endnotes, and should be treated accordingly. *end example*]

Parent Elements
endnotePr (§2.11.4)

Attributes	Description
id (Footnote/Endnote ID)	Specifies a unique ID which shall be used to match the contents of a footnote or endnote to the associated footnote/endnote reference mark in the document using the footnoteRef or endnoteRef element, as appropriate. If more than one footnote shares the same ID, then this document shall be considered non-conformant. If more than one endnote shares the same ID, then this document shall be considered non-conformant. [*Example*: Consider the following footnote as defined in the footnotes part:

Attributes	Description
	<w:footnotes> <w:footnote w:type="normal" w:id="0"> … </w:footnote> … </w:footnotes> The contents of this footnote are associated with the footnoteReference with a matching ID, as follows: <w:p> <w:r> <w:footnoteReference w:id="0" /> </w:r> </w:p> The resulting paragraph will have a footnote reference mark which references the footnote number value of the footnote with an id of 0. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

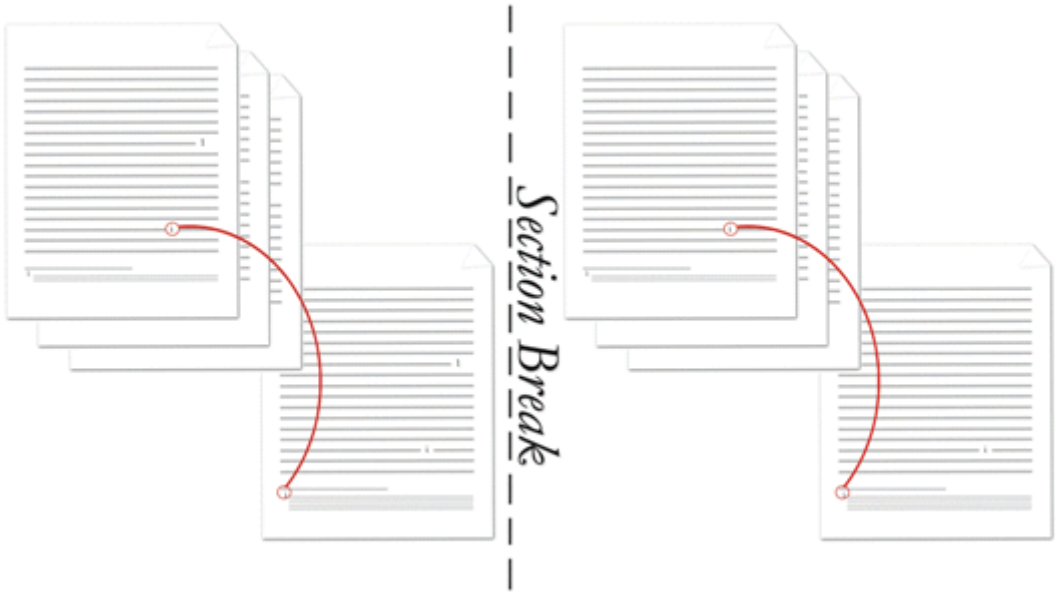
```
<complexType name="CT_FtnEdnSepRef">
    <attribute name="id" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.11.4 endnotePr (Document-Wide Endnote Properties)

This element specifies the endnote properties for the current document. Each of these properties are stored as a child element within the endnotePr element.

These properties may be overridden for a specific section via the section-wide endnotePr element (§2.11.5).

[*Example*: Consider the following document with two sections, where the endnotes for each section appears at the end of that section and use lower case roman numerals:



Since both sections are identical, the endnote properties are specified as document-wide level properties (this is not necessary but is most efficient) as follows:

```
<w:settings>
  …
  <w:endnotePr>
    <w:numFmt w:val="lowerRoman" />
    <w:pos w:val="sectEnd"/>
  </w:endnotePr>
  …
</w:settings>
```

Note that the pos element could have been omitted since it is using its default value. *end example*]

Parent Elements
settings (§2.15.1.78)

Child Elements	Subclause
endnote (Special Endnote List)	§2.11.3
numFmt (Endnote Numbering Format)	§2.11.18
numRestart (Footnote and Endnote Numbering Restart Location)	§2.11.19
numStart (Footnote and Endnote Numbering Starting Value)	§2.11.20
pos (Endnote Placement)	§2.11.22

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EdnDocProps">
    <complexContent>
        <extension base="CT_EdnProps">
            <sequence>
                <element name="endnote" type="CT_FtnEdnSepRef" minOccurs="0" maxOccurs="3"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

2.11.5 endnotePr (Section-Wide Endnote Properties)

This element specifies the endnote properties for the current section. Each of these properties are an override of the document-wide endnote properties (§2.11.4) and are stored as a child element within the endnotePr element.

If this element is omitted for a given section, then that section shall use the endnote properties defined at the document-wide level.

[*Example*: Consider a document consisting of three sections, which has endnotes in the first section which use lowercase roman numerals, and endnotes in the third section which use the Chicago Manual of Style format. The WordprocessingML for each section would be specified as follows:

```
<w:sectPr>
  <w:endnotePr>
    <w:numFmt w:val="lowerRoman" />
  </w:endnotePr>
</w:sectPr>
…
<w:sectPr>
  …
</w:sectPr>
…
<w:sectPr>
  …
</w:sectPr>
```

This assumes that the document-wide endnote settings are specified to use the Chicago Manual of Style format, as follows:

```
<w:settings>
  <w:endnotePr>
    <w:numFmt w:val="chicago" />
  </w:endnotePr>
</w:settings>
```

The resulting document would override the endnote numbering format for the first section to `lowerRoman`, but would use the `chicago` endnote numbering format for section three (and would also use it for section two if that section had endnotes. *end example*]

Parent Elements
sectPr (§2.6.17); sectPr (§2.6.18); sectPr (§2.6.19)

Child Elements	Subclause
numFmt (Endnote Numbering Format)	§2.11.18
numRestart (Footnote and Endnote Numbering Restart Location)	§2.11.19
numStart (Footnote and Endnote Numbering Starting Value)	§2.11.20
pos (Endnote Placement)	§2.11.22

The following XML Schema fragment defines the contents of this element:

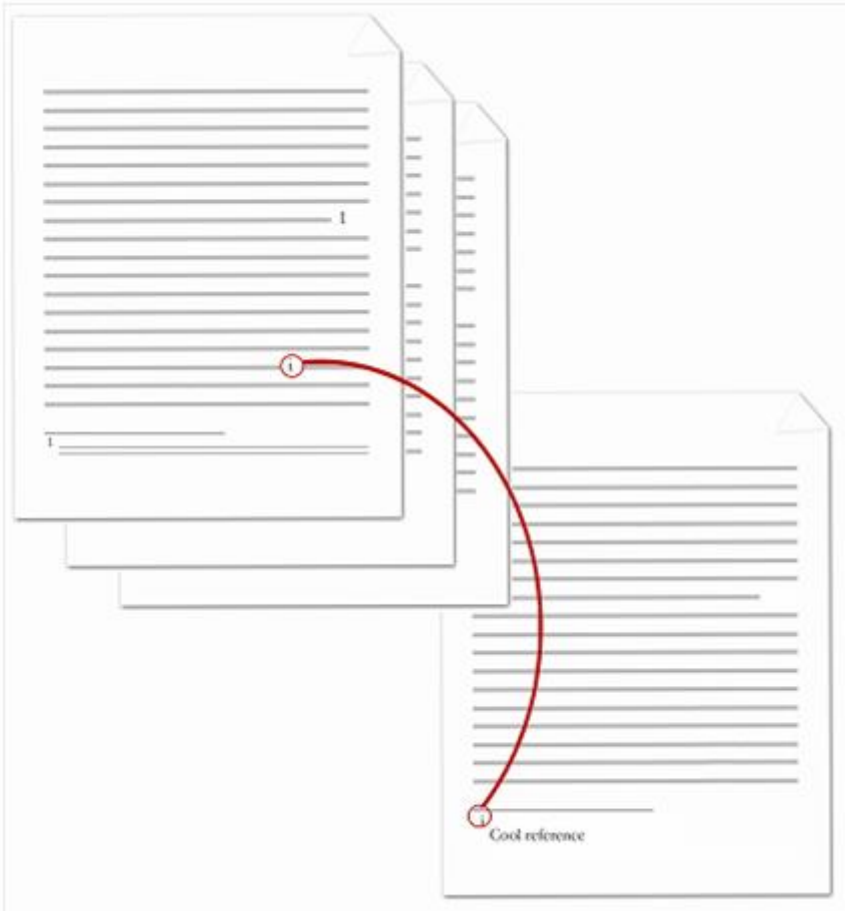
```
<complexType name="CT_EdnProps">
   <sequence>
      <element name="pos" type="CT_EdnPos" minOccurs="0"/>
      <element name="numFmt" type="CT_NumFmt" minOccurs="0"/>
      <group ref="EG_FtnEdnNumProps" minOccurs="0"/>
   </sequence>
</complexType>
```

2.11.6 endnoteRef (Endnote Reference Mark)

This element specifies the presence of an endnote reference mark. An *endnote reference mark* is a run of automatically numbered text which follows the numbering format set forth via the numFmt element (§2.11.18).

If an endnote reference mark is specified within a run which is not part of an endnote, then that endnote reference mark may be ignored.

[*Example*: Consider the following document where some text is referenced by an endnote at the end of the document:



The endnote reference mark is the lower case roman numeral within the actual endnote itself in the diagram above. The contents of the endnote (including the endnote reference mark) are represented by the following WordprocessingML:

```
<w:endnote w:id="2">
  <w:p>
    <w:pPr>
      <w:pStyle w:val="EndnoteText" />
    </w:pPr>
    <w:r>
      <w:rPr>
        <w:rStyle w:val="EndnoteReference" />
      </w:rPr>
      <w:endfootnoteRef />
    </w:r>
    <w:r>
      <w:t>Cool reference</w:t>
    </w:r>
  </w:p>
</w:endnote>
```

The resulting endnote contains the literal endnote content of `Cool reference`, preceding by an automatically numbered endnote reference mark. Since this is the first endnote in the document, that automatically numbered reference mark uses the lower case roman numeral `i`. *end example*]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

The following XML Schema fragment defines the contents of this element:

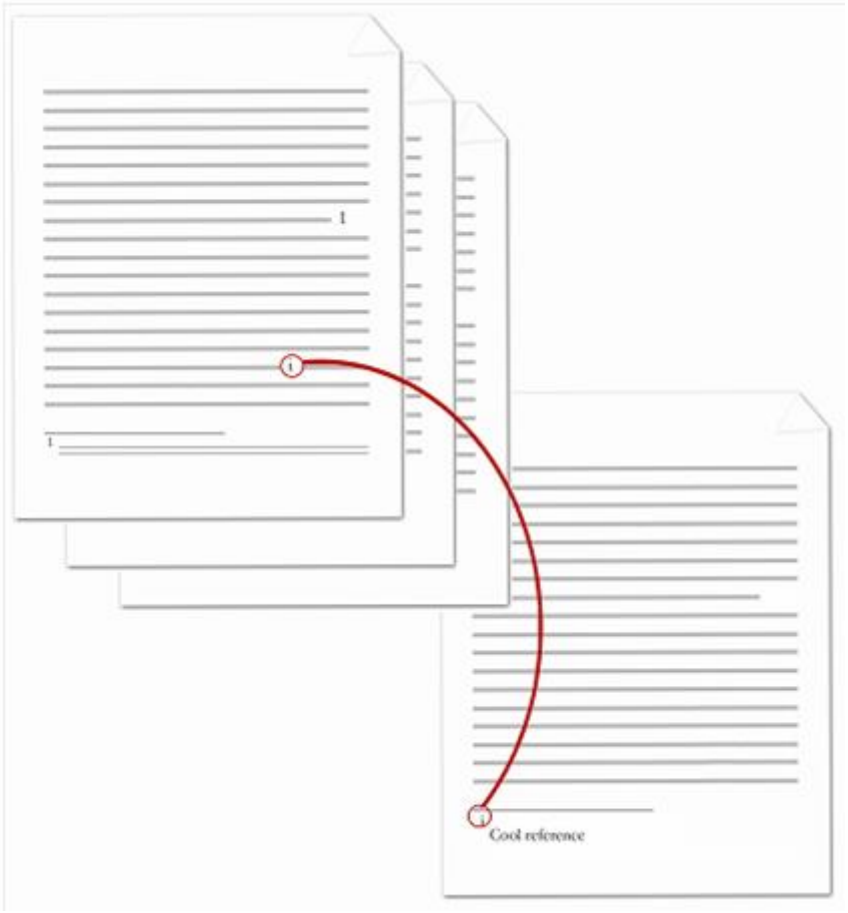
```
<complexType name="CT_Empty"/>
```

2.11.7 endnoteReference (Endnote Reference)

This element specifies the presence of an endnote reference. An *endnote reference* is a run of automatically numbered text which references a particular endnote within the parent document, and inherits the endnote reference mark's numbering.

If an endnote reference is specified within a footnote or endnote, then the document shall be considered non-conformant.

[*Example*: Consider the following document where some text is referenced by an endnote at the end of the document:



The endnote reference is the lower case roman numeral within the document content in the diagram above. The contents of the paragraph which contains the endnote reference are represented by the following WordprocessingML:

```
<w:p>
  <w:r>
    <w:t>This text is followed by an endnote</w:t>
  </w:r>
  <w:r>
    <w:rPr>
      <w:rStyle w:val="EndnoteReference" />
    </w:rPr>
    <w:endnoteReference w:id="2" />
  </w:r>
  <w:r>
    <w:t>.</w:t>
  </w:r>
</w:p>
```

The resulting paragraph contains the literal text content of `This text is followed by an endnote`, followed by an automatically numbered endnote reference. Since this is the first endnote in the document, that automatically numbered reference inherits the lower case roman numeral `i` from the endnote reference mark. *end example*]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

Attributes	Description

customMarkFollows (Suppress Footnote/Endnote Reference Mark)	Specifies that the current footnote or endnote shall not have an associated footnote or endnote reference mark, as appropriate.
	
	This attribute shall be used to specify that a particular footnote or endnote shall not increment the numbering for its associated footnote/endnote numbering format, so that the use of a footnote with a custom footnote mark does not cause a missing value in the footnote/endnote values. The display of the mark is specified via the footnoteRef/endnoteRef elements, as appropriate.
	
	If this attribute is omitted, then the footnote or endnote reference mark shall not be skipped when incrementing over this footnote or endnote.
	
	[*Example*: Consider a footnote with an id value of 1 that uses a custom footnote mark:
	
	`<w:footnotes>`
	`  <w:footnote w:id="0">`
	`  …`
	`  </w:footnote>`
	`  <w:footnote w:suppressRef="1" w:id="2">`
	`  …`
	`  </w:footnote>`
	`  <w:footnote w:id="2">`
	`  …`
	`  </w:footnote>`
	`</w:footnotes>`
	
	If the numbering format for footnotes in this document is upperRoman, then the first footnote shall be `I`, the second is suppressed, and the third is `II`, noticing that the second does not increment the numbering sequence. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

id (Footnote/Endnote ID Reference)	Specifies the footnote or endnote which is being referenced by the current footnote or endnote reference in the document. If the resulting footnote or endnote ID is not present in the footnotes or endnote part (as appropriate), then this document shall be considered non-conformant.

Attributes	Description

	[*Example*: Consider a paragraph with an endnote reference, represented by the following WordprocessingML:
	
	`<w:p>`
	`  <w:r>`
	`  <w:t>This text is followed by an endnote</w:t>`
	`  </w:r>`
	`  <w:r>`
	`  <w:endnoteReference w:id="2" />`
	`  </w:r>`
	`  <w:r>`
	`  <w:t>.</w:t>`
	`  </w:r>`
	`</w:p>`
	
	This text references the endnote in the document's endnotes part which has an id value of 2. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FtnEdnRef">
   <attribute name="customMarkFollows" type="ST_OnOff" use="optional"/>
   <attribute name="id" use="required" type="ST_DecimalNumber"/>
</complexType>
```

2.11.8 endnotes (Document Endnotes)

This element specifies the set of all endnotes in the document, including endnote separators and continuation notices. This element is the root node for the Endnotes part.

[*Example*: Consider the following example of the contents of the endnotes part:

```
<w:endnotes>
  <w:endnote w:type="separator" w:id="0">
    …
  </w:endnote>
  <w:endnote w:type="continuationSeparator" w:id="1">
    …
  </w:endnote>
  <w:endnote w:id="2">
    …
  </w:endnote>
</w:endnotes>
```

The endnotes part contains the definition for one normal endnote, as well as the separator and continuation separator endnote for this document. *end example*]

Parent Elements
Root element of WordprocessingML Endnotes part

Child Elements	Subclause
endnote (Endnote Content)	§2.11.2

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Endnotes">
    <sequence maxOccurs="unbounded">
        <element name="endnote" type="CT_FtnEdn" minOccurs="0"/>
    </sequence>
</complexType>
```

2.11.9 footnote (Special Footnote List)

This element specifies the ID for all footnotes which are located in the current document that are not of type `normal`. Each other type of footnote shall be referenced in this list, or it shall not be loaded. This means that if a special footnote is not listed beneath this element, and it is required by the document content, then the document shall be considered non-conformant.

[*Example*: Consider a document that has three footnotes represented by the following WordprocessingML:

```
<w:footnotes>
  <w:footnote w:type="separator" w:id="0">
    ...
  </w:footnote>
  <w:footnote w:type="continuationSeparator" w:id="1">
    ...
  </w:footnote >
  <w:footnote w:id="2">
    ...
  </w:footnote>
</w:footnotes>
```

Each of the footnotes which are not of type `normal` must be specified in the footnotePr element, as follows:

```
<w:footnotePr>
  <w:footnote w:id="0" />
  <w:footnote w:id="1" />
</w:footnotePr>
```

This indicates to the consumer that the footnotes with an id attribute value of 0 and 1 are special footnotes, and should be treated accordingly. *end example*]

Parent Elements
footnotePr (§2.11.11)

Attributes	Description

id (Footnote/Endnote ID)	Specifies a unique ID which shall be used to match the contents of a footnote or endnote to the associated footnote/endnote reference mark in the document using the footnoteRef or endnoteRef element, as appropriate.
	
	If more than one footnote shares the same ID, then this document shall be considered non-conformant. If more than one endnote shares the same ID, then this document shall be considered non-conformant.
	
	[*Example*: Consider the following footnote as defined in the footnotes part:
	
	`<w:footnotes>`
	`  <w:footnote w:type="normal" w:id="0">`
	`  …`
	`  </w:footnote>`
	`  …`
	`</w:footnotes>`
	
	The contents of this footnote are associated with the footnoteReference with a matching ID, as follows:
	
	`<w:p>`
	`  <w:r>`
	`  <w:footnoteReference w:id="0" />`
	`  </w:r>`
	`</w:p>`
	
	The resulting paragraph will have a footnote reference mark which references the footnote number value of the footnote with an id of 0. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FtnEdnSepRef">
    <attribute name="id" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.11.10 footnote (Footnote Content)

This element specifies the content of a single footnote within a WordprocessingML document. Each footnote shall be represented by a single footnote element, which may contain any valid *block-level content*.

[*Example*: Consider a document with a single footnote, identified by a footnote element, defined in the footnotes part as follows:

```
<w:footnotes>
  <w:footnote w:id="2">
    <w:p>
      <w:pPr>
        <w:pStyle w:val="FootnoteText" />
      </w:pPr>
      <w:r>
        <w:rPr>
          <w:rStyle w:val="FootnoteReference" />
        </w:rPr>
        <w:footnoteRef />
      </w:r>
      <w:r>
        <w:t xml:space="preserve">This is a sample footnote</w:t>
      </w:r>
    </w:p>
  </w:footnote>
</w:footnotes>
```

This footnote contains an footnote reference mark, as well as the endnote text `This is a sample footnote`. *end example*]

Parent Elements
footnotes (§2.11.15)

Child Elements	Subclause
altChunk (Anchor for Imported External Content)	§2.17.3.1
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Block-Level Custom XML Element)	§2.5.1.6
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5

Child Elements	Subclause
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
del (Deleted Run Content)	§2.13.5.12
ins (Inserted Run Content)	§2.13.5.20
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
p (Paragraph)	§2.3.1.22
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
sdt (Block-Level Structured Document Tag)	§2.5.2.30
tbl (Table)	§2.4.36

Attributes	Description

id (Footnote/Endnote ID)	Specifies a unique ID which shall be used to match the contents of a footnote or endnote to the associated footnote/endnote reference mark in the document using the footnoteRef or endnoteRef element, as appropriate.
	
	If this attribute is omitted, then this footnote or endnote shall have no ID. If more than one footnote shares the same ID, then this document shall be considered non-conformant. If more than one endnote shares the same ID, then this document shall be considered non-conformant.
	
	[*Example*: Consider the following footnote as defined in the footnotes part:
	
	`<w:footnotes>`
	`  <w:footnote w:type="normal" w:id="0">`

Attributes	Description
	… </w:footnote> … </w:footnotes> The contents of this footnote are associated with the footnoteReference with a matching ID, as follows: <w:p> <w:r> <w:footnoteReference w:id="0" /> </w:r> </w:p> The resulting paragraph will have a footnote reference mark which references the footnote number value of the footnote with an id of 0. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
type (Footnote/Endnote Type)	Specifies the type of footnote or endnote contained within the current footnote or endnote content definition. If this attribute is omitted, then it shall be considered to be of type normal. If a footnote or endnote is not of type normal, then it shall not be referenced by a footnoteReference or endnoteReference element within the main document story. [*Example*: Consider the following example of a footnote defined in a WordprocessingML document as follows: <w:footnote w:type="continuationSeparator" w:id="1"> <w:p> <w:r> <w:continuationSeparator /> </w:r> </w:p> </w:footnote> In this example, the footnote is of type continuationSeparator and shall be used by a consumer to separate continued footnotes from the main document contents (see simple type for full details). *end example*] The possible values for this attribute are defined by the ST_FtnEdn simple type (§2.18.37).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FtnEdn">
    <sequence>
        <group ref="EG_BlockLevelElts" minOccurs="1" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="type" type="ST_FtnEdn" use="optional"/>
    <attribute name="id" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.11.11 footnotePr (Document-Wide Footnote Properties)

This element specifies the footnote properties for this document. Each property is stored as a unique element within the footnotePr element.

These properties may be overridden for a specific section via the section-wide footnotePr element (§2.11.12).

[*Example*: Consider the following one page document, where the footnote appears beneath the text it references:

Since the document consists of a single footnote, the footnote properties may be stored in either the section-wide or document-wide footnote properties. Assuming that they are stored in the latter, the footnote properties are represented by the following WordprocessingML:

```
<w:settings>
  …
  <w:footnotePr>
    <w:pos w:val="beneathText" />
  </w:footnotePr>
  …
```

```
</w:settings>
```

The footnote properties specify that footnotes will appear below the noted text *end example*]

Parent Elements
settings (§2.15.1.78)

Child Elements	Subclause
footnote (Special Footnote List)	§2.11.9
numFmt (Footnote Numbering Format)	§2.11.17
numRestart (Footnote and Endnote Numbering Restart Location)	§2.11.19
numStart (Footnote and Endnote Numbering Starting Value)	§2.11.20
pos (Footnote Placement)	§2.11.21

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FtnDocProps">
   <complexContent>
      <extension base="CT_FtnProps">
         <sequence>
            <element name="footnote" type="CT_FtnEdnSepRef" minOccurs="0" maxOccurs="3"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.11.12 footnotePr (Section-Wide Footnote Properties)

This element specifies the footnote properties for the current section. Each of these properties are an override of the document-wide footnote properties (§2.11.11) and are stored as a child element within the footnotePr element.

If this element is omitted for a given section, then that section shall use the footnote properties defined at the document-wide level.

[*Example*: Consider a document consisting of three sections, which has footnotes in the first section which appear below text, and footnotes in the third section which appear at the bottom of the page. The WordprocessingML for each section would be specified as follows:

```
<w:sectPr>
  <w:footnotePr>
    <w:pos w:val="beneathText" />
  </w:footnotePr>
</w:sectPr>
```

```
…
<w:sectPr>
  …
</w:sectPr>
…
<w:sectPr>
  …
</w:sectPr>
```

This assumes that the document-wide footnote settings are specified as the default positioning at the bottom of the page by omitting the pos element (§2.11.21), as follows:

```
<w:settings>
  <w:footnotePr>
    …
  </w:footnotePr>
</w:settings>
```

The resulting document would override the footnote positioning for the first section to `beneathText`, but would use the `pageBottom` footnote positioning for section three (and would also use it for section two if that section had footnotes. *end example*]

Parent Elements
sectPr (§2.6.17); sectPr (§2.6.18); sectPr (§2.6.19)

Child Elements	Subclause
numFmt (Footnote Numbering Format)	§2.11.17
numRestart (Footnote and Endnote Numbering Restart Location)	§2.11.19
numStart (Footnote and Endnote Numbering Starting Value)	§2.11.20
pos (Footnote Placement)	§2.11.21

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FtnProps">
   <sequence>
      <element name="pos" type="CT_FtnPos" minOccurs="0"/>
      <element name="numFmt" type="CT_NumFmt" minOccurs="0"/>
      <group ref="EG_FtnEdnNumProps" minOccurs="0"/>
   </sequence>
</complexType>
```

2.11.13 footnoteRef (Footnote Reference Mark)

This element specifies the presence of a footnote reference mark. A *footnote reference mark* is a run of automatically numbered text which follows the numbering format set forth via the footnote numFmt element (§2.11.17).

If a footnote reference mark is specified within a run which is not part of a footnote, then that footnote reference mark may be ignored.

[*Example*: Consider the following document where some text is referenced by an footnote at the end of the page:

The footnote reference mark is the decimal number within the actual footnote itself in the image above. The contents of the footnote (including the footnote reference mark) are represented by the following WordprocessingML:

```
<w:footnote w:id="2">
  <w:p>
    <w:pPr>
      <w:pStyle w:val="FootnoteText" />
    </w:pPr>
    <w:r>
      <w:rPr>
        <w:rStyle w:val="FootnoteReference" />
      </w:rPr>
      <w:footnoteRef />
    </w:r>
    <w:r>
```

```
        <w:t>Cool reference</w:t>
      </w:r>
    </w:p>
  </w:footnote>
```

The resulting footnote contains the literal endnote content of `Cool reference`, preceding by an automatically numbered footnote reference mark. Since this is the first footnote in the document, that automatically numbered reference mark uses the first decimal number `1`. It is also important to note that the use of styles `FootnoteText` and `FootnoteReference` is not required, these may simply be added by a particular producer automatically to give the footnote contents are particular style (just like any other use of styles). *end example*]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

2.11.14 footnoteReference (Footnote Reference)

This element specifies the presence of a footnote reference. A *footnote reference* is a run of automatically numbered text which references a particular footnote within the parent document, and inherits the footnote reference mark's numbering.

If an footnote reference is specified within a footnote or endnote, then the document shall be considered non-conformant.

[*Example*: Consider the following document where some text is referenced by a footnote at the bottom of the page:

The footnote reference is the superscript decimal number within the document content in the diagram above. The contents of the paragraph which contains the footnote reference are represented by the following WordprocessingML:

```
<w:p>
  <w:r>
    <w:t>Some referenced text.</w:t>
  </w:r>
  <w:r>
    <w:rPr>
      <w:rStyle w:val="FootnoteReference" />
    </w:rPr>
    <w:footnoteReference w:id="2" />
  </w:r>
</w:p>
```

The resulting paragraph contains the literal text content of `Some referenced text.`, followed by an automatically numbered footnote reference. Since this is the first footnote in the document, that automatically numbered reference inherits the decimal number 1 from the footnote reference mark. *end example*]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

Attributes	Description

customMarkFollows (Suppress Footnote/Endnote Reference Mark)	Specifies that the current footnote or endnote shall not have an associated footnote or endnote reference mark, as appropriate.
	
	This attribute shall be used to specify that a particular footnote or endnote shall not increment the numbering for its associated footnote/endnote numbering format, so that the use of a footnote with a custom footnote mark does not cause a missing value in the footnote/endnote values. The display of the mark is specified via the footnoteRef/endnoteRef elements, as appropriate.
	
	If this attribute is omitted, then the footnote or endnote reference mark shall not be skipped when incrementing over this footnote or endnote.
	
	[*Example*: Consider a footnote with an id value of 1 that uses a custom footnote mark:
	
	`<w:footnotes>`
	`  <w:footnote w:id="0">`
	`    …`
	`  </w:footnote>`
	`  <w:footnote w:suppressRef="1" w:id="2">`
	`    …`

Attributes	Description
	</w:footnote> <w:footnote w:id="2"> … </w:footnote> </w:footnotes> If the numbering format for footnotes in this document is upperRoman, then the first footnote shall be I, the second is suppressed, and the third is II, noticing that the second does not increment the numbering sequence. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
id (Footnote/Endnote ID Reference)	Specifies the footnote or endnote which is being referenced by the current footnote or endnote reference in the document. If the resulting footnote or endnote ID is not present in the footnotes or endnote part (as appropriate), then this document shall be considered non-conformant. [*Example*: Consider a paragraph with an endnote reference, represented by the following WordprocessingML: <w:p> <w:r> <w:t>This text is followed by an endnote</w:t> </w:r> <w:r> <w:endnoteReference w:id="2" /> </w:r> <w:r> <w:t>.</w:t> </w:r> </w:p> This text references the endnote in the document's endnotes part which has an id value of 2. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FtnEdnRef">
   <attribute name="customMarkFollows" type="ST_OnOff" use="optional"/>
   <attribute name="id" use="required" type="ST_DecimalNumber"/>
</complexType>
```

2.11.15 footnotes (Document Footnotes)

This element specifies the set of all footnotes in the document, including footnote separators and continuation notices. This element is the root node for the Footnotes part.

[*Example*: Consider the following example of the contents of the footnotes part:

```
<w:footnotes>
  <w:footnote w:type="separator" w:id="0">
    …
  </w:footnote >
  <w:footnote w:type="continuationSeparator" w:id="1">
    …
  </w:footnote>
  <w:footnote w:id="2">
    …
  </w:footnote>
</w:footnotes>
```

The footnotes part contains the definition for one normal footnote, as well as the separator and continuation separator footnotes for this document. *end example*]

Parent Elements
Root element of WordprocessingML Footnotes part

Child Elements	Subclause
footnote (Footnote Content)	§2.11.10

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Footnotes">
   <sequence maxOccurs="unbounded">
      <element name="footnote" type="CT_FtnEdn" minOccurs="0"/>
   </sequence>
</complexType>
```

2.11.16 noEndnote (Suppress Endnotes In Document)

This element specifies that all endnotes in this document shall not be displayed or printed. If this element is placed on any section break other than the first section break in the document, it shall be ignored.

If this element is omitted, endnotes shall not be suppressed in the current document.

[*Example*: Consider a document in which in the first section endnotes are marked to be hidden:

```
<w:sectPr>
  <w:noEndnote />
```

```
</w:sectPr>
```

In this example, this document will not display endnotes. *end example*]

Parent Elements
sectPr (§2.6.17); sectPr (§2.6.18); sectPr (§2.6.19)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.11.17 numFmt (Footnote Numbering Format)

This element specifies the numbering format which shall be used to determine the footnote or endnote reference mark value for all automatically numbered footnote and endnote reference marks (those without the suppressRef attribute set).

If this element is omitted, then the numbering format shall be assume to be decimal.

[*Example*: Consider the following footnote reference with the number format set to upper case letters:

This footnote numbering format is specified by the following WordprocessingML:

```
<w:footnotePr>
  <w:numFmt w:val="upperLetter" />
</w:footnotePr>
```

end example]

Parent Elements
footnotePr (§2.11.11); footnotePr (§2.11.12)

Attributes	Description
val (Numbering Format Type)	Specifies the number format that shall be used for all numbering in the parent object. [*Example*: A value of `lowerLetter` indicates that a consumer shall use lowercase letters for each number in this grouping: a,b,c… *end example*] The possible values for this attribute are defined by the ST_NumberFormat simple type (§2.18.66).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NumFmt">
   <attribute name="val" type="ST_NumberFormat" use="required"/>
</complexType>
```

2.11.18 numFmt (Endnote Numbering Format)

This element specifies the numbering format which shall be used to determine the footnote or endnote reference mark value for all automatically numbered footnote and endnote reference marks (those without the suppressRef attribute set).

If this element is omitted, then the numbering format shall be assume to be `decimal`.

[*Example*: Consider the following footnote reference with the number format set to upper case letters:

This footnote numbering format is specified by the following WordprocessingML:

```
<w:footnotePr>
  <w:numFmt w:val="upperLetter" />
</w:footnotePr>
```

end example]

Parent Elements
endnotePr (§2.11.4); endnotePr (§2.11.5)

Attributes	Description
val (Numbering Format Type)	Specifies the number format that shall be used for all numbering in the parent object. [*Example*: A value of `lowerLetter` indicates that a consumer shall use lowercase letters for each number in this grouping: a,b,c… *end example*] The possible values for this attribute are defined by the ST_NumberFormat simple type (§2.18.66).

The following XML Schema fragment defines the contents of this element:

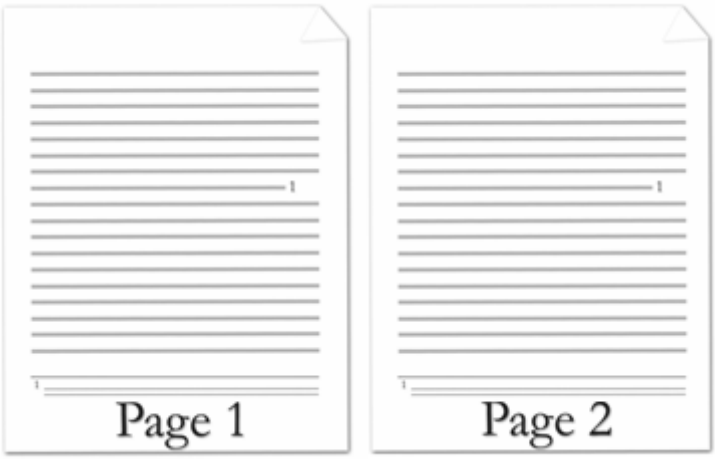
```
<complexType name="CT_NumFmt">
   <attribute name="val" type="ST_NumberFormat" use="required"/>
</complexType>
```

2.11.19 numRestart (Footnote and Endnote Numbering Restart Location)

This element specifies when all automatic numbering for the footnote or endnote reference marks shall be restarted. When restarted, the next automatically numbered footnote or endnote in the document (each type is handled independently) shall restart to the specified numStart value (§2.11.20).

If this element is omitted, then automatic numbering shall not be restarted between each page or section (a vlaue of `continuous`).

[*Example*: Consider the following two page document where the numbering shall be reset after each page to its starting value:



The footnote automatic restarting of the numbering is represented by the following WordprocessingML:

```
<w:footnotePr>
  <w:numRestart w:val="eachPage" />
</w:footnotePr>
```

end example]

Parent Elements
endnotePr (§2.11.4); endnotePr (§2.11.5); footnotePr (§2.11.11); footnotePr (§2.11.12)

Attributes	Description
val (Automatic Numbering Restart Value)	Specifies when the automatic numbering shall be restarted for the current set of footnotes or endnotes. [*Example*: Consider a WordprocessingML document where the numbering for its endnotes shall be restarted after each section shall be restarted after each page. This setting is represented by the following WordprocessingML:

Attributes	Description
	<w:footnotePr> <w:numRestart w:val="eachSect" /> </w:footnotePr> The val attribute value of eachSect specifies that numbering shal be restarted after each section. *end example*] The possible values for this attribute are defined by the ST_RestartNumber simple type (§2.18.81).

The following XML Schema fragment defines the contents of this element:

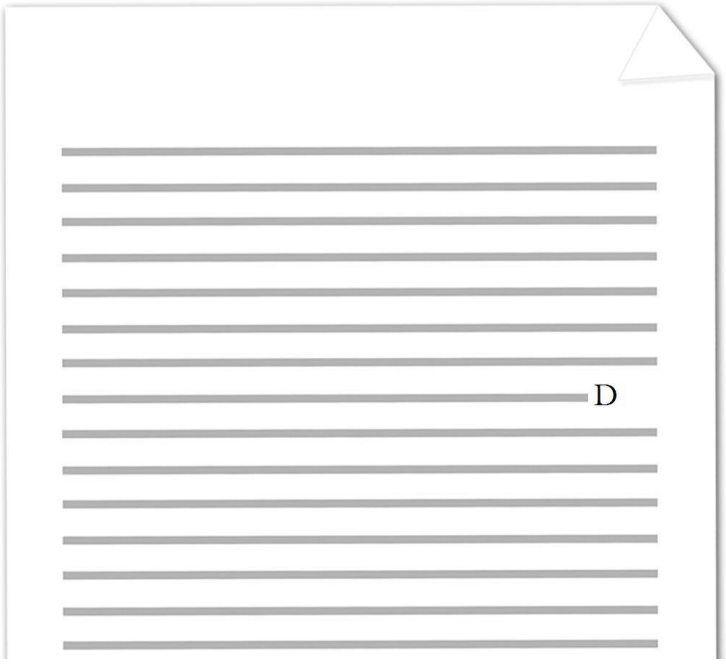
```
<complexType name="CT_NumRestart">
   <attribute name="val" type="ST_RestartNumber" use="required"/>
</complexType>
```

2.11.20 numStart (Footnote and Endnote Numbering Starting Value)

This element specifies the starting number or character for the first automatically numbered footnotes or endnote in the document, as well as the first automatically numbered footnotes after each restart point specified by the numRestart element (§2.11.19). This value shall be specified in decimal number units, then translated accordingly to the appropriate numbering format.

If this element is omitted, then the starting value shall be 1.

[*Example*: Consider the following footnote reference with the number format set to upper case letters and starting character set to D:



The number format is specified by the following WordprocessingML:

```
<w:footnotePr>
  <w:numFmt w:val="upperLetter" />
  <w:numStart w:val="4" />
</w:footnotePr>
```

Since D is the fourth letter in the alphabet, the starting character is set to 4. *end example*]

Parent Elements
endnotePr (§2.11.4); endnotePr (§2.11.5); footnotePr (§2.11.11); footnotePr (§2.11.12)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.11.21 pos (Footnote Placement)

This element specifies where footnotes shall be placed on the page when they are referenced by text in the current document.

If this element is present at the section level, then it shall be ignored.

If this element is omitted at the document level, then footnotes shall be located at the bottom of the current page.

[*Example*: Consider the following one page document, where the footnote appears beneath the text that it is referencing:

The footnote references the text reading `Some reference text.` and is represented by the following WordprocessingML:

```
<w:p>
  <w:r>
     <w:t>Some referenced text</w:t>
  </w:r>
  <w:r>
    <w:rPr>
      <w:rStyle w:val="FootnoteReference" />
    </w:rPr>
    <w:footnoteReference w:id="2" />
  </w:r>
</w:p>
```

Since the footnote location must be beneath the current text, the section properties shall be declared as follows:

```
<w:settings>
  …
  <w:footnotePr>
    <w:pos w:val="beneathText" />
  </w:footnotePr>
  …
</w:sectPr>
```

The footnote references the footnote in the footnotes part with an id attribute value equal to 2. Within the section properties of the document, the position of footnotes is specified to be beneath the page's text. *end example*]

Parent Elements
footnotePr (§2.11.11); footnotePr (§2.11.12)

Attributes	Description

val (Footnote Position Type)	Specifies the position of footnotes in the document.
	
	[*Example*: Consider a document in which footnotes shall be positioned beneath their text. The footnote properties for this document shall be declared as follows:
	
	`<w:sectPr>`
	`  <w:footnotePr>`
	`  <w:pos w:val="beneathText" />`
	`  </w:footnotePr>`
	`  …`
	`</w:sectPr>`
	
	The val attribute is beneathText, therefore the position of footnotes is specified to be beneath the page's text. *end example*]
	
	The possible values for this attribute are defined by the ST_FtnPos simple type (§2.18.38).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FtnPos">
   <attribute name="val" type="ST_FtnPos" use="required"/>
</complexType>
```

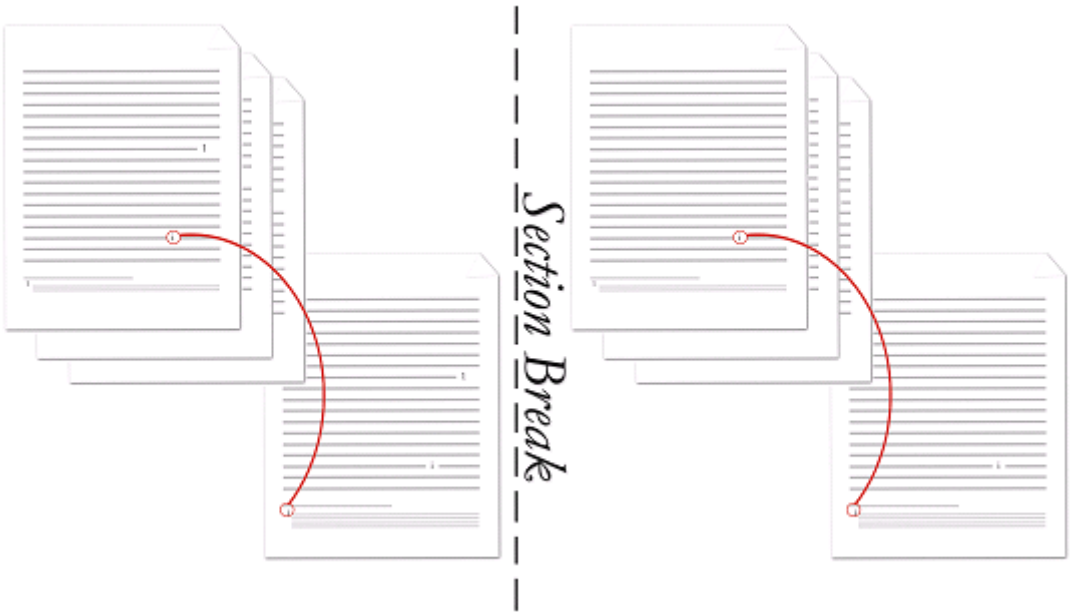
2.11.22 pos (Endnote Placement)

This element specifies where endnotes shall be placed on the page when they are referenced by text in the current document.

If this element is present at the section level, then it shall be ignored.

If this element is omitted at the document level, then endnotes shall be located at the end of the document.

[*Example*: Consider the following multi-page, multi-section document, where the endnote appears at the end of the section in which it is referenced:



The endnote setting is the same for all sections, and is represented by the following WordprocessingML at the document level:

```
<w:settings>
  …
  <w:endnotePr>
    <w:pos w:val="sectEnd" />
    …
  </w:endnotePr>
  …
</w:settings>
```

Within the properties of the document, the position of endnotes is specified to be at the end of each section. *end example*]

Parent Elements
endnotePr (§2.11.4); endnotePr (§2.11.5)

Attributes	Description
val (Endnote Position Type)	Specifies the position of endnotes on the parent section or the document. [*Example*: Consider a document in which endnotes shall be positioned at the end of the section. The section properties for this section shall be declared as follows: `<w:settings>`

Attributes	Description
	<w:endnotePr> <w:pos w:val="endSect" /> </w:endnotePr> … </w:settings> The val attribute is endSect, therefore the position of endnotes is specified to be at the end of the section. *end example*] The possible values for this attribute are defined by the ST_EdnPos simple type (§2.18.26).

The following XML Schema fragment defines the contents of this element:

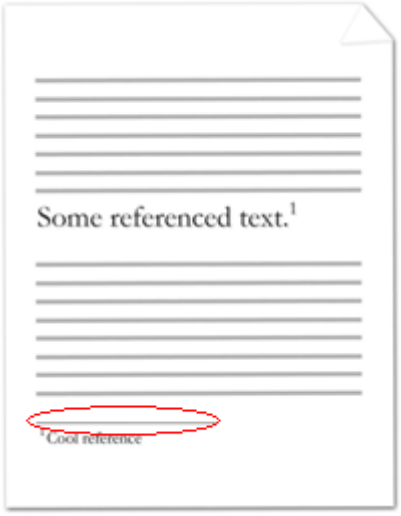
```
<complexType name="CT_EdnPos">
   <attribute name="val" type="ST_EdnPos" use="required"/>
</complexType>
```

2.11.23 separator (Footnote/Endnote Separator Mark)

This element specifies the presence of a separator mark within the current run. A *separator mark* is a horizontal line which spans part of the width text extents.

[*Note*: The separator mark is typically used within the context of separator footnotes or endnotes. These footnote and endnote types define the footnote/endnote used to separate the contents of the main document story from the contents of footnotes or endnotes on that page. *end note*]

[*Example*: Consider the following page in a document, where some text is referenced by a footnote that is located at the bottom of the current page (with the separator circled in red):



The line separating the document text from the normal footnotes is the footnote separator, and is represented by the following WordprocessingML:

```
<w:footnote w:type="separator" w:id="0">
  <w:p>
    <w:r>
      <w:separator />
    </w:r>
  </w:p>
</w:footnote>
```

In this example, the footnote has a content which consists of a single separator, which is displayed as a horizontal line across part of the text extents. *end example*]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

2.12 Glossary Document

Within a WordprocessingML file, the *glossary document* is a supplemental storage location for additional document content which shall travel with the document, but which shall not be displayed for printed as part of the main document until it is explicitly added to that document by deliberate action.

The glossary document shall also be afforded a separate instance of all of the relationships which are provided on the main document part - this means that the glossary document shall have its own style definitions, numbering definitions, comments, headers, footers, etc. within the WordprocessingML document.

[*Example*: Consider a document which shall include ten optional clauses that may be inserted through a user interface. It is clearly not desirable to have these ten clauses appear in the main document story's contents before they are explicitly inserted, therefore each of them may be stored in the glossary document and inserted via the user interface as needed. *end example*]

Within the glossary document, each distinct region of document content is referred to as a *glossary document entry*, and is defined via the docPart element (§2.12.5). These document parts may contain any block-level WordprocessingML element, and may also have a set of classifications and behaviors applied to them via the glossary document entry's properties.

[*Example*: Consider the following definition for the contents of a glossary document part within a WordprocessingML document:

```
<w:glossaryDocument>
  <w:docParts>
    <w:docPart>
      <w:docPartPr>
        …
      </w:docPartPr>
      <w:docPartBody>
        <w:p>
          <w:r>
            <w:t>Sample entry.</w:t>
          </w:r>
        </w:p>
      </w:docPartBody>
    </w:docPart>

    <w:docPart>
      …
    </w:docPart>
  </w:docParts>
</w:glossaryDocument>
```

The glossaryDocument element (§2.12.10) defines the contents of the glossary document part. Within the glossary document, each docPart element contains the definition for one glossary document entry: in this case, there are two entries in the glossary document, the first of which contains a single paragraph with a single run of text. *end example*]

2.12.1 behavior (Entry Insertion Behavior)

This element specifies a single behavior which shall be applied to the contents of the parent glossary document entry (§2.12.5) when it is added to the main document story of a WordprocessingML document. These behaviors shall be used to format the surrounding WordprocessingML around insertion, and do not require the presence of a user interface (i.e. applications without a user interface shall also utilize these settings).

[*Example*: Consider the WordprocessingML fragment for a glossary document entry containing a single run, defined as follows:

```
<w:docPart>
  <w:docPartPr>
    <w:behaviors>
      <w:behavior w:val="p"/>
    </w:behavior>
    …
  </w:docPartPr>
  <w:docPartBody>
    <w:p>
      <w:r>
        <w:t>Sample entry.</w:t>
      </w:r>
    </w:p>
  </w:docPartBody>
</w:docPart>
```

The behavior element has a value of p, which specifies that the contents of the parent glossary document entry shall be inserted in their own paragraph when they are added to the contents of a document. If the document content to which they are added is defined as follows (and the part is added between the two text runs):

```
<w:body>
  <w:p>
    <w:r>
      <w:t>After this text</w:t>
    </w:r>
    <w:r>
      <w:t>Before this text</w:t>
    </w:r>
  </w:p>
</w:body>
```

This setting specifies that although the part would normally be inserted between the two existing runs in the paragraph, that behavior shall ensure that the part is inserted into its own paragraph, resulting in the following WordprocessingML:

```
<w:body>
  <w:p>
    <w:r>
      <w:t>After this text</w:t>
    </w:r>
  </w:p>
  <w:p>
    <w:r>
      <w:t>Sample entry.</w:t>
    </w:r>
  </w:p>
  <w:p>
    <w:r>
      <w:t>Before this text</w:t>
    </w:r>
  </w:p>
</w:body>
```

end example]

Parent Elements
behaviors (§2.12.2)

Attributes	Description

val (Insertion Behavior Value)	Specifies the insertion behavior which shall be associated with the current glossary document entry.
	
	[*Example*: Consider the WordprocessingML fragment for a glossary document entry's properties, defined as follows:
	
	`<w:docPartPr>`
	`  <w:behaviors>`
	`  <w:behavior w:val="content"/>`
	`  </w:behavior>`
	`</w:docPartPr>`
	
	The val attribute value of `content` specifies that the insertion of this glossary document entry shall include only the content (the last paragraph in the part shall be merged into the current paragraph in the document). *end example*]
	
	The possible values for this attribute are defined by the ST_DocPartBehavior simple type (§2.18.19).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DocPartBehavior">
   <attribute name="val" use="required" type="ST_DocPartBehavior"/>
</complexType>
```

2.12.2 behaviors (Entry Insertion Behaviors)

This element specifies the set of behaviors which shall be applied to the contents of the parent glossary document entry (§2.12.5) when it is added to the main document story of a WordprocessingML document. Since multiple behaviors can be specified for a single part, the sum total of all behaviors shall be used to insert the parent entry into the contents of the WordprocessingML document.

[*Example*: Consider the WordprocessingML fragment for a glossary document entry containing a single run, defined as follows:

```
<w:docPart>
  <w:docPartPr>
    <w:behaviors>
      <w:behavior w:val="p" />
      <w:behavior w:val="pg" />
    </w:behavior>
    …
  </w:docPartPr>
  <w:docPartBody>
    <w:p>
      <w:r>
        <w:t>Sample entry.</w:t>
      </w:r>
    </w:p>
  </w:docPartBody>
</w:docPart>
```

The behaviors element contains the set of behaviors which shall be applied to this entry when it is inserted into the document, in this case:

- The entry shall be inserted into its own paragraph in the document
- The entry shall be inserted onto a new page in the document (i.e. it shall be preceded by a page break)

end example]

Parent Elements
docPartPr (§2.12.7)

Child Elements	Subclause

Child Elements	Subclause
behavior (Entry Insertion Behavior)	§2.12.1

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DocPartBehaviors">
   <choice>
      <element name="behavior" type="CT_DocPartBehavior" maxOccurs="unbounded"/>
   </choice>
</complexType>
```

2.12.3 category (Entry Categorization)

This element specifies the categorization for the parent glossary document entry. This categorization shall not imply any behaviors around the entry, and is only used to organize the set of glossary document entries within an application or user interface (i.e. to disambiguate between two entries with the same entry name (§2.12.13)).

[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:

```
<w:docPartPr>
  <w:category>
    …
  </w:category>
  …
<w:docPartPr>
```

The category element specifies the categorization applied to the current entry, for the purposes of classification or user interface sorting, for example. *end example*]

Parent Elements
docPartPr (§2.12.7)

Child Elements	Subclause
gallery (Gallery Associated With Entry)	§2.12.9
name (Category Associated With Entry)	§2.12.12

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DocPartCategory">
   <sequence>
      <element name="name" type="CT_String" minOccurs="1" maxOccurs="1"/>
      <element name="gallery" type="CT_DocPartGallery" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

2.12.4 description (Description for Entry)

This element specifies a description for the contents of this glossary document entry. This description may contain any string content, and allows the entry to have additional information contained within the definition for this glossary document entry. [Note: This description may be surfaced in a user interface, for example. *end note*]

[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:

```
<w:docPartPr>
  …
  <w:name w:val="Sample Entry" />
  <w:description w:val="This is an example of a glossary document entry for
example purposes." />
  …
<w:docPartPr>
```

The description element specifies that the long description associated with the parent entry shall be `This is an example of a glossary document entry for example purposes.` This value may be used as needed by an application, for example, to display in a user interface. *end example*]

Parent Elements
docPartPr (§2.12.7)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`

Attributes	Description
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.12.5 docPart (Glossary Document Entry)

This element specifies the details for a single glossary document entry contained in the document. This glossary document entry may consist of one or both of the following:

- The glossary document entry's properties, which define its name, categorization, and behaviors
- The glossary document entry's contents, which consists of one or more block-level elements of WordprocessingML content

Each of these two components is specified by one of the child elements of this element, as seen in the child elements table below.

[*Example*: Consider the following definition for the contents of a Glossary Document part within a WordprocessingML document:

```
<w:glossaryDocument>
  <w:docParts>
    <w:docPart>
      …
    </w:docPart>
    <w:docPart>
      …
    </w:docPart>
  </w:docParts>
</w:glossaryDocument>
```

The docPart element uniquely defines one glossary document entry within the glossary document part, therefore there are two unique entries stored in the current example of a glossary document part. *end example*]

Parent Elements
docParts (§2.12.8)

Child Elements	Subclause
docPartBody (Contents of Glossary Document Entry)	§2.12.6
docPartPr (Glossary Document Entry Properties)	§2.12.7

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DocPart">
   <sequence>
      <element name="docPartPr" type="CT_DocPartPr" minOccurs="0"/>
      <element name="docPartBody" type="CT_Body" minOccurs="0"/>
   </sequence>
</complexType>
```

2.12.6 docPartBody (Contents of Glossary Document Entry)

This element specifies the contents of the parent glossary document entry (§2.12.5). These contents shall consist of one or more block-level elements, analogous to the body element (§2.2.2) of the main document story for the current document.

[*Example*: Consider the WordprocessingML fragment for a glossary document entry containing a single run, defined as follows:

```
<w:docPart>
  …
  <w:docPartBody>
    <w:p>
      <w:r>
        <w:t>Sample entry.</w:t>
      </w:r>
    </w:p>
  </w:docPartBody>
</w:docPart>
```

The docPartBody element specifies the block-level elements which comprise the contents of the current glossary document entry, in this case, a single paragraph using the p element (§2.3.1.22). *end example*]

When the contents of a glossary document entry are added to a document, the styles, numbering definitions, and all other related parts for this entry shall be taken from the relationships from the Glossary Document part and not from the main document part. These references shall be moved to their main document equivalents when the entry is added to the document.

When the part is inserted, it shall be inserted as though its last paragraph mark does not exist (the content of the final paragraph mark shall be merged with the contents of the paragraph into which this entry is being added).

[*Example*: Consider the WordprocessingML fragment for a glossary document entry containing a single run, defined as follows:

```
<w:docPart>
  <w:docPartPr>
    <w:behaviors>
      <w:behavior w:val="p"/>
    </w:behavior>
    …
  </w:docPartPr>
  <w:docPartBody>
    <w:p>
      <w:r>
        <w:t>Sample entry.</w:t>
      </w:r>
    </w:p>
  </w:docPartBody>
</w:docPart>
```

If this entry is inserted into document content to which is defined as follows (and the part is added between the two text runs):

```
<w:body>
  <w:p>
    <w:r>
      <w:t>After this text</w:t>
    </w:r>
    <w:r>
      <w:t>Before this text</w:t>
    </w:r>
  </w:p>
</w:body>
```

This entry has only a single paragraph, which is removed before insertion, and barring any special insertion behaviors (§2.12.2), only the text run is inserted, resulting in the following WordprocessingML:

```
<w:body>
  <w:p>
    <w:r>
      <w:t>After this text</w:t>
    </w:r>
    <w:r>
      <w:t>Sample entry.</w:t>
    </w:r>
    <w:r>
      <w:t>Before this text</w:t>
    </w:r>
  </w:p>
</w:body>
```

end example]

Parent Elements
docPart (§2.12.5)

Child Elements	Subclause
altChunk (Anchor for Imported External Content)	§2.17.3.1
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Block-Level Custom XML Element)	§2.5.1.6
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
del (Deleted Run Content)	§2.13.5.12
ins (Inserted Run Content)	§2.13.5.20
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23

Child Elements	Subclause
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
p (Paragraph)	§2.3.1.22
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
sdt (Block-Level Structured Document Tag)	§2.5.2.30
sectPr (Document Final Section Properties)	§2.6.18
tbl (Table)	§2.4.36

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Body">
   <sequence>
      <group ref="EG_BlockLevelElts" minOccurs="0" maxOccurs="unbounded"/>
      <element name="sectPr" minOccurs="0" maxOccurs="1" type="CT_SectPr"/>
   </sequence>
</complexType>
```

2.12.7 docPartPr (Glossary Document Entry Properties)

This element specifies the set of properties which shall be applied to the parent glossary document entry. These properties define its name, categorization, and behaviors.

[*Example*: Consider the WordprocessingML fragment for a glossary document entry containing a single run, defined as follows:

```
<w:docPart>
  <w:docPartPr>
    <w:name w:val="Sample Entry" />
    …
  </w:docPartPr>
  …
</w:docPart>
```

The docPartPr element specifies the set of properties which have been specified for the parent glossary document entry, the only one visible above being the entry's name of `Sample Entry`. *end example*]

Parent Elements

Parent Elements
docPart (§2.12.5)

Child Elements	Subclause
behaviors (Entry Insertion Behaviors)	§2.12.2
category (Entry Categorization)	§2.12.3
description (Description for Entry)	§2.12.4
guid (Entry ID)	§2.12.11
name (Entry Name)	§2.12.13
style (Associated Paragraph Style Name)	§2.12.14
types (Entry Types)	§2.12.16

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DocPartPr">
   <choice maxOccurs="unbounded">
      <element name="name" type="CT_DocPartName" minOccurs="1"/>
      <element name="style" type="CT_String"/>
      <element name="category" type="CT_DocPartCategory"/>
      <element name="types" type="CT_DocPartTypes"/>
      <element name="behaviors" type="CT_DocPartBehaviors"/>
      <element name="description" type="CT_String"/>
      <element name="guid" type="CT_Guid"/>
   </choice>
</complexType>
```

2.12.8 docParts (List of Glossary Document Entries)

This element specifies the collection of glossary document entries which are stored in the current Glossary Document part.

[*Example*: Consider the following definition for the contents of a glossary document part within a WordprocessingML document:

```
<w:glossaryDocument>
  <w:docParts>
    <w:docPart>
      …
    </w:docPart>
    <w:docPart>
      …
    </w:docPart>
  </w:docParts>
</w:glossaryDocument>
```

The docParts element defines the set of entries which are stored in the glossary document part. *end example*]

Parent Elements
glossaryDocument (§2.12.10)

Child Elements	Subclause
docPart (Glossary Document Entry)	§2.12.5

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DocParts">
   <choice>
      <element name="docPart" type="CT_DocPart" minOccurs="1" maxOccurs="unbounded"/>
   </choice>
</complexType>
```

2.12.9 gallery (Gallery Associated With Entry)

This element specifies the predefined gallery into which the current glossary document part shall be classified. This classification, although its enumeration values may be interpreted to imply semantics around the contents of the parent glossary document entry, shall only be used to classify and sort this entry (via an application or a user interface).

[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:

```
<w:docPartPr>
  <w:category>
    <w:gallery w:val="coverPg" />
    <w:name w:val="Internal Memo Covers" />
  </w:category>
  …
<w:docPartPr>
```

The gallery element with a value of coverPg specifies that the gallery categorization applied to the current entry, for the purposes of classification or user interface sorting, puts this entry into the Cover Pages classification. *end example*]

Parent Elements
category (§2.12.3)

Attributes	Description
val (Gallery Value)	Specifies the classification of gallery which shall be associated with the parent glossary document entry.

Attributes	Description
	[*Example*: Consider the following WordprocessingML fragment for a single glossary document entry: `<w:gallery w:val="custom1" />` The val attribute with a value of `custom1` specifies that the gallery categorization applied to the current entry is the `Custom 1` classification. *end example*] The possible values for this attribute are defined by the ST_DocPartGallery simple type (§2.18.20).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DocPartGallery">
   <attribute name="val" type="ST_DocPartGallery" use="required"/>
</complexType>
```

2.12.10 glossaryDocument (Glossary Document Root Element)

This element specifies the root element for a glossary document part within a WordprocessingML document. A glossary document is an supplementary document story in a WordprocessingML that shall be afforded all of the relationships of the Main Document part, such as:

- Style definitions
- Numbering definitions
- Comments
- Headers/footers
- Etc.

The entries stored in this part shall have all of its implicit relationships target these parts, rather than their analogues stored off of the main document part.

[*Example*: Consider the following definition for the contents of a glossary document part within a WordprocessingML document:

```
<w:glossaryDocument>
  <w:docParts>
    <w:docPart>
      …
    </w:docPart>
    <w:docPart>
      …
    </w:docPart>
  </w:docParts>
</w:glossaryDocument>
```

The glossaryDocument element defines the contents of the glossary document part. *end example*]

Parent Elements
Root element of WordprocessingML Glossary Document part

Child Elements	Subclause
background (Document Background)	§2.2.1
docParts (List of Glossary Document Entries)	§2.12.8

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GlossaryDocument">
   <complexContent>
      <extension base="CT_DocumentBase">
         <sequence>
            <element name="docParts" type="CT_DocParts" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.12.11 guid (Entry ID)

This element specifies a unique identifier (specified using a 128-bit GUID stored on the val attribute) that uniquely identifies this document building block.[*Note*: This unique identifier may be used by an application to uniquely reference a single building block regardless of different naming, for example when the same part has different names for localization purposes. *end note*]

[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:

```
<w:docPartPr>
  …
  <w:guid w:val="{00000000-5BD2-4BC8-9F70-7020E1357FB2}" />
  …
<w:docPartPr>
```

The guid element specifies that the unique identifier associated with the parent entry shall be {00000000-5BD2-4BC8-9F70-7020E1357FB2}. This value may be used as needed by an application, for example, to uniquely identify a part regardless of its name. *end example*]

Parent Elements
docPartPr (§2.12.7)

Attributes	Description
val (GUID Value)	Specifies a 128-bit globally unique identifier (GUID) value as defined by the simple type referenced below. The contents of this GUID shall be interpreted based on the context of the parent XML element. If this attribute is omitted, its value shall be assumed to be null (i.e. no GUID shall be associated with the parent XML element). [*Example*: Consider the following WordprocessingML element: `<… w:val="{6A9B8B6F-5BD2-4BC8-9F70-7020E1357FB2}"/>` The val attribute value of `{6A9B8B6F-5BD2-4BC8-9F70-7020E1357FB2}` shall be associated with the context of the parent XML element. *end example*] The possible values for this attribute are defined by the ST_Guid simple type (§2.18.39).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Guid">
   <attribute name="val" type="ST_Guid"/>
</complexType>
```

2.12.12 name (Category Associated With Entry)

This element specifies the category into which the current glossary document part shall be classified. This classification may consist of any string value as determined by its contents, and shall only be used to classify and sort this entry (via an application or a user interface).

[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:

```
<w:docPartPr>
  <w:category>
    <w:gallery w:val="coverPg" />
    <w:name w:val="Internal Memo Covers" />
  </w:category>
  …
<w:docPartPr>
```

The name element with a value of `Internal Memo Covers` specifies that the category grouping applied to the current entry, for the purposes of classification or user interface sorting, puts this entry into the `Internal Memo Covers` classification. This category may be used as desired. *end example*]

Parent Elements
category (§2.12.3)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.12.13 name (Entry Name)

This element specifies a name for the contents of this glossary document entry. This name may contain any string content, and allows the entry to have a friendly identifier contained within the definition for this glossary document entry. [*Note*: This name may be surfaced in a user interface, for example. *end note*]

[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:

```
<w:docPartPr>
  …
  <w:name w:val="Sample Entry" />
  <w:description w:val="This is an example of a glossary document entry for
example purposes." />
  …
<w:docPartPr>
```

The name element specifies that the friendly name associated with the parent entry shall be `Sample Entry`. This value may be used as needed by an application, for example, to display in a user interface. *end example*]

Parent Elements
docPartPr (§2.12.7)

Attributes	Description
decorated (Built-In Entry)	Specifies that the name for the current entry is a built-in entry which should not be displayed in the user interface. [*Note*: This information may be used by an application as needed, for example, to disambiguate an entry from one with the same name, ensuring that the built-in entry can be uniquely identified by the application. *end note*] If this attribute is omitted, its value shall be assumed to be `false`. [*Example*: Consider the following WordprocessingML fragment for the name of a single glossary document entry: `<w:name w:decorated="true" w:val=":-)" />` The decorated attribute specifies that the parent entry is a built-in entry, and shall be treated as such. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
val (Name Value)	Specifies a string value which contains the name of the current glossary document entry. [*Example*: Consider the following WordprocessingML fragment for the name of a single glossary document entry: `<w:name w:val="Sample Entry" />` The val attribute specifies that the name of the parent entry is `Sample Entry`. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DocPartName">
   <attribute name="val" type="ST_String" use="required"/>
   <attribute name="decorated" type="ST_OnOff" use="optional"/>
</complexType>
```

2.12.14 style (Associated Paragraph Style Name)

This element specifies the style ID for a paragraph style which shall be associated with the current glossary document entry. This paragraph style associated shall not imply anything about the formatting or content of the glossary document entry, and shall only be used to filter and/or sort this entry (via an application or a user interface). [*Note*: One example of the level of classification offered by this element is to only show it as available when the formatting of the paragraph matches the specified style. *end note*]

[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:

```
<w:docPartPr>
  <w:style w:val="Heading1" />
  …
<w:docPartPr>
```

The style element with a val attribute value of `Heading1` specifies that the paragraph style associated with the current glossary document entry shall be the style whose style ID is equal to `Heading1`. *end example*]

Parent Elements
docPartPr (§2.12.7)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`

Attributes	Description
	<w:alias w:val="SDT Title Example" /> ... </w:sdtPr> In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.12.15 type (Entry Type)

This element specifies a single type which shall be applied to the properties of the parent glossary document entry (§2.12.5). Each of these types may, based on their values, influence the visibility and behavior of the parent glossary document entry as defined by the associated simple type information.

[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:

```
<w:docPartPr>
  <w:types>
    <w:type w:val="bbPlcHdr" />
  </w:types>
  …
<w:docPartPr>
```

The type element with a value of bbPlcHdr specifies that the parent glossary document entry shall be treated as if it was the placeholder text for one or more structured document tags in the document. *end example*]

Parent Elements
types (§2.12.16)

Attributes	Description
val (Type Value)	Specifies the value for the current entry type. [*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry: <w:type w:val="bbPlcHdr" />

Attributes	Description
	The val attribute value of bbPlcHdr specifies that the parent glossary document entry shall be treated as if it was the placeholder text for one or more structured document tags in the document. *end example*] The possible values for this attribute are defined by the ST_DocPartType simple type (§2.18.21).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DocPartType">
   <attribute name="val" use="required" type="ST_DocPartType"/>
</complexType>
```

2.12.16 types (Entry Types)

This element specifies the set of types which shall be applied to the properties of the parent glossary document entry (§2.12.5). Each of these types may, based on their values, influence the visibility and behavior of the parent glossary document entry.

[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:

```
<w:docPartPr>
  <w:types>
    …
  </w:types>
  …
<w:docPartPr>
```

The types element specifies the set of entry types which shall be associated with the parent glossary document entry. *end example*]

Parent Elements
docPartPr (§2.12.7)

Child Elements	Subclause
type (Entry Type)	§2.12.15

Attributes	Description
all (Entry Is Of All Types)	Specifies that the current glossary document is all types. This attribute shall override any information specified as child elements of this element and shall ensure that the current entry is associated with all available types.

Attributes	Description

	If this attribute is omitted, then its default value shall be assumed to be `false`.
	
	[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:
	
	`<w:docPartPr>`
	`  <w:types w:all="true">`
	`  <w:type w:val="autoExp" />`
	`  </w:types>`
	`  …`
	`<w:docPartPr>`
	
	The types element contains a single entry type definition, but because the all attribute is present with a value of `true`, that type is augmented to place the parent entry into all possible types. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DocPartTypes">
    <choice>
        <element name="type" type="CT_DocPartType" maxOccurs="unbounded"/>
    </choice>
    <attribute name="all" type="ST_OnOff" use="optional"/>
</complexType>
```

2.13 Annotations

Within a WordprocessingML document, *annotations* refer to various types of supplementary markup which may be stored inside or around a region of text within the document's contents. [*Example*: The types of supplementary information stored within a document may include: comments, revisions, spelling and/or grammatical errors, bookmark information and optional editing permissions. *end example*]

Within a document's contents, annotations are stored in one of three different methods:

- Inline
- "Cross Structure"
- Properties

These three forms are needed in order to maintain compatibility with both the legacy annotations functionality of current word processing applications and the requirements of an XML-based format (i.e. wellformedness of the resulting XML markup). These three forms are referenced within the individual annotation types described in the following sub clauses.

2.13.1 Inline Annotations

Inline annotations describe all annotations which do not require special handling in order to maintain the XML wellformedness requirements of the resulting WordprocessingML output. In these cases, a single XML element shall encapsulate the entire contents of the document content which is being annotated.

[*Example*: Consider the following WordprocessingML markup for a paragraph which reads `The quick brown fox jumps over the jet lagged dog.`, where `jet lagged` replaced the previous text `lazy` when the editing application was tracking revisions:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">The quick brown fox jumps over the </w:t>
  </w:r>
  <w:del … >
    <w:r>
      <w:delText>lazy</w:delText>
    </w:r>
  </w:del>
  <w:ins … >
    <w:r>
      <w:t>jet lagged</w:t>
    </w:r>
  </w:ins>
  <w:r>
    <w:t xml:space="preserve"> dog.</w:t>
  </w:r>
</w:p>
```

The del and ins elements (§2.13.5.12; §2.13.5.20) each fully encapsulate the extent of their respective annotations (a marked deletion and insertion, respectively), as they are inline annotations. *end example*]

2.13.2 "Cross Structure" Annotations

"Cross structure" annotations describe the class of annotations which can span portions of WordprocessingML markup [*Example*: Cross structure annotations may span parts of multiple paragraphs, one half of a custom XML markup element's contents, etc. *end example*]. In these cases, the annotation's region is delimited by two elements: a start element and an end element. These two elements mark the start and end points of the annotated content, but do not contain it. The pairing of the start and end marker are linked via a common value for their id attributes.

[*Example*: Consider the following WordprocessingML markup for two paragraphs, each reading `Example Text`, where a bookmark has been added spanning the second word in paragraph one and the first word in paragraph two:

```
<w:p>
  <w:r>
    <w:t>Example</w:t>
  </w:r>
  <w:bookmarkStart w:id="0" w:name="sampleBookmark" />
  <w:r>
    <w:t xml:space="preserve"> text.</w:t>
  </w:r>
</w:p>
<w:p>
  <w:r>
    <w:t>Example</w:t>
  </w:r>
  <w:bookmarkEnd w:id="0" />
  <w:r>
    <w:t xml:space="preserve"> text.</w:t>
  </w:r>
</w:p>
```

The bookmarkStart and bookmarkEnd elements (§2.13.6.2; §2.13.6.1) specify the location where the bookmark starts and ends, but cannot contain it because it spans part of two paragraphs. They are part of one group because the id attribute value specifies `0` for both. *end example*]

2.13.3 Property Annotations

Property annotations describe the class of annotations which are stored as a property on an object [*Example*: Property annotations may appear on paragraph properties, run properties, table rows, etc. *end example*] In these cases, the annotation's semantics are defined by the property, as they can affect content and/or formatting.

[*Example*: Consider the following WordprocessingML markup for a paragraph reading `Example Text`, where the first word had the bold property applied when the editing application was tracking revisions:

```
<w:p>
  <w:r>
    <w:rPr>
      <w:b/>
      <w:rPrChange … >
        <w:rPr/>
      </w:rPrChange>
    </w:rPr>
    <w:t>Example</w:t>
  </w:r>
  <w:r>
    <w:t xml:space="preserve"> text.</w:t>
  </w:r>
</w:p>
```

The rPrChange element (§2.13.5.32; §2.13.5.33) contains the set of previously applied revision properties associated with a particular author at a particular time. It is stored itself as a property on the parent run which was modified. *end example*]

2.13.4 Comments

Comments describe annotations which are anchored to a region of document content, but which contain an arbitrary amount of block-level content stored in their own separate document stories. Within a WordprocessingML document, comments are stored in a separate Comments part within the document package.

A comment in a WordprocessingML document is divided into two components:

- The comment anchor (the text on which the comment applies)
- The comment content (the contents of the comment)

The *comment anchor* is the cross structure annotation which defines the region of text on which the comment in anchored. The *comment content* is the content stored in the comments part which contains the actual content of the comment.

[*Example*: Consider a paragraph in a WordprocessingML document whose second word is annotated with a comment:

The first component to this comment is the document content which defines the extents of the comment and references the specific comment in the comments part:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">Some </w:t>
  </w:r>
  <w:commentRangeStart w:id="0" />
  <w:r>
    <w:t>text.</w:t>
  </w:r>
  <w:commentRangeEnd w:id="0" />
  <w:r>
    <w:commentReference w:id="0" />
  </w:r>
</w:p>
```

The commentRangeStart and commentRangeEnd elements (§2.13.4.4; §2.13.4.3) delimit the run content to which the comment with an id of `0` applies (in this case, the single run of text). The following commentReference element (§2.13.4.5) links the preceding run content with a comment in the comments part with an id of `0`. Without all three of these elements, the range and comment cannot be linked (although the first two elements are optional, in which case the comment shall be anchored at the comment reference mark)

The second component to this comment is the comment content which defines the text in the comment:

```
<w:comment w:id="0" w:author="Joe Smith" w:date="2006-04-06T13:50:00Z"
w:initials="User">
  <w:p>
    <w:pPr>
      <w:pStyle w:val="CommentText" />
    </w:pPr>
    <w:r>
      <w:rPr>
        <w:rStyle w:val="CommentReference" />
      </w:rPr>
      <w:annotationRef />
    </w:r>
    <w:r>
      <w:t>comment</w:t>
    </w:r>
  </w:p>
</w:comment>
```

In this example, the comment specifies that it was inserted by author `Joe Smith` with the initials `User` via the author and date attributes. It is linked to the run content via the id attribute, which matches the value of `0` specified using the commentReference element above. The block-level content of the comment specifies that

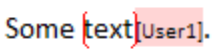
its text is `comment` and the style of the comment content is based off of the character style with the name `CommentReference`. *end example*]

2.13.4.1 annotationRef (Comment Information Block)

This element specifies the presence of an annotation reference mark at the current location in the comment. An *annotation reference mark* is an information block that represents the metadata about the current comment within the document. This annotation reference mark should typically consist of the initials and a unique integer associated with its position in the document, but may be displayed in any desired format.

If this element is omitted from a single comment's contents, then an annotation reference mark may be added at the start of the comment in reading order (right in a right-to-left paragraph or left in a left-to-right paragraph). As well, an annotation reference mark may be relocated as desired within a comment's content.

[*Example*: Consider a document with text with an annotated comment as follows:



This comment is represented as the following WordprocessingML fragment:

```
<w:comment … w:initials="User">
  <w:p>
    <w:r>
      <w:annotationRef />
    </w:r>
    …
  </w:p>
</w:comment>
```

The annotationRef element specifies that the comment shall start with an annotation reference mark. In this example, this mark is displayed as a combination of the user initial, `User`, and a unique sequential number, `1`. *end example*]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

The following XML Schema fragment defines the contents of this element:

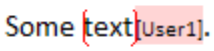
```
<complexType name="CT_Empty"/>
```

2.13.4.2 comment (Comment Content)

This element specifies the content of a single comment stored in the Comments part of a WordprocessingML document.

If a comment is not referenced by document content via a matching id attribute on a valid use of the commentReference element (§2.13.4.5), then it may be ignored when loading the document. If more than one comment shares the same value for the id attribute, then only one comment shall be loaded and the others may be ignored.

[*Example*: Consider a document with text with an annotated comment as follows:



This comment is represented as the following WordprocessingML fragment:

```
<w:comment w:id="1" w:initials="User">
  …
</w:comment>
```

The comment element specifies the presence of a single comment within the comments part. *end example*]

Parent Elements
comments (§2.13.4.6)

Child Elements	Subclause
altChunk (Anchor for Imported External Content)	§2.17.3.1
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Block-Level Custom XML Element)	§2.5.1.6
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
del (Deleted Run Content)	§2.13.5.12
ins (Inserted Run Content)	§2.13.5.20
moveFrom (Move Source Run Content)	§2.13.5.21

Child Elements	Subclause
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
p (Paragraph)	§2.3.1.22
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
sdt (Block-Level Structured Document Tag)	§2.5.2.30
tbl (Table)	§2.4.36

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`…`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard. If this attribute is omitted, then no date information shall be associated with the parent annotation type. [*Example*: Consider a comment represented using the following WordprocessingML fragment:

Attributes	Description
	<w:… w:id="1" w:date="2006-01-01T10:00:00"> … </w:…> The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*] The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).
id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element. If this attribute is omitted, then the document is non-conformant. [*Example*: Consider an annotation represented using the following WordprocessingML fragment: <w:… w:id="1" … > … </w:…> The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
initials (Initials of Comment Author)	Specifies the initials of the author of the current comment. This information may be used to format and present the associated comment information block (§2.13.4.1), or in any user interface supported by an application. If this attribute is omitted, then no author shall be associated with the current comment in the document. [*Example*: Consider a comment represented using the following WordprocessingML fragment: <w:comment w:id="1" w:initials="KB"> … </w:comment> The intials attribute specifies that the initials of the author of the current comment are KB, which may be used as desired. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Comment">
    <complexContent>
        <extension base="CT_TrackChange">
            <sequence>
                <group ref="EG_BlockLevelElts" minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
            <attribute name="initials" type="ST_String" use="optional"/>
        </extension>
    </complexContent>
</complexType>
```

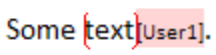
2.13.4.3 commentRangeEnd (Comment Anchor Range End)

This element specifies the end of the range around which a comment is anchored in the content of the WordprocessingML document. The id attribute on this element shall be used to link the corresponding comment anchor range start element and comment reference.

The following restrictions shall be applied to this element:

- If this element occurs without a corresponding commentRangeStart element (§2.13.4.4) with a matching id attribute value, then it shall be considered the single anchor point for the associated comment reference.
- If this element appears without a corresponding commentReference element (§2.13.4.5) in the current document story with a matching id attribute value, then it shall be ignored and the comment has no associated range.
- If this element appears in a comment content story (§2.13.4.2), then it may be ignored.

[*Example*: Consider a paragraph in a WordprocessingML document whose second word is annotated with a comment:



The WordprocessingML fragment for this comment is defined as follows:

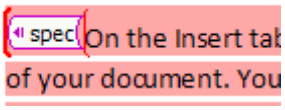
```
<w:p>
  <w:r>
    <w:t xml:space="preserve">Some </w:t>
  </w:r>
  <w:commentRangeStart w:id="0" />
  <w:r>
    <w:t>text.</w:t>
  </w:r>
  <w:commentRangeEnd w:id="0" />
  <w:r>
    <w:commentReference w:id="0" />
  </w:r>
</w:p>
```

The commentRangeEnd element specifies that the end of the comment range for the comment with an id of `0` is after the end of the run containing the word `text`. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32); endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Attributes	Description
displacedByCustomXml (Annotation Marker Relocated For Custom XML Markup)	Specifies that the parent annotation's placement shall be directly linked with the location of the physical presentation of a custom XML element in the document. This element only has an effect when the custom XML element is block-level (i.e. surrounds an entire paragraph), as in this scenario the logical and physical placement of the annotation and custom XML element may differ. Specifically, in this case, the custom XML is presented *around* the block-level object it encloses (the paragraph, table, table row, or table cell), but is physically represented within that same object (i.e. within the paragraph, table, table row or table cell). This requirement stems from the fact that there is no location for the location of the annotation within the document at its logical location (around a table, for example). If this element is omitted, then the annotation shall be anchored inside of all block-level custom XML elements in the paragraph. If this element is present, but no block-level custom XML tag is located at the position it specifies (before or after), then it shall be ignored.

Attributes	Description

	[*Example*: Consider a paragraph with block level custom XML markup and two comment anchor annotations (one before and one after the custom XML element's physical representation), as follows:
	
	 
	
	Since all three of these items are around the entire paragraph, they are stored outside of the paragraph. However, in order to ensure that their relative positions are stored correctly, any annotation which shall be displaced by the physical custom XML element specifies this information, resulting in the following WordprocessingML:
	
	`<w:commentRangeStart w:id="0" />`
	`<w:commentRangeStart w:id="1" w:displaced byCustomXml="next" />`
	`<w:customXml w:element="spec" … />`
	`<w:p>`
	`…`
	`</w:p>`
	
	The displacedByCustomXml attribute specifies that even though all three of these items are around the paragraph and will be moved inside the paragraph to be represented physically, the comment with ID 0 shall be inside the custom XML, but the comment with ID 1 shall be displaced to stay outside of the relative location of the next custom XML element (the spec element). *end example*]
	
	The possible values for this attribute are defined by the ST_DisplacedByCustomXml simple type (§2.18.17).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MarkupRange">
   <complexContent>
      <extension base="CT_Markup">
         <attribute name="displacedByCustomXml" type="ST_DisplacedByCustomXml" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

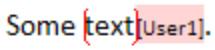
2.13.4.4 commentRangeStart (Comment Anchor Range Start)

This element specifies the start of the range around which a comment is anchored in the content of the WordprocessingML document. The id attribute on this element shall be used to link the corresponding comment anchor range end element and comment reference.

The following restrictions shall be applied to this element:

- If this element occurs without a corresponding commentRangeEnd element (§2.13.4.3) with a matching id attribute value, then it shall be considered the single anchor point for the associated comment reference.
- If this element appears without a corresponding commentReference element (§2.13.4.5) in the current document story with a matching id attribute value, then it shall be ignored and the comment content has no associated range.
- If this element appears in a comment content story (§2.13.4.2), then it may be ignored.

[*Example*: Consider a paragraph in a WordprocessingML document whose second word is annotated with a comment:



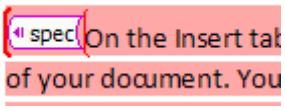
The WordprocessingML fragment for this comment is defined as follows:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">Some </w:t>
  </w:r>
  <w:commentRangeStart w:id="0" />
  <w:r>
    <w:t>text.</w:t>
  </w:r>
  <w:commentRangeEnd w:id="0" />
  <w:r>
    <w:commentReference w:id="0" />
  </w:r>
</w:p>
```

The commentRangeStart element specifies that the start of the comment range for the comment with an id of `0` is after the end of the run containing the word `Some`. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32); endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Attributes	Description

displacedByCustomXml (Annotation Marker Relocated For Custom XML Markup)	Specifies that the parent annotation's placement shall be directly linked with the location of the physical presentation of a custom XML element in the document. This element only has an effect when the custom XML element is block-level (i.e. surrounds an entire paragraph), as in this scenario the logical and physical placement of the annotation and custom XML element may differ.
	
	Specifically, in this case, the custom XML is presented *around* the block-level object it encloses (the paragraph, table, table row, or table cell), but is physically represented within that same object (i.e. within the paragraph, table, table row or table cell). This requirement stems from the fact that there is no location for the location of the annotation within the document at its logical location (around a table, for example).
	
	If this element is omitted, then the annotation shall be anchored inside of all block-level custom XML elements in the paragraph. If this element is present, but no block-level custom XML tag is located at the position it specifies (before or after), then it shall be ignored.
	
	[*Example*: Consider a paragraph with block level custom XML markup and two comment anchor annotations (one before and one after the custom XML element's physical representation), as follows:
	
	 
	
	Since all three of these items are around the entire paragraph, they are stored outside of the paragraph. However, in order to ensure that their relative positions are stored correctly, any annotation which shall be displaced by the physical custom XML element specifies this information, resulting in the following WordprocessingML:
	
	`<w:commentRangeStart w:id="0" />`
	`<w:commentRangeStart w:id="1" w:displaced byCustomXml="next" />`

Attributes	Description
	<w:customXml w:element="spec" … /> <w:p> … </w:p> The displacedByCustomXml attribute specifies that even though all three of these items are around the paragraph and will be moved inside the paragraph to be represented physically, the comment with ID 0 shall be inside the custom XML, but the comment with ID 1 shall be displaced to stay outside of the relative location of the next custom XML element (the spec element). *end example*] The possible values for this attribute are defined by the ST_DisplacedByCustomXml simple type (§2.18.17).
id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element. If this attribute is omitted, then the document is non-conformant. [*Example*: Consider an annotation represented using the following WordprocessingML fragment: <w:… w:id="1" … > … </w:…> The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MarkupRange">
   <complexContent>
      <extension base="CT_Markup">
         <attribute name="displacedByCustomXml" type="ST_DisplacedByCustomXml" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

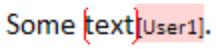
2.13.4.5 commentReference (Comment Content Reference Mark)

This element specifies the presence of a comment content reference mark, which links the comment content (§2.13.4.2) with the contents of a document story. This link is established by matching the comment whose id attribute matches the id attribute on this element. The resulting comment is anchored to the range with comment range elements with the same id attribute values (if present) as follows:

- If either or both of the commentRangeStart and commentRangeEnd elements (§2.13.4.4; §2.13.4.3) are present, then the comment reference shall anchor the comment to the resulting range.
- If neither element is present, then the comment reference shall anchor the comment to its current location.

If this element appears in a comment content story (§2.13.4.2), then it may be ignored. If no comment exists with an id attribute which matches the id attribute on this element, then this document is non-conformant.

[*Example*: Consider a paragraph in a WordprocessingML document whose second word is annotated with a comment:



The WordprocessingML fragment for this comment is defined as follows:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">Some </w:t>
  </w:r>
  <w:commentRangeStart w:id="0" />
  <w:r>
    <w:t>text.</w:t>
  </w:r>
  <w:commentRangeEnd w:id="0" />
  <w:r>
    <w:commentReference w:id="0" />
  </w:r>
</w:p>
```

The commentReference element specifies that the associated comment in the comments part shall be the comment whose id attribute value is 0. As well, since a start and end marker exist with a matching ID, this comment is anchored to that region of the document. *end example*]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

Attributes	Description
id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element. If this attribute is omitted, then the document is non-conformant.

Attributes	Description

	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Markup">
   <attribute name="id" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.13.4.6 comments (Comments Collection)

This element specifies all of the comments defined in the current document. It is the root element of the Comments part of a WordprocessingML document.

[*Example*: Consider the following WordprocessingML fragment for the content of a comments part in a WordprocessingML document:

```
<w:comments>
  <w:comment … >
    …
  </w:comment>
</w:comments>
```

The comments element contains the single comment specified by this document in this example. *end example*]

Parent Elements
Root element of WordprocessingML Comments part

Child Elements	Subclause
comment (Comment Content)	§2.13.4.2

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Comments">
   <sequence>
      <element name="comment" type="CT_Comment" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

2.13.5 Revisions

Revisions in WordprocessingML provide a mechanism for storing information about the evolution of the document (i.e. the set of modifications made to a document by one of more authors). When an application adds revisions to the content of a WordprocessingML document, they are specifying this by storing either (depending on the revision type):

- The current state of the document (a deletion stores the current state of the text as deleted, and implies that its original state was the content used to exist)
- The initial state of the document (a run's initial properties are explicitly stored in a previous run properties block, as the current run properties are always those that are the child of the rPr element (§2.7.8.1))

A revision consists of two required pieces of information:

- The revision type (specified via the name of the revision element)
- A unique revision identifier (used to uniquely identify revisions)

As well as optional information:

- The author of the revision
- The date and time of the revision

[*Example*: Consider a paragraph of text in a WordprocessingML document in which one word has been inserted, as follows:

Some text

This paragraph has the word `text` marked inserted as a revision, and is represented as the following WordprocessingML:

```
<w:p>
  <w:r>
    <w:t>Some</w:t>
  </w:r>
  <w:ins w:id="0" w:author="Joe Smith" w:date="2006-03-31T12:50:00Z">
    <w:r>
      <w:t>text</w:t>
    </w:r>
```

```
    </w:ins>
  </w:p>
```

The ins element contains both the required information: all of the content which shall be treated as revision marked as inserted (the word `text`); a unique revision identifier of `0`.

The element also stores the optional information about the revision: the word `text` was inserted by `Joe Smith` on March 31, 2006 at 12:50pm. *end example*]

Within a WordprocessingML document, the following types of revisions may be used to track the changes to a document:

- Insertions
- Deletions
- Moves
- Changes to run/paragraph/table/numbering/section properties
- Changes to custom XML markup

2.13.5.1 cellDel (Table Cell Deletion)

This element specifies that the parent table cell shall be treated as though it was deleted from the document while revisions were being recorded. This means that although the table cell element exists in the structure of the table, the table cell technically no longer exists in the document.

[*Example*: Consider a document with a two row by two columns table as follows:

One	Two
Three	Four

If this table has each cell in its final column deleted and this is tracked as a revision, the resulting WordprocessingML would show each of these cells as deleted as follows:

```
<w:tbl>
  …
  <w:tr>
    <w:tc>
      <w:r>
        <w:t>One</w:t>
      </w:r>
    </w:tc>
    <w:tc>
      <w:tcPr>
        <w:cellDel w:id="0" … />
      </w:tcPr>
      <w:r>
        <w:t>Two</w:t>
      </w:r>
    </w:tc>
  </w:tr>
  <w:tr>
    <w:tc>
      <w:r>
        <w:t>Three</w:t>
      </w:r>
    </w:tc>
    <w:tc>
      <w:tcPr>
        <w:cellDel w:id="1" … />
      </w:tcPr>
      <w:r>
        <w:t>Four</w:t>
      </w:r>
    </w:tc>
  </w:tr>
</w:tbl>
```

The cellDel elements in the table cell properties of the cells with text Two and Four specify that each of those cells have been deleted from the document. Their attributes (omitted) may optionally provide information about the time at which this deletion took place. *end example*]

Parent Elements
tcPr (§2.7.5.8); tcPr (§2.4.66); tcPr (§2.7.5.9); tcPr (§2.4.67)

Attributes	Description

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`  …`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`  …`
	`</w:…>`

Attributes	Description
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TrackChange">
   <complexContent>
      <extension base="CT_Markup">
         <attribute name="author" type="ST_String" use="required"/>
         <attribute name="date" type="ST_DateTime" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.2 cellIns (Table Cell Insertion)

This element specifies that the parent table cell shall be treated as though it was inserted into the document while revisions were being recorded.

[*Example*: Consider a document with a two row by two columns table as follows:

One	Two
Three	Four

If this table has two cells added by appending an additional column, and this is tracked as a revision, the resulting WordprocessingML would show each of these cells as inserted as follows:

```
<w:tbl>
  …
  <w:tr>
    <w:tc>
      <w:r>
        <w:t>One</w:t>
      </w:r>
    </w:tc>
    <w:tc>
      <w:r>
        <w:t>Two</w:t>
      </w:r>
    </w:tc>
    <w:tc>
      <w:tcPr>
        <w:cellIns w:id="0" … />
      </w:tcPr>
      <w:r>
        <w:t>New</w:t>
      </w:r>
    </w:tc>
  </w:tr>
  <w:tr>
    <w:tc>
      <w:r>
        <w:t>Three</w:t>
      </w:r>
    </w:tc>
    <w:tc>
      <w:r>
        <w:t>Four</w:t>
      </w:r>
    </w:tc>
    <w:tc>
      <w:tcPr>
        <w:cellIns w:id="1" … />
      </w:tcPr>
      <w:r>
        <w:t>New</w:t>
      </w:r>
    </w:tc>
  </w:tr>
</w:tbl>
```

The cellIns elements in the table cell properties of the cells with text New specify that each of those cells have been inserted into the document. Their attributes (omitted) may optionally provide information about the insertion of these cells (author, date, etc.). *end example*]

Parent Elements
tcPr (§2.7.5.8); tcPr (§2.4.66); tcPr (§2.7.5.9); tcPr (§2.4.67)

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`…`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`…`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation	Specifies a unique identifier for an annotation within a WordprocessingML document.

Attributes	Description

Identifier)	The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TrackChange">
   <complexContent>
      <extension base="CT_Markup">
         <attribute name="author" type="ST_String" use="required"/>
         <attribute name="date" type="ST_DateTime" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.3 cellMerge (Vertically Merged/Split Table Cells)

This element specifies that the vertical merge state of the parent table cell has been modified while revisions were being tracked for the document. The vmerge and vmergeOrig attributes on this element specify the original and revised vertical merge states of the table cell.

[*Example*: Consider a document with a two row by two columns table as follows:

One	Two
Three	Four

If this table has the two cells in the second column merged into one and this modification is tracked as a revision, as follows:

One	Two
Three	Four

The resulting WordprocessingML for the revision would appear as follows:

```
<w:tbl>
  …
  <w:tr>
    <w:tc>
      <w:r>
        <w:t>One</w:t>
      </w:r>
    </w:tc>
    <w:tc>
      <w:r>
        <w:t>Two</w:t>
      </w:r>
    </w:tc>
  </w:tr>
  <w:tr>
    <w:tc>
      <w:r>
        <w:t>Three</w:t>
      </w:r>
    </w:tc>
    <w:tc>
      <w:tcPr>
        <w:cellMerge w:id="0" w:vmerge="cont"/>
      </w:tcPr>
      <w:r>
        <w:t>Four</w:t>
      </w:r>
    </w:tc>
  </w:tr>
</w:tbl>
```

The cellMerge element specifies that changes were made to the vertical merge settings of the last cell in the table, specifically; the cell was vertically merged with the cell above it (gaining a revised vmerge attribute value of cont). *end example*]

Parent Elements
tcPr (§2.7.5.8); tcPr (§2.4.66); tcPr (§2.7.5.9); tcPr (§2.4.67)

Attributes	Description
author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document. If this attribute is omitted, then no author shall be associated with the parent annotation

Attributes	Description

	type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`  …`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`  …`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

vMerge (Revised Vertical Merge Setting)	Specifies the vertical merge setting which was applied to the parent table cell by this revision.
	
	If this attribute is omitted, then no revised vertical merge setting is supplied for this revision (if neither this nor the vmergeOrig attribute is specified, the revision may be ignored).
	
	[*Example*: Consider a two row by two column table in which the cells in the second column are merged, and this change is tracked as a revision. The annotation on the last cell in the table would appear as follows:
	
	`<w:tc>`
	`  <w:tcPr>`
	`  <w:cellMerge … w:vmerge="cont" />`
	`  </w:tcPr>`
	`  …`
	`</w:tc>`
	
	The vmerge attribute value of `cont` specifies that the revision on the table cell resulted in it being merged with the previous set of vertically merged cells above it (whether that was one cell or many). *end example*]
	
	The possible values for this attribute are defined by the ST_AnnotationVMerge simple type (§2.18.3).

vMergeOrig (Vertical Merge Setting Removed by Revision)	Specifies the vertical merge setting which was removed from the parent table cell by this revision.
	
	If this attribute is omitted, then the original vertical merge setting shall be assumed to be `rest` (not merged).
	
	[*Example*: Consider a two row by two column table in which the merged cells in the second column are split, and this change is tracked as a revision. The annotation on the last cell in the table would appear as follows:
	
	`<w:tc>`
	`  <w:tcPr>`
	`  <w:cellMerge … w:vmergeOrig="cont" />`
	`  </w:tcPr>`
	`  …`
	`</w:tc>`
	
	The vmergeOrig attribute value of `cont` specifies that the revision on the table cell resulted in it having its vertical merge property removed. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_AnnotationVMerge simple type (§2.18.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CellMergeTrackChange">
   <complexContent>
      <extension base="CT_TrackChange">
         <attribute name="vMerge" type="ST_AnnotationVMerge" use="optional"/>
         <attribute name="vMergeOrig" type="ST_AnnotationVMerge" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.4 customXmlDelRangeEnd (Custom XML Markup Deletion End)

This element specifies the end of a region in which custom XML markup has been deleted and tracked as a revision. The id attribute on this element shall be used to link this element with the corresponding custom XML markup deletion start marker in the document.

Providing a physical representation of custom XML markup results in regions which can be inserted and deleted, but cannot be encapsulated by a single revision element, since their representation in WordprocessingML is the start or end XML tag for the custom XML markup which it represents. Therefore, the start/end "cross structure" annotation format surrounds the WordprocessingML region to which this deletion applies.

The following restrictions shall be applied to this element:

- If this element occurs without a corresponding customXmlDelRangeStart element (§2.13.5.5) with a matching id attribute value, then it shall be ignored and no deletions shall be present in the document.
- If this element and its paired start encapsulate a range with no custom XML markup, then they shall be ignored and may be omitted when the document is subsequently saved.
- If multiple end elements exist with the same id attribute value, then the first instance in the document shall be used and subsequent elements should be treated as unmatched (no corresponding start).

[*Example*: Consider a document with two inline custom XML markup elements, as follows:

```
<w:p>
  <w:customXml … >
    <w:customXml … >
      <w:r>
        <w:t>Text.</w:t>
      </w:r>
    </w:customXml>
  </w:customXml>
  <w:r>
    <w:t>More text.</w:t>
  </w:r>
</w:p>
```

Now, if each custom XML markup element's start and end tag have a physical representation, imagine that the region from the start of the paragraph until the point between the two end points is deleted with revisions enabled. This revision cannot be encapsulated by one del element, since it starts outside of the first custom XML markup element and ends just inside of it, so it must be done using the custom XML markup revision "cross structure" syntax, as follows:

```
<w:p>
  <w:customXmlDelRangeStart w:id="0" />
  <w:customXml … >
    <w:customXml … >
      <w:del … >
        <w:r>
          <w:delText>Text.</w:delText>
        </w:r>
      </w:del>
    </w:customXml>
    <w:customXmlDelRangeEnd w:id="0" />
  </w:customXml>
  <w:r>
    <w:t>More text.</w:t>
  </w:r>
</w:p>
```

The customXmlDelRangeEnd element delimits the end of the region in which all custom XML elements have been deleted with revisions enabled, and the del element (§2.13.5.12) handles the deletion of the text performed by this revision. Since the end of the outer customXml element was not in the deleted range, it is not revision marked deleted, but the corresponding physical character for the start element is. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32);

Parent Elements
endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Attributes	Description

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Markup">
   <attribute name="id" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.13.5.5 customXmlDelRangeStart (Custom XML Markup Deletion Start)

This element specifies the beginning of a region in which all custom XML markup has been deleted and tracked as a revision. The id attribute on this element shall be used to link this element with the corresponding custom XML markup deletion end marker in the document.

Providing a physical representation of custom XML markup results in regions which can be inserted and deleted, but cannot be encapsulated by a single revision element, since their representation in WordprocessingML is the start or end XML tag for the custom XML markup which it represents. Therefore, the start/end "cross structure" annotation format surrounds the WordprocessingML region to which this deletion applies.

The following restrictions shall be applied to this element:

- If this element occurs without a corresponding customXmlDelRangeEnd element (§2.13.5.4) with a matching id attribute value, then this revision is ill-formed, and the revision may be ignored or all custom XML from this point forward may be treated as deleted.
- If this element and its paired end encapsulate a range with no custom XML markup, then they shall be ignored and may be omitted when the document is subsequently saved.
- If multiple start elements exist with the same id attribute value, then the each instance in the document shall be matched with an end in document order, and unmatched starts (no corresponding end) shall be handled as described above.

[*Example*: Consider a document with two inline custom XML markup elements, as follows:

```
<w:p>
  <w:customXml … >
    <w:customXml … >
      <w:r>
        <w:t>Text.</w:t>
      </w:r>
    </w:customXml>
  </w:customXml>
  <w:r>
    <w:t>More text.</w:t>
  </w:r>
</w:p>
```

Now, if each custom XML markup element's start and end tag have a physical representation, imagine that the region from the start of the paragraph until the point between the two end points is deleted with revisions enabled. This revision cannot be encapsulated by one del element, since it starts outside of the first custom XML markup element and ends just inside of it, so it must be done using the custom XML markup revision "cross structure" syntax, as follows:

```
<w:p>
```

```
<w:customXmlDelRangeStart w:id="0" />
<w:customXml … >
  <w:customXml … >
    <w:del … >
      <w:r>
        <w:delText>Text.</w:delText>
      </w:r>
    </w:del>
  </w:customXml>
  <w:customXmlDelRangeEnd w:id="0" />
</w:customXml>
<w:r>
  <w:t>More text.</w:t>
</w:r>
</w:p>
```

The customXmlDelRangeStart element delimits the start of the region in which all custom XML elements have been deleted with revisions enabled, and the del element (§2.13.5.12) handles the deletion of the text performed by this revision. Since the end of the outer customXml element was not in the deleted range, it is not revision marked deleted, but the corresponding physical character for the start element is. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32); endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`…`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example`

Attributes	Description
	Author, which may be used as desired. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`…`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TrackChange">
   <complexContent>
      <extension base="CT_Markup">
         <attribute name="author" type="ST_String" use="required"/>
         <attribute name="date" type="ST_DateTime" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.6 customXmlInsRangeEnd (Custom XML Markup Insertion End)

This element specifies the end of a region within which all custom XML markup has been inserted and tracked as a revision. The id attribute on this element shall be used to link this element with the corresponding custom XML markup insertion start marker in the document.

Providing a physical representation of the start and end tags of custom XML markup results in regions which can be inserted and deleted independently, but cannot be encapsulated by a single revision element, since their representation in WordprocessingML is the start or end XML tag for the custom XML markup which it represents. Therefore, the start/end "cross structure" annotation format surrounds the WordprocessingML region to which this insertion applies.

The following restrictions shall be applied to this element:

- If this element occurs without a corresponding customXmlInsRangeStart element (§2.13.5.7) with a matching id attribute value, then it shall be ignored and no insertions shall be present in the document.
- If this element and its paired start encapsulate a range with no custom XML markup, then they shall be ignored and may be omitted when the document is subsequently saved.
- If multiple end elements exist with the same id attribute value, then the first instance in the document shall be used and subsequent elements should be treated as unmatched (no corresponding start).

[*Example*: Consider a document with two inline custom XML markup elements, as follows:

```
<w:p>
  <w:customXml … >
    <w:customXml … >
      <w:r>
        <w:t>Text.</w:t>
      </w:r>
    </w:customXml>
  </w:customXml>
  <w:r>
    <w:t>More text.</w:t>
  </w:r>
</w:p>
```

If each custom XML markup element's start and end tag have a physical representation, consider that the inner XML element (but not its content) is inserted with revisions enabled. This revision cannot be encapsulated by one ins element, since the text in the element is not an insertion, so it must be done using the custom XML markup revision "cross structure" syntax, as follows:

```
<w:p>
  <w:customXml … >
    <w:customXmlInsRangeStart w:id="0" />
    <w:customXml … >
      <w:r>
        <w:t>Text.</w:t>
      </w:r>
    </w:customXml>
    <w:customXmlInsRangeEnd w:id="0" />
  </w:customXml>
  <w:r>
    <w:t>More text.</w:t>
  </w:r>
</w:p>
```

The customXmlInsRangeEnd element delimits the end of the region in which all custom XML elements have been inserted with revisions enabled. Since this element only affects custom XML, the text is not revision marked inserted, but the corresponding physical characters for the custom XML element are. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32); endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Attributes	Description

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`

Attributes	Description
	</w:…> The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Markup">
   <attribute name="id" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.13.5.7 customXmlInsRangeStart (Custom XML Markup Insertion Start)

This element specifies the beginning of a region in which all custom XML markup has been inserted and tracked as a revision. The id attribute on this element shall be used to link this element with the corresponding custom XML markup insertion end marker in the document.

Providing a physical representation of custom XML markup start and end tags results in regions which can be inserted and deleted independently, but cannot be encapsulated by a single revision element, since their representation in WordprocessingML is the start or end XML tag for the custom XML markup which it represents. Therefore, the start/end "cross structure" annotation format surrounds the WordprocessingML region to which this deletion applies.

The following restrictions shall be applied to this element:

- If this element occurs without a corresponding customXmlInsRangeEnd element (§2.13.5.6) with a matching id attribute value, then this revision is ill-formed, and the revision may be ignored or all custom XML from this point forward may be treated as inserted.
- If this element and its paired end encapsulate a range with no custom XML markup, then they shall be ignored and may be omitted when the document is subsequently saved.
- If multiple start elements exist with the same id attribute value, then the each instance in the document shall be matched with an end in document order, and unmatched starts (no corresponding end) shall be handled as described above.

This element specifies the end of a region within which all custom XML markup has been inserted and tracked as a revision. The id attribute on this element shall be used to link this element with the corresponding custom XML markup insertion start marker in the document.

Providing a physical representation of the start and end tags of custom XML markup results in regions which can be inserted and deleted independently, but cannot be encapsulated by a single revision element, since their representation in WordprocessingML is the start or end XML tag for the custom XML markup which it

represents. Therefore, the start/end "cross structure" annotation format surrounds the WordprocessingML region to which this insertion applies.

The following restrictions shall be applied to this element:

- If this element occurs without a corresponding customXmlInsRangeStart element (§2.13.5.7) with a matching id attribute value, then it shall be ignored and no insertions shall be present in the document.
- If this element and its paired start encapsulate a range with no custom XML markup, then they shall be ignored and may be omitted when the document is subsequently saved.
- If multiple end elements exist with the same id attribute value, then the first instance in the document shall be used and subsequent elements should be treated as unmatched (no corresponding start).

[*Example*: Consider a document with two inline custom XML markup elements, as follows:

```
<w:p>
  <w:customXml … >
    <w:customXml … >
      <w:r>
        <w:t>Text.</w:t>
      </w:r>
    </w:customXml>
  </w:customXml>
  <w:r>
    <w:t>More text.</w:t>
  </w:r>
</w:p>
```

If each custom XML markup element's start and end tag have a physical representation, consider that the inner XML element (but not its content) is inserted with revisions enabled. This revision cannot be encapsulated by one ins element, since the text in the element is not an insertion, so it must be done using the custom XML markup revision "cross structure" syntax, as follows:

```
<w:p>
  <w:customXml … >
    <w:customXmlInsRangeStart w:id="0" />
    <w:customXml … >
      <w:r>
        <w:t>Text.</w:t>
      </w:r>
    </w:customXml>
    <w:customXmlInsRangeEnd w:id="0" />
  </w:customXml>
  <w:r>
    <w:t>More text.</w:t>
  </w:r>
</w:p>
```

The customXmlInsRangeStart element delimits the start of the region in which all custom XML elements have been inserted with revisions enabled. Since this element only affects custom XML, the text is not revision marked inserted, but the corresponding physical characters for the custom XML element are. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32); endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`  …`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`…`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TrackChange">
   <complexContent>
      <extension base="CT_Markup">
         <attribute name="author" type="ST_String" use="required"/>
         <attribute name="date" type="ST_DateTime" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.8 customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)

This element specifies the end of a region within which all custom XML markup was moved to another location in the document and this move was tracked as a revision. The id attribute on this element shall be used to link this element with the corresponding custom XML move source start marker in the document.

Providing a physical representation of the start and end tags of custom XML markup results in regions which can be inserted and deleted independently, but cannot be encapsulated by a single revision element, since their representation in WordprocessingML is the start or end XML tag for the custom XML markup which it represents. Therefore, the start/end "cross structure" annotation format surrounds the WordprocessingML region to which this move source applies.

The following restrictions shall be applied to this element:

- If this element occurs without a corresponding customXmlMoveFromRangeStart element (§2.13.5.9) with a matching id attribute value, then it shall be ignored and no move source information shall be applied to the custom XML elements by this element.
- If this element and its paired start encapsulate a range with no custom XML markup, then they shall be ignored and may be omitted when the document is subsequently saved.
- If this element and its paired start occur outside of a valid move source container (§2.13.5.24; §2.13.5.23) with a matching move destination container (§2.13.5.28; §2.13.5.27), then custom XML markup in this region shall be treated as if it was deleted
- If multiple end elements exist with the same id attribute value, then the first instance in the document shall be used and subsequent elements should be treated as unmatched (no corresponding start).

[*Example*: Consider a three-paragraph document with a single block-level custom XML markup element, as follows:

```
<w:body>
  <w:p/>
  <w:customXml … >
    <w:p/>
  </w:customXml>
  <w:p/>
</w:body>
```

If the second paragraph is moved to the end of the document with revisions enabled. This revision must therefore be stored using the custom XML markup revision "cross structure" syntax, as follows:

```
<w:body>
  <w:p/>
  <w:moveFromRangeStart w:id="0" w:name="move1" w:displacedByCustomXml="next"/>
  <w:customXmlMoveFromRangeStart w:id="1"/>
  <w:customXml … >
    <w:p/>
  </w:customXml>
  <w:customXmlMoveFromRangeEnd w:id="1"/>
  <w:moveFromRangeEnd w:id="0" w:displacedByCustomXml="prev"/>
  <w:p/>
  <w:moveToRangeStart w:id="2" w:name="move1" w:displacedByCustomXml="next"/>
  <w:customXmlMoveToRangeStart w:id="3"/>
  <w:customXml … >
    <w:p/>
  </w:customXml>
  <w:customXmlMoveToRangeEnd w:id="3"/>
  <w:moveFromRangeEnd w:id="2" w:displacedByCustomXml="prev"/>
</w:body>
```

The customXmlMoveFromRangeEnd element delimits the end of the region in which all custom XML elements have been moved from this location with revisions enabled. Since this element only affects custom XML, any text in the region is not revision marked moved by this element when present, but the corresponding physical characters for the custom XML element are. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32); endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Attributes	Description
id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element. If this attribute is omitted, then the document is non-conformant. [*Example*: Consider an annotation represented using the following WordprocessingML fragment: `<w:… w:id="1" … >`

Attributes	Description
	… </w:…> The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Markup">
   <attribute name="id" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.13.5.9 customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)

This element specifies the start of a region within which all custom XML markup was moved to another location in the document and this move was tracked as a revision. The id attribute on this element shall be used to link this element with the corresponding custom XML move source end marker in the document.

Providing a physical representation of the start and end tags of custom XML markup results in regions which can be inserted and deleted independently, but cannot be encapsulated by a single revision element, since their representation in WordprocessingML is the start or end XML tag for the custom XML markup which it represents. Therefore, the start/end "cross structure" annotation format surrounds the WordprocessingML region to which this move source applies.

The following restrictions shall be applied to this element:

- If this element occurs without a corresponding customXmlMoveFromRangeStart element (§2.13.5.9) with a matching id attribute value, then it shall be ignored and no move source information shall be applied to the custom XML elements by this element.
- If this element and its paired start encapsulate a range with no custom XML markup, then they shall be ignored and may be omitted when the document is subsequently saved.
- If this element and its paired end occur outside of a valid move source container (§2.13.5.24; §2.13.5.23) with a matching move destination container (§2.13.5.28; §2.13.5.27), then custom XML markup in this region shall be treated as if it was deleted
- If multiple start elements exist with the same id attribute value, then the each instance in the document shall be matched with an end in document order, and unmatched starts (no corresponding end) shall be handled as described above.

[*Example*: Consider a three-paragraph document with a single block-level custom XML markup element, as follows:

```
<w:body>
  <w:p/>
```

```
    <w:customXml … >
      <w:p/>
    </w:customXml>
    <w:p/>
  </w:body>
```

If the second paragraph is moved to the end of the document with revisions enabled. This revision must therefore be stored using the custom XML markup revision "cross structure" syntax, as follows:

```
  <w:body>
    <w:p/>
    <w:moveFromRangeStart w:id="0" w:name="move1" w:displacedByCustomXml="next"/>
    <w:customXmlMoveFromRangeStart w:id="1"/>
    <w:customXml … >
      <w:p/>
    </w:customXml>
    <w:customXmlMoveFromRangeEnd w:id="1"/>
    <w:moveFromRangeEnd w:id="0" w:displacedByCustomXml="prev"/>
    <w:p/>
    <w:moveToRangeStart w:id="2" w:name="move1" w:displacedByCustomXml="next"/>
    <w:customXmlMoveToRangeStart w:id="3"/>
    <w:customXml … >
      <w:p/>
    </w:customXml>
    <w:customXmlMoveToRangeEnd w:id="3"/>
    <w:moveFromRangeEnd w:id="2" w:displacedByCustomXml="prev"/>
  </w:body>
```

The customXmlMoveFromRangeStart element delimits the start of the region in which all custom XML elements have been moved from this location with revisions enabled. Since this element only affects custom XML, any text in the region is not revision marked moved by this element when present, but the corresponding physical characters for the custom XML element are. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32); endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Attributes	Description

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`  …`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`  …`
	`</w:…>`

Attributes	Description
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TrackChange">
   <complexContent>
      <extension base="CT_Markup">
         <attribute name="author" type="ST_String" use="required"/>
         <attribute name="date" type="ST_DateTime" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.10 customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)

This element specifies the end of a region within which all custom XML markup was moved to this location in the document and this move was tracked as a revision. The id attribute on this element shall be used to link this element with the corresponding custom XML move destination start marker in the document.

Providing a physical representation of the start and end tags of custom XML markup results in regions which can be inserted and deleted independently, but cannot be encapsulated by a single revision element, since their representation in WordprocessingML is the start or end XML tag for the custom XML markup which it represents. Therefore, the start/end "cross structure" annotation format surrounds the WordprocessingML region to which this move destination applies.

The following restrictions shall be applied to this element:

- If this element occurs without a corresponding customXmlMoveToRangeStart element (§2.13.5.11) with a matching id attribute value, then it shall be ignored and no move destination information shall be applied to the custom XML elements by this element.
- If this element and its paired start encapsulate a range with no custom XML markup, then they shall be ignored and may be omitted when the document is subsequently saved.
- If this element and its paired start occur outside of a valid move source container (§2.13.5.24; §2.13.5.23) with a matching move destination container (§2.13.5.28; §2.13.5.27), then custom XML markup in this region shall be treated as if it was inserted
- If multiple end elements exist with the same id attribute value, then the first instance in the document shall be used and subsequent elements should be treated as unmatched (no corresponding start).

[*Example*: Consider a three-paragraph document with a single block-level custom XML markup element, as follows:

```
<w:body>
  <w:p/>
  <w:customXml … >
    <w:p/>
  </w:customXml>
  <w:p/>
</w:body>
```

If the second paragraph is moved to the end of the document with revisions enabled. This revision must therefore be stored using the custom XML markup revision "cross structure" syntax, as follows:

```
<w:body>
  <w:p/>
  <w:moveFromRangeStart w:id="0" w:name="move1" w:displacedByCustomXml="next"/>
  <w:customXmlMoveFromRangeStart w:id="1"/>
  <w:customXml … >
    <w:p/>
  </w:customXml>
  <w:customXmlMoveFromRangeEnd w:id="1"/>
  <w:moveFromRangeEnd w:id="0" w:displacedByCustomXml="prev"/>
  <w:p/>
  <w:moveToRangeStart w:id="2" w:name="move1" w:displacedByCustomXml="next"/>
  <w:customXmlMoveToRangeStart w:id="3"/>
  <w:customXml … >
    <w:p/>
  </w:customXml>
  <w:customXmlMoveToRangeEnd w:id="3"/>
  <w:moveFromRangeEnd w:id="2" w:displacedByCustomXml="prev"/>
</w:body>
```

The customXmlMoveToRangeEnd element delimits the end of the region in which all custom XML elements have been moved to this location with revisions enabled. Since this element only affects custom XML, any text in the region is not revision marked moved by this element when present, but the corresponding physical characters for the custom XML element are. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32); endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Attributes	Description

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Markup">
   <attribute name="id" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.13.5.11 customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)

This element specifies the start of a region within which all custom XML markup was moved to this location in the document and this move was tracked as a revision. The id attribute on this element shall be used to link this element with the corresponding custom XML move destination end marker in the document.

Providing a physical representation of the start and end tags of custom XML markup results in regions which can be inserted and deleted independently, but cannot be encapsulated by a single revision element, since their representation in WordprocessingML is the start or end XML tag for the custom XML markup which it represents. Therefore, the start/end "cross structure" annotation format surrounds the WordprocessingML region to which this move destination applies.

The following restrictions shall be applied to this element:

- If this element occurs without a corresponding customXmlMoveFromRangeEnd element (§2.13.5.8) with a matching id attribute value, then it shall be ignored and no move source information shall be applied to the custom XML elements by this element.
- If this element and its paired start encapsulate a range with no custom XML markup, then they shall be ignored and may be omitted when the document is subsequently saved.

- If this element and its paired end occur outside of a valid move source container (§2.13.5.24; §2.13.5.23) with a matching move destination container (§2.13.5.28; §2.13.5.27), then custom XML markup in this region shall be treated as if it was inserted
- If multiple start elements exist with the same id attribute value, then the each instance in the document shall be matched with an end in document order, and unmatched starts (no corresponding end) shall be handled as described above.

[*Example*: Consider a three-paragraph document with a single block-level custom XML markup element, as follows:

```
<w:body>
  <w:p/>
  <w:customXml … >
    <w:p/>
  </w:customXml>
  <w:p/>
</w:body>
```

If the second paragraph is moved to the end of the document with revisions enabled. This revision must therefore be stored using the custom XML markup revision "cross structure" syntax, as follows:

```
<w:body>
  <w:p/>
  <w:moveFromRangeStart w:id="0" w:name="move1" w:displacedByCustomXml="next"/>
  <w:customXmlMoveFromRangeStart w:id="1"/>
  <w:customXml … >
    <w:p/>
  </w:customXml>
  <w:customXmlMoveFromRangeEnd w:id="1"/>
  <w:moveFromRangeEnd w:id="0" w:displacedByCustomXml="prev"/>
  <w:p/>
  <w:moveToRangeStart w:id="2" w:name="move1" w:displacedByCustomXml="next"/>
  <w:customXmlMoveToRangeStart w:id="3"/>
  <w:customXml … >
    <w:p/>
  </w:customXml>
  <w:customXmlMoveToRangeEnd w:id="3"/>
  <w:moveFromRangeEnd w:id="2" w:displacedByCustomXml="prev"/>
</w:body>
```

The customXmlMoveFromRangeStart element delimits the start of the region in which all custom XML elements have been moved from this location with revisions enabled. Since this element only affects custom XML, any text in the region is not revision marked moved by this element when present, but the corresponding physical characters for the custom XML element are. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32); endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`  …`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation	Specifies a unique identifier for an annotation within a WordprocessingML document.

Attributes	Description

Identifier)	The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TrackChange">
   <complexContent>
      <extension base="CT_Markup">
         <attribute name="author" type="ST_String" use="required"/>
         <attribute name="date" type="ST_DateTime" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.12 del (Deleted Run Content)

This element specifies that the inline-level content contained within it shall be treated as deleted content which has been tracked as a revision.

[*Example*: Consider a paragraph of text in a WordprocessingML document in which one word has been deleted, as follows:

Some ~~text~~

This paragraph has the word `text` marked deleted as a revision, and is represented as the following WordprocessingML:

```
<w:p>
  <w:r>
    <w:t>Some</w:t>
  </w:r>
  <w:del w:id="0" w:author="Joe Smith" w:date="2006-03-31T12:50:00Z">
    <w:r>
```

```
      <w:delText>text</w:delText>
    </w:r>
  </w:del>
</w:p>
```

The del element contains all of the content which shall be treated as revision marked as deleted; in this case, the word `text` was deleted by `Joe Smith` on March 31, 2006 at 12:50pm. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32); endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Child Elements	Subclause
acc (Accent)	§7.1.2.1
bar (Bar)	§7.1.2.7
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
borderBox (Border-Box Function)	§7.1.2.11
box (Box Function)	§7.1.2.13
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Inline-Level Custom XML Element)	§2.5.1.5
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
d (Delimiter Function)	§7.1.2.24
del (Deleted Run Content)	§2.13.5.12
eqArr (Equation-Array Function)	§7.1.2.34

Child Elements	Subclause
f (Fraction Function)	§7.1.2.36
func (Function Apply Function)	§7.1.2.39
groupChr (Group-Character Function)	§7.1.2.41
ins (Inserted Run Content)	§2.13.5.20
limLow (Lower-Limit Function)	§7.1.2.54
limUpp (Upper-Limit Function)	§7.1.2.56
m (Matrix Function)	§7.1.2.60
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
nary (n-ary Operator Function)	§7.1.2.70
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
phant (Phantom Function)	§7.1.2.81
proofErr (Proofing Error Anchor)	§2.13.8.1
r (Text Run)	§2.3.2.23
r (Run)	§7.1.2.87
rad (Radical Function)	§7.1.2.88
sdt (Inline-Level Structured Document Tag)	§2.5.2.29
smartTag (Inline-Level Smart Tag)	§2.5.1.9
sPre (Pre-Sub-Superscript Function)	§7.1.2.99
sSub (Subscript Function)	§7.1.2.101
sSubSup (Sub-Superscript Function)	§7.1.2.103
sSup (Superscript Function)	§7.1.2.105

Attributes	Description
author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document. If this attribute is omitted, then no author shall be associated with the parent annotation

Attributes	Description

	type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`  …`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`  …`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RunTrackChange">
   <complexContent>
      <extension base="CT_TrackChange">
         <choice minOccurs="0" maxOccurs="unbounded">
            <group ref="EG_ContentRunContent"/>
            <group ref="m:EG_OMathMathElements"/>
         </choice>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.13 del (Deleted Paragraph)

This element specifies that the paragraph mark delimiting the end of a paragraph within a WordprocessingML document shall be treated as deleted (i.e. the contents of this paragraph are no longer delimited by this paragraph mark, and are combined with the following paragraph - but those contents shall not automatically be marked as deleted) as part of a tracked revision.

[*Example*: Consider a document consisting of two paragraphs (with each paragraph delimited by a pilcrow ¶), as follows:

This·is·paragraph·one.¶

This·is·paragraph·two.¶

If the physical character delimiting the end of the first paragraph is deleted and this change is tracked as a revision, resulting in the following:

This·is·paragraph·one.This·is·paragraph·two.¶

This revision is represented using the following WordprocessingML:

```
<w:p>
  <w:pPr>
    <w:rPr>
      <w:del w:id="0" … />
    </w:rPr>
  </w:pPr>
  <w:r>
    <w:t>This is paragraph one.</w:t>
  </w:r>
</w:p>
<w:p>
  <w:r>
    <w:t>This is paragraph two.</w:t>
  </w:r>
</w:p>
```

The del element on the run properties for the first paragraph mark specifies that this paragraph mark was deleted, and this deletion was tracked as a revision. *end example*]

Parent Elements
rPr (§2.3.1.29); rPr (§2.3.1.30)

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`  …`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard. If this attribute is omitted, then no date information shall be associated with the parent annotation type.

Attributes	Description

	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`…`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TrackChange">
   <complexContent>
      <extension base="CT_Markup">
         <attribute name="author" type="ST_String" use="required"/>
         <attribute name="date" type="ST_DateTime" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.14 del (Deleted Table Row)

This element specifies that the parent table row shall be treated as a deleted row whose deletion has been tracked as a revision. This setting shall not imply any revision state about the table cells in this row or their contents (which must be revision marked independently), and shall only affect the table row itself.

[*Example*: Consider a two row by two column table in which the second row has been marked as deleted using a revision. This requirement would be specified using the following WordprocessingML:

```
<w:tbl>
  <w:tr>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
  </w:tr>
  <w:tr>
    <w:trPr>
      <w:del w:id="0" … />
    </w:trPr>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
  </w:tr>
</w:tbl>
```

The del element on the table row properties for the second table row specifies that this row was deleted, and this deletion was tracked as a revision. *end example*]

Parent Elements
trPr (§2.7.5.10); trPr (§2.7.5.11); trPr (§2.4.78)

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`  …`
	`</w:…>`

Attributes	Description
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`…`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TrackChange">
   <complexContent>
      <extension base="CT_Markup">
         <attribute name="author" type="ST_String" use="required"/>
         <attribute name="date" type="ST_DateTime" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.15 del (Deleted Math Control Character)

This element specifies that the Office Open XML Math control character which contains this element was deleted and tracked as a revision. [*Example*: The deletion of a fraction bar. *end example*]

[*Example*: Consider a region of Office Open XML Math in a WordprocessingML document in which the control character for the fraction bar has been deleted, as follows:

This deletion is represented as the following WordprocessingML:

```
<m:f>
  <m:fPr>
    <m:ctrlPr>
      <w:del w:id="0" w:author="Joe Smith" w:date="2006-03-31T12:50:00Z">
        ...
      </w:del>
    </m:ctrlPr>
  </m:fPr>
  ...
</m:f>
```

The del element contains all of the content which shall be treated as revision marked as deleted; in this case, the fraction bar was deleted by `Joe Smith` on March 31, 2006 at 12:50pm. *end example*]

Parent Elements
ctrlPr (§7.1.2.23)

Child Elements	Subclause
rPr (Previous Run Properties)	§2.3.2.26

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`  …`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`  …`
	`</w:…>`

Attributes	Description
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RPrChange">
   <complexContent>
      <extension base="CT_TrackChange">
         <sequence>
            <element name="rPr" type="CT_RPrOriginal" minOccurs="1"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.16 ins (Inserted Table Row)

This element specifies that the parent table row shall be treated as an inserted row whose insertion has been tracked as a revision. This setting shall not imply any revision state about the table cells in this row or their contents (which must be revision marked independently), and shall only affect the table row itself.

[*Example*: Consider a two row by two column table in which the second row has been marked as inserted using a revision. This requirement would be specified using the following WordprocessingML:

```
<w:tbl>
  <w:tr>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
  </w:tr>
  <w:tr>
    <w:trPr>
      <w:ins w:id="0" … />
    </w:trPr>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
  </w:tr>
</w:tbl>
```

The ins element on the table row properties for the second table row specifies that this row was inserted, and this insertion was tracked as a revision. *end example*]

Parent Elements
trPr (§2.7.5.10); trPr (§2.7.5.11); trPr (§2.4.78)

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`  …`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`…`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TrackChange">
   <complexContent>
      <extension base="CT_Markup">
         <attribute name="author" type="ST_String" use="required"/>
         <attribute name="date" type="ST_DateTime" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.17 ins (Inserted Math Control Character)

This element specifies that the Office Open XML Math control character which contains this element was inserted and tracked as a revision. [*Example*: The insertion of a fraction bar. *end example*]

[*Example*: Consider a region of Office Open XML Math in a WordprocessingML document in which the control character for the fraction bar has been inserted, as follows:

This insertion is represented as the following WordprocessingML:

```
<m:f>
  <m:fPr>
    <m:ctrlPr>
      <w:ins w:id="0" w:author="Joe Smith" w:date="2006-03-31T12:50:00Z">
        ...
      </w:ins>
    </m:ctrlPr>
  </m:fPr>
  ...
</m:f>
```

The ins element contains all of the content which shall be treated as revision marked as inserted; in this case, the fraction bar was inserted by `Joe Smith` on March 31, 2006 at 12:50pm. *end example*]

Parent Elements
ctrlPr (§7.1.2.23)

Child Elements	Subclause
rPr (Previous Run Properties)	§2.3.2.26

Attributes	Description
author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document. If this attribute is omitted, then no author shall be associated with the parent annotation type. [*Example*: Consider a comment represented using the following WordprocessingML fragment: `<w:… w:id="1" w:author="Example Author">`

Attributes	Description
	… </w:…> The author attribute specifies that the author of the current annotation is Example Author, which may be used as desired. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).
date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard. If this attribute is omitted, then no date information shall be associated with the parent annotation type. [*Example*: Consider a comment represented using the following WordprocessingML fragment: <w:… w:id="1" w:date="2006-01-01T10:00:00"> … </w:…> The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*] The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).
id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element. If this attribute is omitted, then the document is non-conformant. [*Example*: Consider an annotation represented using the following WordprocessingML fragment: <w:… w:id="1" … > … </w:…> The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

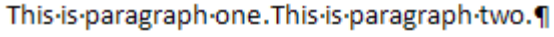
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RPrChange">
   <complexContent>
      <extension base="CT_TrackChange">
         <sequence>
            <element name="rPr" type="CT_RPrOriginal" minOccurs="1"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

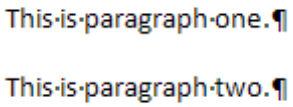
2.13.5.18 ins (Inserted Paragraph)

This element specifies that the paragraph mark delimiting the end of a paragraph within a WordprocessingML document shall be treated as deleted (i.e. the contents of this paragraph are no longer delimited by this paragraph mark, and are combined with the following paragraph) as part of a tracked revision.

[*Example*: Consider a document consisting of a single paragraph, as follows:



If the first sentence is moved into its own new paragraph, and this change is tracked as a revision, resulting in the following:



This revision is represented using the following WordprocessingML:

```
<w:p>
  <w:pPr>
    <w:rPr>
      <w:ins w:id="0" … />
    </w:rPr>
  </w:pPr>
  <w:r>
    <w:t>This is paragraph one.</w:t>
  </w:r>
</w:p>
<w:p>
  <w:r>
    <w:t>This is paragraph two.</w:t>
  </w:r>
</w:p>
```

The ins element on the run properties for the first paragraph mark specifies that this paragraph mark was inserted, and this insertion was tracked as a revision. *end example*]

Parent Elements
rPr (§2.3.1.29); rPr (§2.3.1.30)

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`…`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`…`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element. If this attribute is omitted, then the document is non-conformant.

Attributes	Description

	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TrackChange">
   <complexContent>
      <extension base="CT_Markup">
         <attribute name="author" type="ST_String" use="required"/>
         <attribute name="date" type="ST_DateTime" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.19 ins (Inserted Numbering Properties)

This element specifies that the numbering information defined by the parent element shall be treated as numbering information which was recorded as an insertion using revisions.

[*Example*: Consider two paragraphs in a WordprocessingML document, with the words one and two respectively, as follows:

One

Two

If numbering is then applied to these two paragraphs, and this numbering is tracked as a revision, this revision is represented using the following WordprocessingML:

```
<w:p>
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="0" />
      <w:numId w:val="1" />
      <w:ins w:id="0" w:author="Joe Smith" w:date="20050101T10:00:00Z" />
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>one</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="0" />
      <w:numId w:val="1" />
      <w:ins w:id="0" w:author="Joe Smith" w:date="20050101T10:00:00Z" />
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>two</w:t>
  </w:r>
</w:p>
```

The ins element as a child of the numbering properties specifies that the paragraphs in this document have been given numbering properties by `Joe Smith` and that this change was marked as a revision. *end example*]

Parent Elements
numPr (§2.3.1.19)

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`…`
	`</w:…>`

Attributes	Description
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`  …`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TrackChange">
   <complexContent>
      <extension base="CT_Markup">
         <attribute name="author" type="ST_String" use="required"/>
         <attribute name="date" type="ST_DateTime" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.20 ins (Inserted Run Content)

This element specifies that the inline-level content contained within it shall be treated as inserted content which has been tracked as a revision.

[*Example*: Consider a paragraph of text in a WordprocessingML document in which one word has been inserted, as follows:

Some text

This paragraph has the word `text` marked inserted as a revision, and is represented as the following WordprocessingML:

```
<w:p>
  <w:r>
    <w:t>Some</w:t>
  </w:r>
  <w:ins w:id="0" w:author="Joe Smith" w:date="2006-03-31T12:50:00Z">
    <w:r>
      <w:t>text</w:t>
    </w:r>
  </w:ins>
</w:p>
```

The ins element contains all of the content which shall be treated as revision marked as inserted; in this case, the word `text` was inserted by `Joe Smith` on March 31, 2006 at 12:50pm. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32); endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Child Elements	Subclause
acc (Accent)	§7.1.2.1
bar (Bar)	§7.1.2.7
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
borderBox (Border-Box Function)	§7.1.2.11
box (Box Function)	§7.1.2.13
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Inline-Level Custom XML Element)	§2.5.1.5
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
d (Delimiter Function)	§7.1.2.24
del (Deleted Run Content)	§2.13.5.12
eqArr (Equation-Array Function)	§7.1.2.34
f (Fraction Function)	§7.1.2.36
func (Function Apply Function)	§7.1.2.39
groupChr (Group-Character Function)	§7.1.2.41
ins (Inserted Run Content)	§2.13.5.20
limLow (Lower-Limit Function)	§7.1.2.54
limUpp (Upper-Limit Function)	§7.1.2.56
m (Matrix Function)	§7.1.2.60
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
nary (n-ary Operator Function)	§7.1.2.70

Child Elements	Subclause
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
phant (Phantom Function)	§7.1.2.81
proofErr (Proofing Error Anchor)	§2.13.8.1
r (Text Run)	§2.3.2.23
r (Run)	§7.1.2.87
rad (Radical Function)	§7.1.2.88
sdt (Inline-Level Structured Document Tag)	§2.5.2.29
smartTag (Inline-Level Smart Tag)	§2.5.1.9
sPre (Pre-Sub-Superscript Function)	§7.1.2.99
sSub (Subscript Function)	§7.1.2.101
sSubSup (Sub-Superscript Function)	§7.1.2.103
sSup (Superscript Function)	§7.1.2.105

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`…`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard. If this attribute is omitted, then no date information shall be associated with the parent annotation type.

Attributes	Description

	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`…`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RunTrackChange">
   <complexContent>
      <extension base="CT_TrackChange">
         <choice minOccurs="0" maxOccurs="unbounded">
            <group ref="EG_ContentRunContent"/>
            <group ref="m:EG_OMathMathElements"/>
         </choice>
      </extension>
   </complexContent>
</complexType>
```

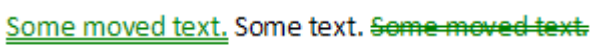
2.13.5.21 moveFrom (Move Source Run Content)

This element specifies that the inline-level content contained within it shall be treated as content which has been moved away from this location and tracked as a revision.

The following restrictions shall be applied to this content:

- If this element occurs outside of a valid move source container (§2.13.5.24; §2.13.5.23) for which a matching move destination container (§2.13.5.28; §2.13.5.27) exists in the document, then content in this region shall be treated as deleted, rather than moved.

[*Example*: Consider a WordprocessingML document in which the first paragraph contains two sentences, and the first sentence is moved before the second sentence, and this move is tracked as a revision, as follows (in this image, green underline indicates the move destination and the green strikethrough indicates the move source location):



This document has the sentence `Some moved text.` moved to the first sentence in the document. This revision is represented using the following WordprocessingML:

```
<w:p>
  <w:moveToRangeStart w:id="0" … w:name="move1" />
  <w:moveTo w:id="1" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveTo>
  <w:moveToRangeEnd w:id="0" />
  <w:r>
    <w:t xml:space="preserve">Some text.</w:t>
  </w:r>
  <w:moveFromRangeStart w:id="2" … w:name="move1" />
  <w:moveFrom w:id="3" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveFrom>
  <w:moveFromRangeEnd w:id="2" />
</w:p>
```

The moveFrom element specifies that all of the inline-level content contained within shall be revision marked as content which was moved from its current location in the document. Because this moved content is contained within a complete move source container (moveFromRangeStart and moveFromRangeEnd) with a corresponding move destination, this content is tracked as a move. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32);

Parent Elements
endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Child Elements	Subclause
acc (Accent)	§7.1.2.1
bar (Bar)	§7.1.2.7
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
borderBox (Border-Box Function)	§7.1.2.11
box (Box Function)	§7.1.2.13
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Inline-Level Custom XML Element)	§2.5.1.5
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
d (Delimiter Function)	§7.1.2.24
del (Deleted Run Content)	§2.13.5.12
eqArr (Equation-Array Function)	§7.1.2.34
f (Fraction Function)	§7.1.2.36
func (Function Apply Function)	§7.1.2.39
groupChr (Group-Character Function)	§7.1.2.41
ins (Inserted Run Content)	§2.13.5.20
limLow (Lower-Limit Function)	§7.1.2.54
limUpp (Upper-Limit Function)	§7.1.2.56
m (Matrix Function)	§7.1.2.60
moveFrom (Move Source Run Content)	§2.13.5.21

Child Elements	Subclause
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
nary (n-ary Operator Function)	§7.1.2.70
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
phant (Phantom Function)	§7.1.2.81
proofErr (Proofing Error Anchor)	§2.13.8.1
r (Text Run)	§2.3.2.23
r (Run)	§7.1.2.87
rad (Radical Function)	§7.1.2.88
sdt (Inline-Level Structured Document Tag)	§2.5.2.29
smartTag (Inline-Level Smart Tag)	§2.5.1.9
sPre (Pre-Sub-Superscript Function)	§7.1.2.99
sSub (Subscript Function)	§7.1.2.101
sSubSup (Sub-Superscript Function)	§7.1.2.103
sSup (Superscript Function)	§7.1.2.105

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`…`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`…`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RunTrackChange">
   <complexContent>
      <extension base="CT_TrackChange">
         <choice minOccurs="0" maxOccurs="unbounded">
            <group ref="EG_ContentRunContent"/>
            <group ref="m:EG_OMathMathElements"/>
         </choice>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.22 moveFrom (Move Source Paragraph)

This element specifies that the parent paragraph has been moved away from this location and tracked as a revision. This does not imply anything about the revision state of the contents of the paragraph, and applies only to the existence of the paragraph as its own unique paragraph.

The following restrictions shall be applied to this content:

- If this element occurs outside of a valid move source container (§2.13.5.24; §2.13.5.23) for which a matching move destination container (§2.13.5.28; §2.13.5.27) exists in the document, then content in this region shall be treated as deleted, rather than moved.

[*Example*: Consider a WordprocessingML document in which a paragraph of text is moved down in the document. This moved paragraph would be represented using the following WordprocessingML markup:

```
<w:moveFromRangeStart w:id="0" w:name="aMove"/>
<w:p>
  <w:pPr>
    <w:rPr>
      <w:moveFrom w:id="1" … />
    </w:rPr>
  </w:pPr>
  …
</w:p>
<w:moveFromRangeEnd w:id="0"/>
```

The moveFrom element as a child of the run properties of the paragraph mark specify that this paragraph mark was part of the content which was moved in the document. This implies nothing about the contents, since they may have been added later and not tracked as a revision (they must be marked as a move using the moveFrom element (§2.13.5.21) around the run content). *end example*]

Parent Elements
rPr (§2.3.1.29); rPr (§2.3.1.30)

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`  …`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`  …`
	`</w:…>`

Attributes	Description
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TrackChange">
   <complexContent>
      <extension base="CT_Markup">
         <attribute name="author" type="ST_String" use="required"/>
         <attribute name="date" type="ST_DateTime" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.23 moveFromRangeEnd (Move Source Location Container - End)

This element specifies the end of a region whose move source contents are part of a single named move. When a move source is stored as a revision in a WordprocessingML document, two pieces of information must be stored about that move source:

- A set of pieces of content which were moved - both inline-level content (§2.13.5.21) and paragraphs (§2.13.5.22)
- A move source container (or "bookmark") which specifies that all content within it which marked as a move source is part of a single named move. The name attribute on the move container links a group of move source content with the corresponding group of move destination content.

This element defines the end of the latter piece of the move revision data - the container. The id attribute on this element shall be used to link this element with the corresponding start of a move source container in the document.

The following restrictions are applied to the use of this element:

- If this element occurs without a corresponding moveFromRangeStart element (§2.13.5.24) with a matching id attribute value, then it shall be ignored and no move source container exists
- If this element and its paired end occur without a matching move destination container (§2.13.5.28; §2.13.5.27), then moved content in this region shall be treated as if it was deleted
- If multiple move source containers surround the same text, the last valid container (determined by the location of the container start elements, in document order) should be the container associated with that text.

[*Example*: Consider a WordprocessingML document in which the first paragraph contains two sentences, and the first sentence is moved before the second sentence, and this move is tracked as a revision, as follows (in this

image, green underline indicates the move destination and the green strikethrough indicates the move source location):

Some moved text. Some text. ~~Some moved text.~~

This document has the sentence `Some moved text.` moved to the first sentence in the document. This revision is represented using the following WordprocessingML:

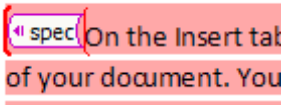
```
<w:p>
  <w:moveToRangeStart w:id="0" … w:name="move1" />
  <w:moveTo w:id="1" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveTo>
  <w:moveToRangeEnd w:id="0" />
  <w:r>
    <w:t xml:space="preserve">Some text.</w:t>
  </w:r>
  <w:moveFromRangeStart w:id="2" … w:name="move1" />
  <w:moveFrom w:id="3" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveFrom>
  <w:moveFromRangeEnd w:id="2" />
</w:p>
```

The moveFromRangeEnd element specifies the end of the move source container within which all moved content is part of the move named `move1`. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32); endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Attributes	Description
displacedByCustomXml (Annotation	Specifies that the parent annotation's placement shall be directly linked with the location of the physical presentation of a custom XML element in the document. This element

Attributes	Description

Marker Relocated For Custom XML Markup)	only has an effect when the custom XML element is block-level (i.e. surrounds an entire paragraph), as in this scenario the logical and physical placement of the annotation and custom XML element may differ.
	
	Specifically, in this case, the custom XML is presented *around* the block-level object it encloses (the paragraph, table, table row, or table cell), but is physically represented within that same object (i.e. within the paragraph, table, table row or table cell). This requirement stems from the fact that there is no location for the location of the annotation within the document at its logical location (around a table, for example).
	
	If this element is omitted, then the annotation shall be anchored inside of all block-level custom XML elements in the paragraph. If this element is present, but no block-level custom XML tag is located at the position it specifies (before or after), then it shall be ignored.
	
	[*Example*: Consider a paragraph with block level custom XML markup and two comment anchor annotations (one before and one after the custom XML element's physical representation), as follows:
	
	 
	
	Since all three of these items are around the entire paragraph, they are stored outside of the paragraph. However, in order to ensure that their relative positions are stored correctly, any annotation which shall be displaced by the physical custom XML element specifies this information, resulting in the following WordprocessingML:
	
	`<w:commentRangeStart w:id="0" />`
	`<w:commentRangeStart w:id="1" w:displaced byCustomXml="next" />`
	`<w:customXml w:element="spec" … />`
	`<w:p>`
	`…`
	`</w:p>`
	
	The displacedByCustomXml attribute specifies that even though all three of these items are around the paragraph and will be moved inside the paragraph to be represented physically, the comment with ID 0 shall be inside the custom XML, but the comment with ID 1 shall be displaced to stay outside of the relative location of the next custom XML element (the spec element). *end example*]
	
	The possible values for this attribute are defined by the ST_DisplacedByCustomXml simple type (§2.18.17).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.

Attributes	Description
	If this attribute is omitted, then the document is non-conformant. [*Example*: Consider an annotation represented using the following WordprocessingML fragment: `<w:… w:id="1" … > … </w:…>` The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MarkupRange">
   <complexContent>
      <extension base="CT_Markup">
         <attribute name="displacedByCustomXml" type="ST_DisplacedByCustomXml" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.24 moveFromRangeStart (Move Source Location Container - Start)

This element specifies the start of a region whose move source contents are part of a single named move. When a move source is stored as a revision in a WordprocessingML document, two pieces of information must be stored about that move source:

- A set of pieces of content which were moved - both inline-level content (§2.13.5.21) and paragraphs (§2.13.5.22)
- A move source container (or "bookmark") which specifies that all content within it which marked as a move source is part of a single named move. The name attribute on the move container links a group of move source content with the corresponding group of move destination content.

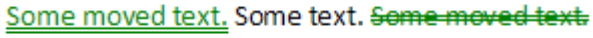
This element defines the start of the latter piece of the move revision data - the container. The id attribute on this element shall be used to link this element with the corresponding end of a move source container in the document.

The following restrictions are applied to the use of this element

- If this element occurs without a corresponding moveFromRangeEnd element (§2.13.5.23) with a matching id attribute value, then it shall be ignored and no move source container exists
- If this element and its paired end occur without a matching move destination container (§2.13.5.28; §2.13.5.27), then moved content in this region shall be treated as if it was deleted

- If multiple start elements exist with the same id attribute value, then the each instance in the document shall be matched with an end in document order, and unmatched starts (no corresponding end) shall be handled as described above.
- If multiple move source containers surround the same text, the last valid container (determined by the location of the container start elements, in document order) should be the container associated with that text.

[*Example*: Consider a WordprocessingML document in which the first paragraph contains two sentences, and the first sentence is moved before the second sentence, and this move is tracked as a revision, as follows (in this image, green underline indicates the move destination and the green strikethrough indicates the move source location):



This document has the sentence `Some moved text.` moved to the first sentence in the document. This revision is represented using the following WordprocessingML:

```
<w:p>
  <w:moveToRangeStart w:id="0" … w:name="move1" />
  <w:moveTo w:id="1" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveTo>
  <w:moveToRangeEnd w:id="0" />
  <w:r>
    <w:t xml:space="preserve">Some text.</w:t>
  </w:r>
  <w:moveFromRangeStart w:id="2" … w:name="move1" />
  <w:moveFrom w:id="3" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveFrom>
  <w:moveFromRangeEnd w:id="2" />
</w:p>
```

The moveFromRangeStart element specifies the start of the move source container within which all moved content is part of the move named `move1`. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32);

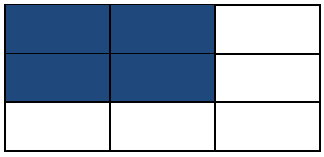
Parent Elements
endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

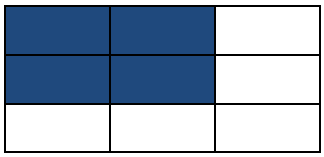
Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`  …`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

colFirst (First Table Column Covered By Bookmark)	Specifies the zero-based index of the first column in this row which shall be part of this bookmark. When a bookmark is contained within a table, it is possible for that bookmark to only cover cells within a certain column and row range within that table, by specifying: • The first row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkStart element in the first table cell in that row. • The first column included in the bookmark for each of the specified row(s) via this attribute. • The last column included in the bookmark for each of the specified row(s) via the colLast attribute. • The last row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkEnd element at the end of that table row. If this attribute appears, then the colLast attribute must also appear (regardless of where this bookmark is located) or the document shall be considered non-conformant. If this attibute and its pair occur on a bookmark which is not contained in a table, then their values should be ignored. If this value exceeds the value of colLast or the number of columns in the table, then both values should be ignored.

Attributes	Description

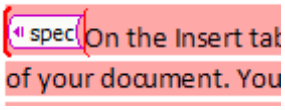
	[*Example*: Consider a three row by three column table where a table bookmark shall be applied to the contents of the first two cells in the first two rows in the table (the cells shaded below):
	 
	This bookmark would be specified using the following WordprocessingML for the table's conents:
	
	`<w:tbl>`
	`  …`
	`  <w:tr>`
	`  <w:tc>`
	`  <w:bookMarkStart w:colFirst="0" w:colLast="1" w:id="0"`
	`w:name="table"/>`
	`  <w:p/>`
	`  </w:tc>`
	`  <w:tc>`
	`  <w:p/>`
	`  </w:tc>`
	`  <w:tc>`
	`  <w:p/>`
	`  </w:tc>`
	`  </w:tr>`
	`  <w:tr>`
	`  <w:tc>`
	`  <w:p/>`
	`  </w:tc>`
	`  <w:tc>`
	`  <w:p/>`
	`  </w:tc>`
	`  <w:tc>`
	`  <w:p/>`
	`  </w:tc>`
	`  <w:bookmarkEnd w:id="0" />`
	`  </w:tr>`
	`  <w:tr>`
	`  <w:tc>`
	`  <w:p/>`
	`  </w:tc>`
	`  <w:tc>`
	`  <w:p/>`
	`  </w:tc>`

Attributes	Description
	<w:tc> <w:p/> </w:tc> </w:tr> </w:tbl> The colFirst attribute specifies that all columns starting with the first column shall be included in the table bookmark. This will apply starting with the first row and ending with the second row (the two rows within the bookmark's start and end). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
colLast (Last Table Column Covered By Bookmark)	Specifies the zero-based index of the last column in this row which shall be part of this bookmark. When a bookmark is contained within a table, it is possible for that bookmark to only cover cells within a certain column and row range within that table, by specifying: • The first row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkStart element in the first table cell in that row. • The first column included in the bookmark for each of the specified row(s) via the colFirst attribute. • The last column included in the bookmark for each of the specified row(s) via this attribute. • The last row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkEnd element at the end of that table row. If this attribute appears, then the colFirst attribute must also appear (regardless of where this bookmark is located) or the document shall be considered non-conformant. If this attibute and its pair occur on a bookmark which is not contained in a table, then their values should be ignored. If this value does not equal or exceed the value of colFirst or the number of columns in the table, then both values should be ignored. [*Example*: Consider a three row by three column table where a table bookmark shall be applied to the contents of the first two cells in the first two rows in the table (the cells shaded below):  This bookmark would be specified using the following WordprocessingML for the table's conents:

Attributes	Description
	<w:tbl> … <w:tr> <w:tc> <w:bookMarkStart w:colFirst="0" w:colLast="1" w:id="0" w:name="table"/> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> </w:tr> <w:tr> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> <w:bookmarkEnd w:id="0" /> </w:tr> <w:tr> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> </w:tr> </w:tbl> The colLast attribute specifies that the last column that shall be included in the table bookmark is the second column. This will apply starting with the first row and ending with the second row (the two rows within the bookmark's start and end). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
date (Annotation	Specifies the date information for an annotation within a WordprocessingML document.

Attributes	Description

Date)	The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`…`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

displacedByCustomXml (Annotation Marker Relocated For Custom XML Markup)	Specifies that the parent annotation's placement shall be directly linked with the location of the physical presentation of a custom XML element in the document. This element only has an effect when the custom XML element is block-level (i.e. surrounds an entire paragraph), as in this scenario the logical and physical placement of the annotation and custom XML element may differ. Specifically, in this case, the custom XML is presented *around* the block-level object it encloses (the paragraph, table, table row, or table cell), but is physically represented within that same object (i.e. within the paragraph, table, table row or table cell). This requirement stems from the fact that there is no location for the location of the annotation within the document at its logical location (around a table, for example). If this element is omitted, then the annotation shall be anchored inside of all block-level custom XML elements in the paragraph. If this element is present, but no block-level custom XML tag is located at the position it specifies (before or after), then it shall be ignored. [*Example*: Consider a paragraph with block level custom XML markup and two comment anchor annotations (one before and one after the custom XML element's physical representation), as follows:  Since all three of these items are around the entire paragraph, they are stored outside of the paragraph. However, in order to ensure that their relative positions are stored correctly, any annotation which shall be displaced by the physical custom XML element

Attributes	Description

	specifies this information, resulting in the following WordprocessingML:
	
	`<w:commentRangeStart w:id="0" />`
	`<w:commentRangeStart w:id="1" w:displaced byCustomXml="next" />`
	`<w:customXml w:element="spec" … />`
	`<w:p>`
	`…`
	`</w:p>`
	
	The displacedByCustomXml attribute specifies that even though all three of these items are around the paragraph and will be moved inside the paragraph to be represented physically, the comment with ID 0 shall be inside the custom XML, but the comment with ID 1 shall be displaced to stay outside of the relative location of the next custom XML element (the spec element). *end example*]
	
	The possible values for this attribute are defined by the ST_DisplacedByCustomXml simple type (§2.18.17).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

name (Bookmark Name)	Specifies the bookmark name.
	
	If multiple bookmarks in a document share the same name, then the first bookmark (defined by the location of the bookmarkStart element in document order) shall be maintained, and all subsequent bookmarks should be ignored.
	
	[*Example*: Consider the following XML for a bookmark around a single word:
	
	`<w:p>`
	`<w:bookmarkStart w:id="0" w:name="place" />`
	`<w:r>`
	`<w:t>Seattle</w:t>`

Attributes	Description
	</w:r> <w:bookmarkEnd w:id="0" /> </w:p> The name attribute specifies that the name for this bookmark is place. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MoveBookmark">
   <complexContent>
      <extension base="CT_Bookmark">
         <attribute name="author" type="ST_String" use="required"/>
         <attribute name="date" type="ST_DateTime" use="required"/>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.25 moveTo (Move Destination Paragraph)

This element specifies that the parent paragraph has been moved to this location and tracked as a revision. This does not imply anything about the revision state of the contents of the paragraph, and applies only to the existence of the paragraph as its own unique paragraph.

The following restrictions shall be applied to this content:

- If this element occurs outside of a valid move destination container (§2.13.5.28; §2.13.5.27) for which a matching move source container (§2.13.5.24; §2.13.5.23) exists in the document, then content in this region shall be treated as inserted, rather than moved.

[*Example*: Consider a WordprocessingML document in which a paragraph of text is moved down in the document. This moved paragraph would be represented using the following WordprocessingML markup:

```
<w:moveToRangeStart w:id="0" w:name="aMove"/>
<w:p>
  <w:pPr>
    <w:rPr>
      <w:moveTo w:id="1" … />
    </w:rPr>
  </w:pPr>
  …
</w:p>
</w:moveToRangeEnd w:id="0"/>
```

The moveTo element as a child of the run properties of the paragraph mark specify that this paragraph mark was part of the content which was moved in the document. This implies nothing about the contents, since they

may have been added later and not tracked as a revision (they must be marked as a move using the moveTo element (§2.13.5.26) around the run content). *end example*]

Parent Elements
rPr (§2.3.1.29); rPr (§2.3.1.30)

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`…`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`…`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element. If this attribute is omitted, then the document is non-conformant.

Attributes	Description

	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TrackChange">
   <complexContent>
      <extension base="CT_Markup">
         <attribute name="author" type="ST_String" use="required"/>
         <attribute name="date" type="ST_DateTime" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

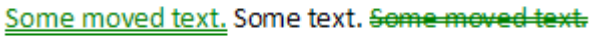
2.13.5.26 moveTo (Move Destination Run Content)

This element specifies that the inline-level content contained within it shall be treated as content which has been moved to this location and tracked as a revision.

The following restrictions shall be applied to this content:

- If this element occurs outside of a valid move destination container (§2.13.5.28; §2.13.5.27) for which a matching move source container (§2.13.5.24; §2.13.5.23) exists in the document, then content in this region shall be treated as inserted, rather than moved.

[*Example*: Consider a WordprocessingML document in which the first paragraph contains two sentences, and the first sentence is moved before the second sentence, and this move is tracked as a revision, as follows (in this image, green underline indicates the move destination and the green strikethrough indicates the move source location):



This document has the sentence `Some moved text.` moved to the first sentence in the document. This revision is represented using the following WordprocessingML:

```
<w:p>
  <w:moveToRangeStart w:id="0" … w:name="move1" />
  <w:moveTo w:id="1" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveTo>
  <w:moveToRangeEnd w:id="0" />
  <w:r>
    <w:t xml:space="preserve">Some text.</w:t>
  </w:r>
  <w:moveFromRangeStart w:id="2" … w:name="move1" />
  <w:moveFrom w:id="3" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveFrom>
  <w:moveFromRangeEnd w:id="2" />
</w:p>
```

The moveTo element specifies that all of the inline-level content contained within shall be revision marked as content which was moved to its current location in the document. Because this moved content is contained within a complete move destination container (moveToRangeStart and moveToRangeEnd) with a corresponding move source, this content is tracked as a move. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32); endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Child Elements	Subclause
acc (Accent)	§7.1.2.1
bar (Bar)	§7.1.2.7
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
borderBox (Border-Box Function)	§7.1.2.11
box (Box Function)	§7.1.2.13

Child Elements	Subclause
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Inline-Level Custom XML Element)	§2.5.1.5
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
d (Delimiter Function)	§7.1.2.24
del (Deleted Run Content)	§2.13.5.12
eqArr (Equation-Array Function)	§7.1.2.34
f (Fraction Function)	§7.1.2.36
func (Function Apply Function)	§7.1.2.39
groupChr (Group-Character Function)	§7.1.2.41
ins (Inserted Run Content)	§2.13.5.20
limLow (Lower-Limit Function)	§7.1.2.54
limUpp (Upper-Limit Function)	§7.1.2.56
m (Matrix Function)	§7.1.2.60
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
nary (n-ary Operator Function)	§7.1.2.70
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
phant (Phantom Function)	§7.1.2.81
proofErr (Proofing Error Anchor)	§2.13.8.1

Child Elements	Subclause
r (Text Run)	§2.3.2.23
r (Run)	§7.1.2.87
rad (Radical Function)	§7.1.2.88
sdt (Inline-Level Structured Document Tag)	§2.5.2.29
smartTag (Inline-Level Smart Tag)	§2.5.1.9
sPre (Pre-Sub-Superscript Function)	§7.1.2.99
sSub (Subscript Function)	§7.1.2.101
sSubSup (Sub-Superscript Function)	§7.1.2.103
sSup (Superscript Function)	§7.1.2.105

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`  …`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at

Attributes	Description
	10:00 AM, which may be used as desired. *end example*] The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RunTrackChange">
    <complexContent>
        <extension base="CT_TrackChange">
            <choice minOccurs="0" maxOccurs="unbounded">
                <group ref="EG_ContentRunContent"/>
                <group ref="m:EG_OMathMathElements"/>
            </choice>
        </extension>
    </complexContent>
</complexType>
```

2.13.5.27 moveToRangeEnd (Move Destination Location Container - End)

This element specifies the end of a region whose move destination contents are part of a single named move. When a move source is stored as a revision in a WordprocessingML document, two pieces of information must be stored about that move destination:

- A set of pieces of content which were moved - both inline-level content (§2.13.5.26) and paragraphs (§2.13.5.25)
- A move destination container (or "bookmark") which specifies that all content within it which marked as a move destination is part of a single named move. The name attribute on the move container links a group of move destination content with the corresponding group of move source content.

This element defines the end of the latter piece of the move revision data - the container. The id attribute on this element shall be used to link this element with the corresponding start of a move destination container in the document.

The following restrictions are applied to the use of this element:

- If this element occurs without a corresponding moveToRangeStart element (§2.13.5.28) with a matching id attribute value, then it shall be ignored and no move source container exists
- If this element and its paired end occur without a matching move source container (§2.13.5.24; §2.13.5.23), then moved content in this region shall be treated as if it was inserted
- If multiple move destination containers surround the same text, the last valid container (determined by the location of the container start elements, in document order) should be the container associated with that text.

[*Example*: Consider a WordprocessingML document in which the first paragraph contains two sentences, and the first sentence is moved before the second sentence, and this move is tracked as a revision, as follows (in this image, green underline indicates the move destination and the green strikethrough indicates the move source location):

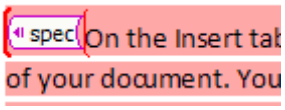
This document has the sentence `Some moved text.` moved to the first sentence in the document. This revision is represented using the following WordprocessingML:

```
<w:p>
  <w:moveToRangeStart w:id="0" … w:name="move1" />
  <w:moveTo w:id="1" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveTo>
  <w:moveToRangeEnd w:id="0" />
  <w:r>
    <w:t xml:space="preserve">Some text.</w:t>
  </w:r>
  <w:moveFromRangeStart w:id="2" … w:name="move1" />
  <w:moveFrom w:id="3" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveFrom>
  <w:moveFromRangeEnd w:id="2" />
</w:p>
```

The moveToRangeEnd element specifies the end of the move destination container within which all moved content is part of the move named `move1`. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32); endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Attributes	Description

displacedByCustomXml (Annotation Marker Relocated For Custom XML Markup)	Specifies that the parent annotation's placement shall be directly linked with the location of the physical presentation of a custom XML element in the document. This element only has an effect when the custom XML element is block-level (i.e. surrounds an entire paragraph), as in this scenario the logical and physical placement of the annotation and custom XML element may differ.
	
	Specifically, in this case, the custom XML is presented *around* the block-level object it encloses (the paragraph, table, table row, or table cell), but is physically represented within that same object (i.e. within the paragraph, table, table row or table cell). This requirement stems from the fact that there is no location for the location of the annotation within the document at its logical location (around a table, for example).
	
	If this element is omitted, then the annotation shall be anchored inside of all block-level custom XML elements in the paragraph. If this element is present, but no block-level custom XML tag is located at the position it specifies (before or after), then it shall be ignored.
	
	[*Example*: Consider a paragraph with block level custom XML markup and two comment anchor annotations (one before and one after the custom XML element's physical representation), as follows:
	
	 
	
	Since all three of these items are around the entire paragraph, they are stored outside of the paragraph. However, in order to ensure that their relative positions are stored correctly, any annotation which shall be displaced by the physical custom XML element specifies this information, resulting in the following WordprocessingML:
	
	`<w:commentRangeStart w:id="0" />`
	`<w:commentRangeStart w:id="1" w:displaced byCustomXml="next" />`

Attributes	Description
	<w:customXml w:element="spec" … /> <w:p> … </w:p> The displacedByCustomXml attribute specifies that even though all three of these items are around the paragraph and will be moved inside the paragraph to be represented physically, the comment with ID 0 shall be inside the custom XML, but the comment with ID 1 shall be displaced to stay outside of the relative location of the next custom XML element (the spec element). *end example*] The possible values for this attribute are defined by the ST_DisplacedByCustomXml simple type (§2.18.17).
id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element. If this attribute is omitted, then the document is non-conformant. [*Example*: Consider an annotation represented using the following WordprocessingML fragment: <w:… w:id="1" … > … </w:…> The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MarkupRange">
   <complexContent>
      <extension base="CT_Markup">
         <attribute name="displacedByCustomXml" type="ST_DisplacedByCustomXml" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.28 moveToRangeStart (Move Destination Location Container - Start)

This element specifies the start of the region whose move destination contents are part of a single named move. When a move destination is stored as a revision in a WordprocessingML document, two pieces of information must be stored about that move destination:

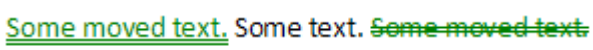
- A set of pieces of content which were moved - both inline-level content (§2.13.5.26) and paragraphs (§2.13.5.25)
- A move destination container (or "bookmark") which specifies that all content within it which marked as a move destination is part of a single named move. The name attribute on the move container links a group of move destination content with the corresponding group of move source content.

This element defines the start of the latter piece of the move revision data - the container. The id attribute on this element shall be used to link this element with the corresponding end of a move destination container in the document.

The following restrictions are applied to the use of this element

- If this element occurs without a corresponding moveToRangeEnd element (§2.13.5.27) with a matching id attribute value, then it shall be ignored and no move source container exists
- If this element and its paired end occur without a matching move source container (§2.13.5.24; §2.13.5.23), then moved content in this region shall be treated as if it was inserted
- If multiple start elements exist with the same id attribute value, then the each instance in the document shall be matched with an end in document order, and unmatched starts (no corresponding end) shall be handled as described above.
- If multiple move destination containers surround the same text, the last valid container (determined by the location of the container start elements, in document order) should be the container associated with that text.

[*Example*: Consider a WordprocessingML document in which the first paragraph contains two sentences, and the first sentence is moved before the second sentence, and this move is tracked as a revision, as follows (in this image, green underline indicates the move destination and the green strikethrough indicates the move source location):



This document has the sentence `Some moved text.` moved to the first sentence in the document. This revision is represented using the following WordprocessingML:

```
<w:p>
  <w:moveToRangeStart w:id="0" … w:name="move1" />
  <w:moveTo w:id="1" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveTo>
  <w:moveToRangeEnd w:id="0" />
  <w:r>
    <w:t xml:space="preserve">Some text.</w:t>
  </w:r>
  <w:moveFromRangeStart w:id="2" … w:name="move1" />
  <w:moveFrom w:id="3" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveFrom>
  <w:moveFromRangeEnd w:id="2" />
</w:p>
```

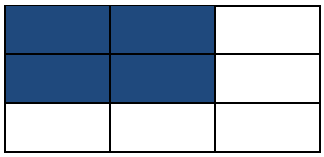
The moveToRangeStart element specifies the start of the move destination container within which all moved content is part of the move named `move1`. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32); endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Attributes	Description

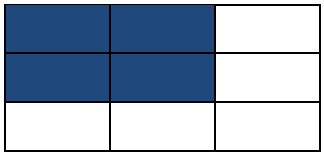
author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`…`
	`</w:…>`

Attributes	Description
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

colFirst (First Table Column Covered By Bookmark)	Specifies the zero-based index of the first column in this row which shall be part of this bookmark.
	
	When a bookmark is contained within a table, it is possible for that bookmark to only cover cells within a certain column and row range within that table, by specifying:
	• The first row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkStart element in the first table cell in that row.
	• The first column included in the bookmark for each of the specified row(s) via this attribute.
	• The last column included in the bookmark for each of the specified row(s) via the colLast attribute.
	• The last row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkEnd element at the end of that table row.
	
	If this attribute appears, then the colLast attribute must also appear (regardless of where this bookmark is located) or the document shall be considered non-conformant. If this attibute and its pair occur on a bookmark which is not contained in a table, then their values should be ignored. If this value exceeds the value of colLast or the number of columns in the table, then both values should be ignored.
	
	[*Example*: Consider a three row by three column table where a table bookmark shall be applied to the contents of the first two cells in the first two rows in the table (the cells shaded below):
	
	 
	
	This bookmark would be specified using the following WordprocessingML for the table's conents:
	
	`<w:tbl>`
	`…`
	`<w:tr>`
	`<w:tc>`
	`<w:bookMarkStart w:colFirst="0" w:colLast="1" w:id="0" w:name="table"/>`
	`<w:p/>`

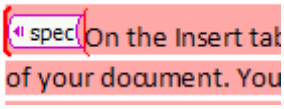
Attributes	Description
	</w:tc> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> </w:tr> <w:tr> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> <w:bookmarkEnd w:id="0" /> </w:tr> <w:tr> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> </w:tr> </w:tbl> The colFirst attribute specifies that all columns starting with the first column shall be included in the table bookmark. This will apply starting with the first row and ending with the second row (the two rows within the bookmark's start and end). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
colLast (Last Table Column Covered By Bookmark)	Specifies the zero-based index of the last column in this row which shall be part of this bookmark. When a bookmark is contained within a table, it is possible for that bookmark to only cover cells within a certain column and row range within that table, by specifying: • The first row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkStart element in the first table cell in that row.

Attributes	Description

	• The first column included in the bookmark for each of the specified row(s) via the colFirst attribute.
	• The last column included in the bookmark for each of the specified row(s) via this attribute.
	• The last row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkEnd element at the end of that table row.
	
	If this attribute appears, then the colFirst attribute must also appear (regardless of where this bookmark is located) or the document shall be considered non-conformant. If this attibute and its pair occur on a bookmark which is not contained in a table, then their values should be ignored. If this value does not equal or exceed the value of colFirst or the number of columns in the table, then both values should be ignored.
	
	[*Example*: Consider a three row by three column table where a table bookmark shall be applied to the contents of the first two cells in the first two rows in the table (the cells shaded below):
	
	 
	
	This bookmark would be specified using the following WordprocessingML for the table's conents:
	
	`<w:tbl>`
	`  …`
	`  <w:tr>`
	`  <w:tc>`
	`  <w:bookMarkStart w:colFirst="0" w:colLast="1" w:id="0"`
	`w:name="table"/>`
	`  <w:p/>`
	`  </w:tc>`
	`  <w:tc>`
	`  <w:p/>`
	`  </w:tc>`
	`  <w:tc>`
	`  <w:p/>`
	`  </w:tc>`
	`  </w:tr>`
	`  <w:tr>`
	`  <w:tc>`
	`  <w:p/>`
	`  </w:tc>`
	`  <w:tc>`
	`  <w:p/>`

Attributes	Description
	</w:tc> <w:tc> <w:p/> </w:tc> <w:bookmarkEnd w:id="0" /> </w:tr> <w:tr> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> </w:tr> </w:tbl> The colLast attribute specifies that the last column that shall be included in the table bookmark is the second column. This will apply starting with the first row and ending with the second row (the two rows within the bookmark's start and end). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard. If this attribute is omitted, then no date information shall be associated with the parent annotation type. [*Example*: Consider a comment represented using the following WordprocessingML fragment: <w:… w:id="1" w:date="2006-01-01T10:00:00"> … </w:…> The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*] The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).
displacedByCustomXml (Annotation Marker Relocated	Specifies that the parent annotation's placement shall be directly linked with the location of the physical presentation of a custom XML element in the document. This element only has an effect when the custom XML element is block-level (i.e. surrounds an entire

Attributes	Description

For Custom XML Markup)	paragraph), as in this scenario the logical and physical placement of the annotation and custom XML element may differ.
	
	Specifically, in this case, the custom XML is presented *around* the block-level object it encloses (the paragraph, table, table row, or table cell), but is physically represented within that same object (i.e. within the paragraph, table, table row or table cell). This requirement stems from the fact that there is no location for the location of the annotation within the document at its logical location (around a table, for example).
	
	If this element is omitted, then the annotation shall be anchored inside of all block-level custom XML elements in the paragraph. If this element is present, but no block-level custom XML tag is located at the position it specifies (before or after), then it shall be ignored.
	
	[*Example*: Consider a paragraph with block level custom XML markup and two comment anchor annotations (one before and one after the custom XML element's physical representation), as follows:
	
	 
	
	Since all three of these items are around the entire paragraph, they are stored outside of the paragraph. However, in order to ensure that their relative positions are stored correctly, any annotation which shall be displaced by the physical custom XML element specifies this information, resulting in the following WordprocessingML:
	
	`<w:commentRangeStart w:id="0" />`
	`<w:commentRangeStart w:id="1" w:displaced byCustomXml="next" />`
	`<w:customXml w:element="spec" … />`
	`<w:p>`
	`…`
	`</w:p>`
	
	The displacedByCustomXml attribute specifies that even though all three of these items are around the paragraph and will be moved inside the paragraph to be represented physically, the comment with ID 0 shall be inside the custom XML, but the comment with ID 1 shall be displaced to stay outside of the relative location of the next custom XML element (the spec element). *end example*]
	
	The possible values for this attribute are defined by the ST_DisplacedByCustomXml simple type (§2.18.17).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element. If this attribute is omitted, then the document is non-conformant.

Attributes	Description

	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

name (Bookmark Name)	Specifies the bookmark name.
	
	If multiple bookmarks in a document share the same name, then the first bookmark (defined by the location of the bookmarkStart element in document order) shall be maintained, and all subsequent bookmarks should be ignored.
	
	[*Example*: Consider the following XML for a bookmark around a single word:
	
	`<w:p>`
	`<w:bookmarkStart w:id="0" w:name="place" />`
	`<w:r>`
	`<w:t>Seattle</w:t>`
	`</w:r>`
	`<w:bookmarkEnd w:id="0" />`
	`</w:p>`
	
	The name attribute specifies that the name for this bookmark is `place`. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MoveBookmark">
   <complexContent>
      <extension base="CT_Bookmark">
         <attribute name="author" type="ST_String" use="required"/>
         <attribute name="date" type="ST_DateTime" use="required"/>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.29 numberingChange (Previous Numbering Field Properties)

This element specifies the previous state of the numbering displayed by a LISTNUM field (§2.16.5.40) within a WordprocessingML document when additional LISTNUM fields are added and revisions are being tracked.

[*Rationale*: The legacy numbering mechanism provided by the LISTNUM field relies on the presence of fields in the run content of the document, rather than being a paragraph property (as numbering typically is represented). For this reason, these fields must store their previous state as a unique revision type on the field character of the numbering field. *end rationale*]

If this element is supplied for a field which is not of type LISTNUM as defined by its field codes (§2.16.5), then this property shall be ignored.

[*Example*: Consider the following paragraph containing a single LISTNUM field, as follows:

Some 1. text

If another LISTNUM field is added before it in the document, resulting in its evaluation to a different number, as follows:

Some ~~1.~~2. text

This revision to the field result would be stored as follows in the WordprocessingML:

```
<w:fldChar w:type="begin">
  <w:numberingChange w:id="0" … w:original="1." />
</w:fldChar>
<w:r>
  <w:instrText>LISTNUM</w:instrText>
</w:r>
<w:fldChar w:type="separate"/>
<w:r>
  <w:t>2.</w:t>
</w:r>
<w:fldChar w:type="end" />
```

The numberingChange element specifies that the numbering resulting from this LISTNUM field was modified and this change was tracked as a revision. The previous numbering result of `1.` is cached in the original attribute. *end example*]

For numbering fields, the original attribute shall specify the previous numbering displayed by the parent LISTNUM field within a WordprocessingML document. This information is a performance-enhancing cache of the state of the numbering before the revision to allow applications to show the previous state without having to recalculate all of the LISTNUM fields in the document.

If this attribute is omitted, then no previous numbering value is implied and applications may choose to calculate this value, or display no previous numbering value.

[*Example*: Consider the following paragraph containing a single LISTNUM field with a revision, as follows:

Some ~~1.~~2. text

This revision to the field result would be stored as follows in the WordprocessingML:

```
<w:fldChar w:type="begin">
  <w:numberingChange w:id="0" … w:original="1." />
</w:fldChar>
```

The original attribute specifies that the previous numbering value of the field was `1.` *end example*]

Parent Elements
fldChar (§2.16.18)

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`…`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`…`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at

Attributes	Description
	10:00 AM, which may be used as desired. *end example*] The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`  …`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

original (Previous Numbering Value)	Specifies the previous numbering displayed by the parent numbering change revision. Its format is specified by the parent element.
	
	If this attribute is omitted, then no previous numbering value is implied and applications may choose to calculate this value, or display no previous numbering value.
	
	[*Example*: Consider the following paragraph containing a single LISTNUM field with a revision, as follows:
	
	Some ~~1.~~2. text
	
	This revision to the field result would be stored as follows in the WordprocessingML:
	
	`<w:fldChar w:type="begin">`
	`  <w:numberingChange w:id="0" … w:original="1." />`
	`</w:fldChar>`
	
	The original attribute specifies that the previous numbering value of the field was 1. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TrackChangeNumbering">
   <complexContent>
      <extension base="CT_TrackChange">
         <attribute name="original" type="ST_String" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.30 numberingChange (Previous Paragraph Numbering Properties)

This element specifies the previous state of the numbering on a paragraph when revisions are being tracked.

[*Rationale*: This mechanism is simply used to provide storage for revisions to numbering produced by legacy word processing applications, and applications are encouraged to use the pPrChange element to store these changes as changes to the paragraph properties instead. *end rationale*]

[*Example*: Consider the following list using Arabic numerals as the numbering, as follows:

1. one
2. two
3. three

Consider a revision where the numbering definition is changed from Arabic numerals to Roman numerals, as follows:

~~1~~ i. one
~~2~~ ii. two
~~3~~ iii. three

This revision to the numbering definition would be stored as follows in the WordprocessingML:

```
<w:p>
    <w:pPr>
        <w:numPr>
            <w:ilvl w:val="0" />
            <w:numId w:val="1" />
            <w:numberingChange w:id="0" … w:original="%1:1:0:." />
        </w:numPr>
    </w:pPr>
    <w:r>
        <w:t>one</w:t>
    </w:r>
</w:p>
<w:p>
    <w:pPr>
        <w:numPr>
```

```
                <w:ilvl w:val="0" />
                <w:numId w:val="1" />
                <w:numberingChange w:id="1" … w:original="%1:2:0:." />
          </w:numPr>
    </w:pPr>
    <w:r>
          <w:t>two</w:t>
    </w:r>
</w:p>
<w:p>
    <w:pPr>
          <w:numPr>
                <w:ilvl w:val="0" />
                <w:numId w:val="1" />
                <w:numberingChange w:id="2" … w:original="%1:3:0:." />
          </w:numPr>
    </w:pPr>
    <w:r>
          <w:t>three</w:t>
    </w:r>
</w:p>
```

The numberingChange element specifies that the numbering definition was modified and this change was tracked as a revision. The previous Arabic numeral numbering definition is cached in the original attribute. *end example*]

For paragraph numbering, the original attribute shall specify the previous numbering definition for an individual paragraph of text within a WordprocessingML document while revisions are being tracked.

The value of original is represented as separate numbering level definitions defined as follows:

```
<%[numbering level]:[nfc value]:[numbering format]:[separator]>[repeat if more
than one level]
```

where

- numbering level – The level for which the numbering definition is defined
- nfc value – The value of the numbering style at the specific numbering level
- numbering format – The nfc value of the numbering format, as referenced in the table below.
- separator – The separator used to separate the numbering level definitions

The numbering format values are mapped as follows:

nfc Value	ST_NumberFormat enumeration equivalent
0	decimal

nfc Value	ST_NumberFormat enumeration equivalent
1	upperRoman
2	lowerRoman
3	upperLetter
4	lowerLetter
5	ordinal
6	cardinalText
7	ordinalText
8	hex
9	chicago
10	ideographDigital
11	japaneseCounting
12	Aiueo
13	Iroha
14	decimalFullWidth
15	decimalHalfWidth
16	japaneseLegal
17	japaneseDigitalTenThousand
18	decimalEnclosedCircle
19	decimalFullWidth2
20	aiueoFullWidth
21	irohaFullWidth
22	decimalZero
23	bullet
24	ganada
25	chosung
26	decimalEnclosedFullstop
27	decimalEnclosedParen
28	decimalEnclosedCircleChinese
29	ideographEnclosedCircle
30	ideographTraditional
31	ideographZodiac
32	ideographZodiacTraditional
33	taiwaneseCounting
34	ideographLegalTraditional
35	taiwaneseCountingThousand
36	taiwaneseDigital
37	chineseCounting
38	chineseLegalSimplified

nfc Value	ST_NumberFormat enumeration equivalent
39	chineseCountingThousand
40	Application-defined. May be ignored.
41	koreanDigital
42	koreanCounting
43	koreanLegal
44	koreanDigital2
45	hebrew1
46	arabicAlpha
47	hebrew2
48	arabicAbjad
49	hindiVowels
50	hindiConsonants
51	hindiNumbers
52	hindiCounting
53	thaiLetters
54	thaiNumbers
55	thaiCounting
56	vietnameseCounting
57	numberInDash
58	russianLower
59	russianUpper
60 or above	Application-defined. May be ignored.

[*Example*: Consider the following numbered paragraph where the numbering definition has changed while revisions are being tracked, as follows:

~~1.~~ ~~i.~~ ~~1.~~ 1. Three

This revision to the numbered paragraph would be stored as follows in the WordprocessingML:

```
<w:numPr>
  …
  <w:numberingChange … w:original="%1:1:0:.%2:1:2:.%3:1:0:." />
</w:numPr>
```

In the above example there are three levels in the original numbering definition, thus three numbering level definitions are needed to represent the original numbering definition.

The first level is specified by `%1`, and says that it was number value `1` in the nfc format `0` (`arabic`).

The original attribute specifies that the previous numbering definition was made up of three levels whose value was `1.i.1.`. *end example*]

Parent Elements
numPr (§2.3.1.19)

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`…`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`…`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element. If this attribute is omitted, then the document is non-conformant.

Attributes	Description

	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

original (Previous Numbering Value)	Specifies the previous numbering displayed by the parent numbering change revision. Its format is specified by the parent element.
	
	If this attribute is omitted, then no previous numbering value is implied and applications may choose to calculate this value, or display no previous numbering value.
	
	[*Example*: Consider the following paragraph containing a single LISTNUM field with a revision, as follows:
	
	Some ~~1.~~2. text
	
	This revision to the field result would be stored as follows in the WordprocessingML:
	
	`<w:fldChar w:type="begin">`
	`<w:numberingChange w:id="0" … w:original="1." />`
	`</w:fldChar>`
	
	The original attribute specifies that the previous numbering value of the field was 1. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TrackChangeNumbering">
   <complexContent>
      <extension base="CT_TrackChange">
         <attribute name="original" type="ST_String" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.31 pPrChange (Revision Information for Paragraph Properties)

This element specifies the details about a single revision to a set of paragraph properties in a WordprocessingML document.

This element stores this revision as follows:

- The child element of this element contains the complete set of paragraph properties which were applied to this paragraph before this revision
- The attributes of this element contain information about when this revision took place (i.e. when these properties became a 'former' set of paragraph properties).

[*Example*: Consider a paragraph in a WordprocessingML document which is centered, and this change in the paragraph properties is tracked as a revision. This revision would be specified using the following WordprocessingML markup:

```
<w:pPr>
  <w:jc w:val="center"/>
  <w:pPrChange w:id="0" w:date="01-01-2006T12:00:00" w:author="John Doe">
    <w:pPr/>
  </w:pPrChange>
</w:pPr>
```

The pPrChange element specifies that there was a revision to the paragraph properties at `01-01-2006` by `John Doe`, and the previous set of paragraph properties on the paragraph were the null set (i.e. no paragraph properties explicitly present under the pPr element). *end example*]

Parent Elements
pPr (§2.7.4.2); pPr (§2.9.24); pPr (§2.7.5.1); pPr (§2.3.1.26); pPr (§2.7.7.2)

Child Elements	Subclause
pPr (Previous Paragraph Properties)	§2.3.1.25

Attributes	Description
author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document. If this attribute is omitted, then no author shall be associated with the parent annotation type. [*Example*: Consider a comment represented using the following WordprocessingML fragment: `<w:… w:id="1" w:author="Example Author">`

Attributes	Description
	… </w:…> The author attribute specifies that the author of the current annotation is Example Author, which may be used as desired. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).
date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard. If this attribute is omitted, then no date information shall be associated with the parent annotation type. [*Example*: Consider a comment represented using the following WordprocessingML fragment: <w:… w:id="1" w:date="2006-01-01T10:00:00"> … </w:…> The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*] The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).
id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element. If this attribute is omitted, then the document is non-conformant. [*Example*: Consider an annotation represented using the following WordprocessingML fragment: <w:… w:id="1" … > … </w:…> The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PPrChange">
   <complexContent>
      <extension base="CT_TrackChange">
         <sequence>
            <element name="pPr" type="CT_PPrBase" minOccurs="1"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.32 rPrChange (Revision Information for Run Properties)

This element specifies the details about a single revision to a set of run properties in a WordprocessingML document.

This element stores this revision as follows:

- The child element of this element contains the complete set of run properties which were applied to this run before this revision
- The attributes of this element contain information about when this revision took place (i.e. when these properties became a 'former' set of run properties).

[*Example*: Consider an italicized run in a WordprocessingML document which is also made bold and the latter change in the run properties is tracked as a revision. This revision would be specified using the following WordprocessingML markup:

```
<w:rPr>
  <w:b/>
  <w:i/>
  <w:rPrChange w:id="0" w:date="01-01-2006T12:00:00" w:author="John Doe">
    <w:rPr>
      <w:i/>
    </w:rPr>
  </w:rPrChange>
</w:rPr>
```

The rPrChange element specifies that there was a revision to the run properties at `01-01-2006` by `John Doe`, and the previous set of run properties was simply the italicization using the i element (§2.3.2.14). *end example*]

Parent Elements
rPr (§2.7.8.1); rPr (§2.5.2.26); rPr (§2.3.2.25); rPr (§2.7.4.4); rPr (§2.9.26); rPr (§2.5.2.27); rPr (§2.7.5.2)

Child Elements	Subclause
rPr (Previous Run Properties)	§2.3.2.26

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`…`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`…`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`

Attributes	Description
	</w:…> The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RPrChange">
   <complexContent>
      <extension base="CT_TrackChange">
         <sequence>
            <element name="rPr" type="CT_RPrOriginal" minOccurs="1"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.33 rPrChange (Revision Information for Run Properties on the Paragraph Mark)

This element specifies the details about a single revision to a set of run properties applied to a paragraph mark within a WordprocessingML document.

This element stores this revision as follows:

- The child element of this element contains the complete set of run properties which were applied to this paragraph mark before this revision
- The attributes of this element contain information about when this revision took place (i.e. when these properties became a 'former' set of run properties).

[*Example*: Consider an italicized paragraph mark in a WordprocessingML document which is also made bold and the latter change in the run properties is tracked as a revision. This revision would be specified using the following WordprocessingML markup:

```
<w:rPr>
  <w:b/>
  <w:i/>
  <w:rPrChange w:id="0" w:date="01-01-2006T12:00:00" w:author="John Doe">
    <w:rPr>
      <w:i/>
    </w:rPr>
  </w:rPrChange>
</w:rPr>
```

The rPrChange element specifies that there was a revision to the paragraph mark's run properties at 01-01-2006 by John Doe, and the previous set of run properties was simply the italicization using the i element (§2.3.2.14). *end example*]

Parent Elements
rPr (§2.3.1.29)

Child Elements	Subclause
rPr (Previous Run Properties for the Paragraph Mark)	§2.3.1.30

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`…`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is Example Author, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`…`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ParaRPrChange">
   <complexContent>
      <extension base="CT_TrackChange">
         <sequence>
            <element name="rPr" type="CT_ParaRPrOriginal" minOccurs="1"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.34 sectPrChange (Revision Information for Section Properties)

This element specifies the details about a single revision to a set of section properties in a WordprocessingML document.

This element stores this revision as follows:

- The child element of this element contains the complete set of section properties which were applied to the parent section before this revision
- The attributes of this element contain information about when this revision took place (i.e. when these properties became a 'former' set of section properties).

[*Example*: Consider a section in a WordprocessingML document which is set to be divided into three columns, and this change in the section properties is tracked as a revision. This revision would be specified using the following WordprocessingML markup:

```
<w:sectPr>
  <w:cols w:val="3"/>
  <w:sectPrChange w:id="0" w:date="01-01-2006T12:00:00" w:author="John Doe">
    <w:sectPr/>
  </w:sectPrChange>
</w:sectPr>
```

The sectPrChange element specifies that there was a revision to the section properties at `01-01-2006` by `John Doe`, and the previous set of properties on the section were the null set (i.e. no section properties explicitly present under the sectPr element). *end example*]

Parent Elements
sectPr (§2.6.18); sectPr (§2.6.19)

Child Elements	Subclause
sectPr (Previous Section Properties)	§2.6.17

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`…`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard. If this attribute is omitted, then no date information shall be associated with the parent annotation type. [*Example*: Consider a comment represented using the following WordprocessingML fragment:

Attributes	Description
	<w:… w:id="1" w:date="2006-01-01T10:00:00"> … </w:…> The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*] The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).
id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element. If this attribute is omitted, then the document is non-conformant. [*Example*: Consider an annotation represented using the following WordprocessingML fragment: <w:… w:id="1" … > … </w:…> The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SectPrChange">
   <complexContent>
      <extension base="CT_TrackChange">
         <sequence>
            <element name="sectPr" type="CT_SectPrBase" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.35 tblGridChange (Revision Information for Table Grid Column Definitions)

This element specifies the details about a single revision to a table's grid column definitions within a WordprocessingML document.

This element stores this revision as follows:

- The child element of this element contains the definition of the table grid which was applied to the parent table before this revision

[*Example*: Consider a two column table in a WordprocessingML document which has the width of its first column significantly reduced, and this change in the table grid is tracked as a revision. This revision would be specified using the following WordprocessingML markup:

```
<w:tblGrid>
  <w:gridCol w:w="1548" />
  <w:gridCol w:w="8028" />
  <w:tblGridChange w:id="1">
    <w:tblGrid>
      <w:gridCol w:w="4788" />
      <w:gridCol w:w="4788" />
    </w:tblGrid>
  </w:tblGridChange>
</w:tblGrid>
```

The tblGridChange element specifies that there was a revision to the table grid, and the previous table grid had both columns with a width of 4788 twentieths of a point, vs. their current widths of 1548 and 8028 twentieths of a point respectively. *end example*]

Parent Elements
tblGrid (§2.4.44)

Child Elements	Subclause
tblGrid (Previous Table Grid)	§2.4.45

Attributes	Description

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`  …`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblGridChange">
   <complexContent>
      <extension base="CT_Markup">
         <sequence>
            <element name="tblGrid" type="CT_TblGridBase"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.36 tblPrChange (Revision Information for Table Properties)

This element specifies the details about a single revision to a set of table properties in a WordprocessingML document.

This element stores this revision as follows:

- The child element of this element contains the complete set of table properties which were applied to the parent table before this revision
- The attributes of this element contain information about when this revision took place (i.e. when these properties became a 'former' set of table properties).

[*Example*: Consider a table in a WordprocessingML document which has the associated table style changed from `LightList` to `LightShading`, and this change in the table properties is tracked as a revision. This revision would be specified using the following WordprocessingML markup:

```
<w:tblPr>
  <w:tblStyle w:val="LightShading"/>
  <w:tblW w:w="0" w:type="auto"/>
  <w:tblLook w:val="04A0"/>
  <w:tblPrChange w:id="0" w:author="Tristan Davis" w:date="2006-06-
01T13:39:00Z">
    <w:tblPr>
      <w:tblStyle w:val="LightList"/>
      <w:tblW w:w="0" w:type="auto"/>
      <w:tblLook w:val="04A0"/>
    </w:tblPr>
  </w:tblPrChange>
</w:tblPr>
```

The tblPrChange element specifies that there was a revision to the table properties at `2006-06-01` by `Tristan Davis`, and the previous set of properties on the table was the set specifies in the child tblPr element (including the table style of `LightList`). *end example*]

Parent Elements

Parent Elements
tblPr (§2.4.55)

Child Elements	Subclause
tblPr (Previous Table Properties)	§2.4.56

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`…`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`…`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.

Attributes	Description

	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblPrChange">
   <complexContent>
      <extension base="CT_TrackChange">
         <sequence>
            <element name="tblPr" type="CT_TblPrBase"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.37 tblPrExChange (Revision Information for Table-Level Property Exceptions)

This element specifies the details about a single revision to a set of table-level property exceptions in a WordprocessingML document.

This element stores this revision as follows:

- The child element of this element contains the complete set of table-level property exceptions which were applied to the parent table row before this revision
- The attributes of this element contain information about when this revision took place (i.e. when these properties became a 'former' set of table-level exception properties).

[*Example*: Consider a set of table rows which are part of a table in a WordprocessingML document, have table-level property exceptions, and this change in the table-level properties to a fixed table width of ten inches is tracked as a revision. This revision would be specified using the following WordprocessingML markup:

```
<w:tblPrEx>
  <w:tblW w:w="14400" w:type="dxa"/>
  <w:tblPrExChange w:id="0" w:author="Tristan Davis" w:date="2006-06-
01T13:39:00Z">
    <w:tblPrEx>
      <w:tblW w:w="0" w:type="auto"/>
    </w:tblPrEx>
  </w:tblPrExChange>
</w:tblPrEx>
```

The tblPrExChange element specifies that there was a revision to the table-level property exceptions at `2006-06-01` by `Tristan Davis`, and the previous set of table-level property exceptions set specifies in the child tblPrEx element. *end example*]

Parent Elements
tblPrEx (§2.4.57)

Child Elements	Subclause
tblPrEx (Previous Table-Level Property Exceptions)	§2.4.58

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`…`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard. If this attribute is omitted, then no date information shall be associated with the parent annotation type.

Attributes	Description

	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`…`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TblPrExChange">
   <complexContent>
      <extension base="CT_TrackChange">
         <sequence>
            <element name="tblPrEx" type="CT_TblPrExBase" minOccurs="1"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.13.5.38 tcPrChange (Revision Information for Table Cell Properties)

This element specifies the details about a single revision to a set of table cell properties in a WordprocessingML document.

This element stores this revision as follows:

- The child element of this element contains the complete set of table cell properties which were applied to the parent table before this revision
- The attributes of this element contain information about when this revision took place (i.e. when these properties became a 'former' set of table cell properties).

[*Example*: Consider a table cell in a WordprocessingML document which has a change in the table cell properties that is tracked as a revision. This revision would be specified using the following WordprocessingML markup:

```
<w:tcPr>
  <w:cnfStyle w:val="001000000000"/>
  <w:tcW w:w="3192" w:type="dxa"/>
  <w:tcPrChange w:id="8" w:author="Tristan Davis" w:date="2006-06-01T13:39:00Z">
    <w:tcPr>
      <w:tcW w:w="3192" w:type="dxa"/>
    </w:tcPr>
  </w:tcPrChange>
</w:tcPr>
```

The tcPrChange element specifies that there was a revision to the table cell properties at `2006-06-01` by `Tristan Davis`, and the previous set of properties on the table cell was the set specifies in the child tcPr element. *end example*]

Parent Elements
tcPr (§2.7.5.8); tcPr (§2.7.5.9); tcPr (§2.4.67)

Child Elements	Subclause
tcPr (Previous Table Cell Properties)	§2.4.66

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`…`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example`

Attributes	Description
	Author, which may be used as desired. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`…`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TcPrChange">
    <complexContent>
        <extension base="CT_TrackChange">
            <sequence>
                <element name="tcPr" type="CT_TcPrInner" minOccurs="1"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

2.13.5.39 trPrChange (Revision Information for Table Row Properties)

This element specifies the details about a single revision to a set of table row properties in a WordprocessingML document.

This element stores this revision as follows:

- The child element of this element contains the complete set of table row properties which were applied to the parent table row before this revision
- The attributes of this element contain information about when this revision took place (i.e. when these properties became a 'former' set of table row properties).

[*Example*: Consider a table cell in a WordprocessingML document which has a change in the table row properties that is tracked as a revision. This revision would be specified using the following WordprocessingML markup:

```
<w:trPr>
  <w:cantSplit/>
  <w:trPrChange w:id="8" w:author="Tristan Davis" w:date="2006-06-01T13:39:00Z">
    <w:trPr/>
  </w:trPrChange>
</w:trPr>
```

The trPrChange element specifies that there was a revision to the table row properties at `2006-06-01` by `Tristan Davis`, and the previous set of properties on the table row was the set specified in the child trPr element (in this case, the null set). *end example*]

Parent Elements
trPr (§2.7.5.10); trPr (§2.7.5.11); trPr (§2.4.78)

Child Elements	Subclause
trPr (Previous Table Row Properties)	§2.4.79

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:author="Example Author">`
	`  …`
	`</w:…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</w:…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§2.18.15).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`  …`
	`</w:…>`

Attributes	Description
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TrPrChange">
   <complexContent>
      <extension base="CT_TrackChange">
         <sequence>
            <element name="trPr" type="CT_TrPrBase" minOccurs="1"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

2.13.6 Bookmarks

Within a WordprocessingML document, *bookmarks* refer to arbitrary regions of content which are bounded and have a unique name associated with them.

Because bookmarks are a legacy word processing function which predates the concepts of XML and well-formedness, they can start and end at any location within a document's contents and therefore must use the "cross-structure" annotation format described in §2.13.2.

[*Example*: Consider the following WordprocessingML markup for two paragraphs, each reading `Example Text`, where a bookmark has been added spanning the second word in paragraph one and the first word in paragraph two:

```
<w:p>
  <w:r>
    <w:t>Example</w:t>
  </w:r>
  <w:bookmarkStart w:id="0" w:name="sampleBookmark" />
  <w:r>
    <w:t xml:space="preserve"> text.</w:t>
  </w:r>
</w:p>
<w:p>
  <w:r>
    <w:t>Example</w:t>
  </w:r>
  <w:bookmarkEnd w:id="0" />
  <w:r>
    <w:t xml:space="preserve"> text.</w:t>
  </w:r>
</w:p>
```

The bookmarkStart and bookmarkEnd elements (§2.13.6.2; §2.13.6.1) specify the location where the bookmark starts and ends, but cannot contain it using a single tag because it spans part of two paragraphs. However, the two tags are part of one group because the id attribute value specifies `0` for both. *end example*]

2.13.6.1 bookmarkEnd (Bookmark End)

This element specifies the end of a bookmark within a WordprocessingML document. This end marker is matched with the appropriately paired start marker by matching the value of the id attribute from the associated bookmarkStart element.

If no bookmarkStart element exists prior to this element in document order with a matching id attribute value, then this element is ignored and no bookmark is present in the document with this name.

[*Example*: Consider a document with a bookmark which spans half of paragraph one, and part of paragraph two. The following WordprocessingML illustrates an example of content which fufills this constraint:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">This is sentence one.</w:t>
  </w:r>
  <w:bookmarkStart w:id="0" w:name="testing123"/>
  <w:r>
    <w:t>This is sentence two.</w:t>
  </w:r>
</w:p>
```

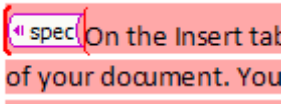
```
<w:p>
  <w:r>
    <w:t xml:space="preserve">This </w:t>
  </w:r>
  <w:bookmarkEnd w:id="0"/>
  <w:r>
    <w:t>is sentence three.</w:t>
  </w:r>
</w:p>
```

The bookmarkEnd element specifies the end of the region for the bookmark whose bookmarkStart element has an id attribute value of `0`. In this case, this refers to the `testing123` bookmark. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32); endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Attributes	Description
displacedByCustomXml (Annotation Marker Relocated For Custom XML Markup)	Specifies that the parent annotation's placement shall be directly linked with the location of the physical presentation of a custom XML element in the document. This element only has an effect when the custom XML element is block-level (i.e. surrounds an entire paragraph), as in this scenario the logical and physical placement of the annotation and custom XML element may differ. Specifically, in this case, the custom XML is presented *around* the block-level object it encloses (the paragraph, table, table row, or table cell), but is physically represented within that same object (i.e. within the paragraph, table, table row or table cell). This requirement stems from the fact that there is no location for the location of the annotation within the document at its logical location (around a table, for example). If this element is omitted, then the annotation shall be anchored inside of all block-level custom XML elements in the paragraph. If this element is present, but no block-level custom XML tag is located at the position it specifies (before or after), then it shall be ignored. [*Example*: Consider a paragraph with block level custom XML markup and two comment anchor annotations (one before and one after the custom XML element's physical representation), as follows:

Attributes	Description

	
	
	Since all three of these items are around the entire paragraph, they are stored outside of the paragraph. However, in order to ensure that their relative positions are stored correctly, any annotation which shall be displaced by the physical custom XML element specifies this information, resulting in the following WordprocessingML:
	
	`<w:commentRangeStart w:id="0" />`
	`<w:commentRangeStart w:id="1" w:displaced byCustomXml="next" />`
	`<w:customXml w:element="spec" … />`
	`<w:p>`
	`…`
	`</w:p>`
	
	The displacedByCustomXml attribute specifies that even though all three of these items are around the paragraph and will be moved inside the paragraph to be represented physically, the comment with ID 0 shall be inside the custom XML, but the comment with ID 1 shall be displaced to stay outside of the relative location of the next custom XML element (the spec element). *end example*]
	
	The possible values for this attribute are defined by the ST_DisplacedByCustomXml simple type (§2.18.17).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MarkupRange">
   <complexContent>
      <extension base="CT_Markup">
         <attribute name="displacedByCustomXml" type="ST_DisplacedByCustomXml" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

2.13.6.2 bookmarkStart (Bookmark Start)

This element specifies the start of a bookmark within a WordprocessingML document. This start marker is matched with the appropriately paired end marker by matching the value of the id attribute from the associated bookmarkEnd element.

If no bookmarkEnd element exists subsequent to this element in document order with a matching id attribute value, then this element is ignored and no bookmark is present in the document with this name.

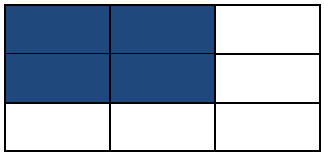
If a bookmark begins and ends within a single table, it is possible for that bookmark to cover discontiguous parts of that table which are logically related (e.g. a single column in a table). This type of placement for a bookmark is accomplished (and described in detail) on the colFirst and colLast attributes on this element.

[*Example*: Consider a document with a bookmark which spans half of paragraph one, and part of paragraph two. The following WordprocessingML illustrates an example of content which fufills this constraint:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">This is sentence one.</w:t>
  </w:r>
  <w:bookmarkStart w:id="0" w:name="testing123"/>
  <w:r>
    <w:t>This is sentence two.</w:t>
  </w:r>
</w:p>
<w:p>
  <w:r>
    <w:t xml:space="preserve">This </w:t>
  </w:r>
  <w:bookmarkEnd w:id="0"/>
  <w:r>
    <w:t>is sentence three.</w:t>
  </w:r>
</w:p>
```

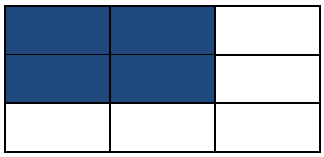
The bookmarkStart element specifies the start of the region for the `testing123` bookmark. This element is then linked to the bookmarkEnd element which also has an id attribute value of `0`. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32); endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Attributes	Description
colFirst (First Table Column Covered By Bookmark)	Specifies the zero-based index of the first column in this row which shall be part of this bookmark. When a bookmark is contained within a table, it is possible for that bookmark to only cover cells within a certain column and row range within that table, by specifying: • The first row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkStart element in the first table cell in that row. • The first column included in the bookmark for each of the specified row(s) via this attribute. • The last column included in the bookmark for each of the specified row(s) via the colLast attribute. • The last row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkEnd element at the end of that table row. If this attribute appears, then the colLast attribute must also appear (regardless of where this bookmark is located) or the document shall be considered non-conformant. If this attibute and its pair occur on a bookmark which is not contained in a table, then their values should be ignored. If this value exceeds the value of colLast or the number of columns in the table, then both values should be ignored. [*Example*: Consider a three row by three column table where a table bookmark shall be applied to the contents of the first two cells in the first two rows in the table (the cells shaded below):  This bookmark would be specified using the following WordprocessingML for the table's conents: `<w:tbl>`

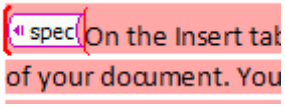
Attributes	Description
	… <w:tr> <w:tc> <w:bookMarkStart w:colFirst="0" w:colLast="1" w:id="0" w:name="table"/> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> </w:tr> <w:tr> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> <w:bookmarkEnd w:id="0" /> </w:tr> <w:tr> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> </w:tr> </w:tbl> The colFirst attribute specifies that all columns starting with the first column shall be included in the table bookmark. This will apply starting with the first row and ending with the second row (the two rows within the bookmark's start and end). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
colLast (Last Table Column Covered By	Specifies the zero-based index of the last column in this row which shall be part of this bookmark.

Attributes	Description

Bookmark)	When a bookmark is contained within a table, it is possible for that bookmark to only cover cells within a certain column and row range within that table, by specifying:
	• The first row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkStart element in the first table cell in that row.
	• The first column included in the bookmark for each of the specified row(s) via the colFirst attribute.
	• The last column included in the bookmark for each of the specified row(s) via this attribute.
	• The last row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkEnd element at the end of that table row.
	
	If this attribute appears, then the colFirst attribute must also appear (regardless of where this bookmark is located) or the document shall be considered non-conformant. If this attibute and its pair occur on a bookmark which is not contained in a table, then their values should be ignored. If this value does not equal or exceed the value of colFirst or the number of columns in the table, then both values should be ignored.
	
	[*Example*: Consider a three row by three column table where a table bookmark shall be applied to the contents of the first two cells in the first two rows in the table (the cells shaded below):
	
	 
	
	This bookmark would be specified using the following WordprocessingML for the table's conents:
	
	`<w:tbl>`
	`…`
	`<w:tr>`
	`<w:tc>`
	`<w:bookMarkStart w:colFirst="0" w:colLast="1" w:id="0" w:name="table"/>`
	`<w:p/>`
	`</w:tc>`
	`<w:tc>`
	`<w:p/>`
	`</w:tc>`
	`<w:tc>`
	`<w:p/>`
	`</w:tc>`

Attributes	Description
	</w:tr> <w:tr> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> <w:bookmarkEnd w:id="0" /> </w:tr> <w:tr> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> </w:tr> </w:tbl> The colLast attribute specifies that the last column that shall be included in the table bookmark is the second column. This will apply starting with the first row and ending with the second row (the two rows within the bookmark's start and end). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
displacedByCustomXml (Annotation Marker Relocated For Custom XML Markup)	Specifies that the parent annotation's placement shall be directly linked with the location of the physical presentation of a custom XML element in the document. This element only has an effect when the custom XML element is block-level (i.e. surrounds an entire paragraph), as in this scenario the logical and physical placement of the annotation and custom XML element may differ. Specifically, in this case, the custom XML is presented *around* the block-level object it encloses (the paragraph, table, table row, or table cell), but is physically represented within that same object (i.e. within the paragraph, table, table row or table cell). This requirement stems from the fact that there is no location for the location of the annotation within the document at its logical location (around a table, for example). If this element is omitted, then the annotation shall be anchored inside of all block-level custom XML elements in the paragraph. If this element is present, but no block-level custom XML tag is located at the position it specifies (before or after), then it shall be

Attributes	Description

	ignored.
	
	[*Example*: Consider a paragraph with block level custom XML markup and two comment anchor annotations (one before and one after the custom XML element's physical representation), as follows:
	
	 
	
	Since all three of these items are around the entire paragraph, they are stored outside of the paragraph. However, in order to ensure that their relative positions are stored correctly, any annotation which shall be displaced by the physical custom XML element specifies this information, resulting in the following WordprocessingML:
	
	`<w:commentRangeStart w:id="0" />`
	`<w:commentRangeStart w:id="1" w:displaced byCustomXml="next" />`
	`<w:customXml w:element="spec" … />`
	`<w:p>`
	`…`
	`</w:p>`
	
	The displacedByCustomXml attribute specifies that even though all three of these items are around the paragraph and will be moved inside the paragraph to be represented physically, the comment with ID 0 shall be inside the custom XML, but the comment with ID 1 shall be displaced to stay outside of the relative location of the next custom XML element (the spec element). *end example*]
	
	The possible values for this attribute are defined by the ST_DisplacedByCustomXml simple type (§2.18.17).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type

Attributes	Description
	(§2.18.16).

name (Bookmark Name)	Specifies the bookmark name.
	
	If multiple bookmarks in a document share the same name, then the first bookmark (defined by the location of the bookmarkStart element in document order) shall be maintained, and all subsequent bookmarks should be ignored.
	
	[*Example*: Consider the following XML for a bookmark around a single word:
	
	`<w:p>`
	`  <w:bookmarkStart w:id="0" w:name="place" />`
	`  <w:r>`
	`  <w:t>Seattle</w:t>`
	`  </w:r>`
	`  <w:bookmarkEnd w:id="0" />`
	`</w:p>`
	
	The name attribute specifies that the name for this bookmark is `place`. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Bookmark">
    <complexContent>
        <extension base="CT_BookmarkRange">
            <attribute name="name" type="ST_String" use="required"/>
        </extension>
    </complexContent>
</complexType>
```

2.13.7 Range Permissions

Range permissions in a WordprocessingML document refer to a special type of bookmark used to control which subset(s) of users may edit a particular region of a document. Range permissions specify the user or set of users which are allowed to edit all content between them whenever the document protection specified by the documentProtection element (§2.15.1.28) is enabled and set to `readOnly` or `comments`.

Like bookmarks, range permissions are a legacy word processing function which predates the concepts of XML and well-formedness, so they can start and end at any location within a document's contents and therefore must use the "cross-structure" annotation format described in §2.13.2.

[*Example*: Consider the following WordprocessingML markup for a single paragraph, where a range permission has been added spanning the words `range permission`:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">This is a </w:t>
  </w:r>
  <w:permStart w:id="0" w:edGrp="everyone"/>
  <w:r>
    <w:t>range permission</w:t>
  </w:r>
  <w:permEnd w:id="0"/>
  <w:r>
    <w:t>.</w:t>
  </w:r>
</w:p>
```

The permStart and permEnd elements (§2.13.7.1; §2.13.7.2) specify the location where the range permission starts and ends. The two tags are part of one group because the id attribute value specifies `0` for both.

If document protection was enabled, then no content in this document shall be editable except for this range permission, which is editable by all users that open the document (specified using an editor group of everyone). *end example*]

2.13.7.1 permEnd (Range Permission End)

This element specifies the end of a single range permission within a WordprocessingML document. This end marker is matched with the appropriately paired start marker by matching the value of the id attribute from the associated permStart element.

If no permStart element exists prior to this element in document order with a matching id attribute value, then this element is ignored and no range permission is present in the document.

[*Example*: Consider a document with a range permission which spans half of paragraph one, and part of paragraph two. The following WordprocessingML illustrates an example of content which fufills this constraint:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">This is sentence one.</w:t>
  </w:r>
  <w:permStart w:id="0" w:edGrp="everyone"/>
  <w:r>
    <w:t>This is sentence two.</w:t>
  </w:r>
</w:p>
```

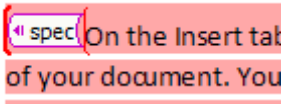
```
<w:p>
  <w:r>
    <w:t xml:space="preserve">This </w:t>
  </w:r>
  <w:permEnd w:id="0"/>
  <w:r>
    <w:t>is sentence three.</w:t>
  </w:r>
</w:p>
```

The permEnd element specifies the end of the region for the range permission whose permStart element has an id attribute value of `0`. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32); endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Attributes	Description
displacedByCustomXml (Annotation Displaced By Custom XML Markup)	Specifies that the parent annotation's placement shall be directly linked with the location of the physical presentation of a custom XML element in the document. This element only has an effect when the custom XML element is block-level (i.e. surrounds an entire paragraph), as in this scenario the logical and physical placement of the annotation and custom XML element may differ. Specifically, in this case, the custom XML is presented *around* the block-level object it encloses (the paragraph, table, table row, or table cell), but is physically represented within that same object (i.e. within the paragraph, table, table row or table cell). This requirement stems from the fact that there is no location for the location of the annotation within the document at its logical location (around a table, for example). If this element is omitted, then the annotation shall be anchored inside of all block-level custom XML elements in the paragraph. If this element is present, but no block-level custom XML tag is located at the position it specifies (before or after), then it shall be ignored. [*Example*: Consider a paragraph with block level custom XML markup and two comment anchor annotations (one before and one after the custom XML element's physical representation), as follows:

Attributes	Description

	
	
	Since all three of these items are around the entire paragraph, they are stored outside of the paragraph. However, in order to ensure that their relative positions are stored correctly, any annotation which shall be displaced by the physical custom XML element specifies this information, resulting in the following WordprocessingML:
	
	`<w:commentRangeStart w:id="0" />`
	`<w:commentRangeStart w:id="1" w:displaced byCustomXml="next" />`
	`<w:customXml w:element="spec" … />`
	`<w:p>`
	`…`
	`</w:p>`
	
	The displacedByCustomXml attribute specifies that even though all three of these items are around the paragraph and will be moved inside the paragraph to be represented physically, the comment with ID 0 shall be inside the custom XML, but the comment with ID 1 shall be displaced to stay outside of the relative location of the next custom XML element (the spec element). *end example*]
	
	The possible values for this attribute are defined by the ST_DisplacedByCustomXml simple type (§2.18.17).

id (Annotation ID)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Perm">
   <attribute name="id" type="ST_String" use="required"/>
   <attribute name="displacedByCustomXml" type="ST_DisplacedByCustomXml" use="optional"/>
</complexType>
```

2.13.7.2 permStart (Range Permission Start)

This element specifies the start of a range permission within a WordprocessingML document. This start marker is matched with the appropriately paired end marker by matching the value of the id attribute from the associated permEnd element.

If no permEnd element exists subsequent to this element in document order with a matching id attribute value, then this element is ignored and no range permission is present in the document.

If a range permission begins and ends within a single table, it is possible for that permission to cover discontiguous parts of that table which are logically related (e.g. a single column in a table). This type of placement for a range permission is accomplished (and described in detail) on the colFirst and colLast attributes on this element.

[*Example*: Consider a document with a range permission which spans half of paragraph one, and part of paragraph two. The following WordprocessingML illustrates an example of content which fufills this constraint:

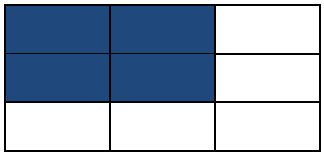
```
<w:p>
  <w:r>
    <w:t xml:space="preserve">This is sentence one.</w:t>
  </w:r>
  <w:permStart w:id="0" w:edGrp="everyone"/>
  <w:r>
    <w:t>This is sentence two.</w:t>
  </w:r>
</w:p>
<w:p>
  <w:r>
    <w:t xml:space="preserve">This </w:t>
  </w:r>
  <w:permEnd w:id="0"/>
  <w:r>
    <w:t>is sentence three.</w:t>
  </w:r>
</w:p>
```

The permStart element specifies the start of the region for the range permission. This element is then linked to the permEnd element which also has an id attribute value of 0. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32); endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32);

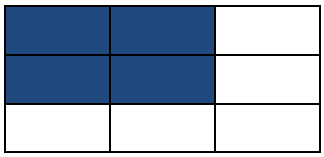
Parent Elements
sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Attributes	Description

colFirst (First Table Column Covered By Range Permission)	Specifies the zero-based index of the first column in this row which shall be part of this range permission.
	
	When a range permission is contained within a table, it is possible for that range permission to only cover cells within a certain column and row range within that table, by specifying:
	• The first row for which the specified columns are part of the table range permission. This is accomplished by placing the permStart element in the first table cell in that row.
	• The first column included in the range permission for each of the specified row(s) via this attribute.
	• The last column included in the range permission for each of the specified row(s) via the colLast attribute.
	• The last row for which the specified columns are part of the table range permission. This is accomplished by placing the permEnd element at the end of that table row.
	
	If this attribute appears, then the colLast attribute must also appear (regardless of where this bookmark is located) or the document shall be considered non-conformant. If this attibute and its pair occur on a range permission which is not contained in a table, then their values should be ignored. If this value exceeds the value of colLast or the number of columns in the table, then both values should be ignored.
	
	[*Example*: Consider a three row by three column table where a table range permission shall be applied to the contents of the first two cells in the first two rows in the table (the cells shaded below):
	
	 
	
	This bookmark would be specified using the following WordprocessingML for the table's conents:
	
	`<w:tbl>`
	`…`
	`<w:tr>`
	`<w:tc>`
	`<w:permStart w:colFirst="0" w:colLast="1" w:id="0"`

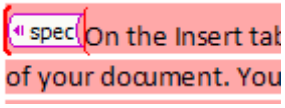
Attributes	Description
	w:edGrp="everyone"/> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> </w:tr> <w:tr> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> <w:permEnd w:id="0" /> </w:tr> <w:tr> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> </w:tr> </w:tbl> The colFirst attribute specifies that all columns starting with the first column shall be included in the table range permission. This will apply starting with the first row and ending with the second row (the two rows within the range permission's start and end). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
colLast (Last Table Column Covered By Range Permission)	Specifies the zero-based index of the last column in this row which shall be part of this range permission. When a range permission is contained within a table, it is possible for that range permission to only cover cells within a certain column and row range within that table, by

Attributes	Description

	specifying:
	• The first row for which the specified columns are part of the table range permission. This is accomplished by placing the permStart element in the first table cell in that row.
	• The first column included in the range permission for each of the specified row(s) via the colFirst attribute.
	• The last column included in the range permission for each of the specified row(s) via this attribute.
	• The last row for which the specified columns are part of the table range permission. This is accomplished by placing the permEnd element at the end of that table row.
	
	If this attribute appears, then the colFirst attribute must also appear (regardless of where this bookmark is located) or the document shall be considered non-conformant. If this attibute and its pair occur on a bookmark which is not contained in a table, then their values should be ignored. If this value does not equal or exceed the value of colFirst or the number of columns in the table, then both values should be ignored.
	
	[*Example*: Consider a three row by three column table where a table range permission shall be applied to the contents of the first two cells in the first two rows in the table (the cells shaded below):
	
	 
	
	This bookmark would be specified using the following WordprocessingML for the table's conents:
	
	`<w:tbl>`
	`…`
	`<w:tr>`
	`<w:tc>`
	`<w:permStart w:colFirst="0" w:colLast="1" w:id="0" w:edGrp="everyone"/>`
	`<w:p/>`
	`</w:tc>`
	`<w:tc>`
	`<w:p/>`
	`</w:tc>`
	`<w:tc>`
	`<w:p/>`
	`</w:tc>`
	`</w:tr>`
	`<w:tr>`

Attributes	Description
	<w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> <w:permEnd w:id="0" /> </w:tr> <w:tr> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> <w:tc> <w:p/> </w:tc> </w:tr> </w:tbl> The colLast attribute specifies that the last column that shall be included in the table range permission is the second column. This will apply starting with the first row and ending with the second row (the two rows within the range permission's start and end). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
displacedByCustomXml (Annotation Displaced By Custom XML Markup)	Specifies that the parent annotation's placement shall be directly linked with the location of the physical presentation of a custom XML element in the document. This element only has an effect when the custom XML element is block-level (i.e. surrounds an entire paragraph), as in this scenario the logical and physical placement of the annotation and custom XML element may differ. Specifically, in this case, the custom XML is presented *around* the block-level object it encloses (the paragraph, table, table row, or table cell), but is physically represented within that same object (i.e. within the paragraph, table, table row or table cell). This requirement stems from the fact that there is no location for the location of the annotation within the document at its logical location (around a table, for example). If this element is omitted, then the annotation shall be anchored inside of all block-level custom XML elements in the paragraph. If this element is present, but no block-level custom XML tag is located at the position it specifies (before or after), then it shall be ignored.

Attributes	Description

	[*Example*: Consider a paragraph with block level custom XML markup and two comment anchor annotations (one before and one after the custom XML element's physical representation), as follows:
	
	 
	
	Since all three of these items are around the entire paragraph, they are stored outside of the paragraph. However, in order to ensure that their relative positions are stored correctly, any annotation which shall be displaced by the physical custom XML element specifies this information, resulting in the following WordprocessingML:
	
	`<w:commentRangeStart w:id="0" />`
	`<w:commentRangeStart w:id="1" w:displaced byCustomXml="next" />`
	`<w:customXml w:element="spec" … />`
	`<w:p>`
	`…`
	`</w:p>`
	
	The displacedByCustomXml attribute specifies that even though all three of these items are around the paragraph and will be moved inside the paragraph to be represented physically, the comment with ID 0 shall be inside the custom XML, but the comment with ID 1 shall be displaced to stay outside of the relative location of the next custom XML element (the spec element). *end example*]
	
	The possible values for this attribute are defined by the ST_DisplacedByCustomXml simple type (§2.18.17).

ed (Single User For Range Permission)	Specifies a single user for which this range permission shall be enabled (i.e. a user which shall be able to edit this range when document protection is enabled).
	
	This editor can be stored in one of the following forms:
	• `DOMAIN\username` - for users whose access shall be authenticated using the current user's domain credentials
	• `user@domain.com` - for users whose access shall be authenticated using the user's e-mail address as credentials
	• `user` - for users whose access shall be authenticated using the current user's machine credentials
	
	[*Example*: Consider a range permission defined as follows:
	
	`<w:permStart w:id="0" w:ed="example@contoso.com" … />`
	`…`
	`<w:permEnd w:id="0" />`

Attributes	Description
	The ed attribute value of `example@contoso.com` specifies that only user(s) who can authenticate with an application as associated with that e-mail address shall be allowed to edit the contents between the start and end markers when document protection is being enforced. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

edGrp (Editor Group For Range Permission)	Specifies an alias (or editing group) which shall be used to determine if the current user shall be allowed to edit this range of the document. This mechanism simply provides a set of predefined editing groups which may be associated with user accounts by applications in any desired manner.
	
	[*Example*: Consider a range permission defined as follows:
	
	`<w:permStart w:id="0" w:edGrp="editors" … />`
	`…`
	`<w:permEnd w:id="0" />`
	
	The edGrp attribute value of `editors` specifies that only user(s) who the current application associates with the editors group shall be allowed to edit the contents between the start and end markers when document protection is being enforced. *end example*]
	
	The possible values for this attribute are defined by the ST_EdGrp simple type (§2.18.25).

id (Annotation ID)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<w:… w:id="1" … >`
	`…`
	`</w:…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PermStart">
   <complexContent>
      <extension base="CT_Perm">
         <attribute name="edGrp" type="ST_EdGrp" use="optional"/>
         <attribute name="ed" type="ST_String" use="optional"/>
         <attribute name="colFirst" type="ST_DecimalNumber" use="optional"/>
         <attribute name="colLast" type="ST_DecimalNumber" use="optional"/>
      </extension>
   </complexContent>
</complexType>
```

2.13.8 Spelling & Grammar

The final type of annotation stored in a WordprocessingML document, *spelling and grammar errors* are annotations used to specify the locations of existing spelling and grammatical errors within the contents of a document.

[*Rationale*: When a WordprocessingML document is saved, applications may choose to save currently flagged spelling and grammar errors, for two reasons:

- In order to increase the performance subsequent loads of the document (as those load operations can rely on the persisted proofing state of the document)
- In order to store words which shall not be marked as proofing errors regardless of how they would normally be flagged by the proofing tools engine (i.e. to store spelling and grammar exceptions).

end rationale]

[*Example*: Consider the following paragraph consisting of two misspelled words, where the second word has been explicitly flagged as not being a spelling error. This paragraph would consist of the following WordprocessingML markup:

```
<w:p>
  <w:proofErr w:val="spellStart"/>
  <w:r>
    <w:t>erqwt</w:t>
  </w:r>
  <w:proofErr w:val="spellEnd"/>
  <w:r>
    <w:t xml:space="preserve"> werewr</w:t>
  </w:r>
</w:p>
```

The proofErr elements, with a val attribute value of `spellStart` and `spellEnd` respectively, delimit the start and end the content in this paragraph which is stored as a spelling error. Since the second word is not included in that range, it is not stored as a spelling error. *end example*]

2.13.8.1 proofErr (Proofing Error Anchor)

This element specifies the presence of a start or end anchor for a single proofing error within a WordprocessingML document.

When proofing errors are stored in a document, their semantics shall be interpreted as follows:

- Each proofing error with a type attribute value of `spellStart` shall be linked with the next error with a type attribute of `spellEnd`. If one does not exist, then this error should be ignored.
- Each proofing error with a type attribute value of `spellEnd` which was not preceded by an error with a type attribute value of `spellStart` (that was not previously matched to an end) should be ignored.
- Each proofing error with a type attribute value of `gramStart` shall be linked with the next error with a type attribute of `gramEnd`. If one does not exist, then this error should be ignored.
- Each proofing error with a type attribute value of `gramEnd` which was not preceded by an error with a type attribute value of `gramStart` (that was not previously matched to an end) should be ignored.

[*Example*: Consider the following sentence with a grammatical error in its subject/verb agreement. If an application recognized this error and wished to persist it to the document, this paragraph would consist of the following WordprocessingML markup:

```
<w:p>
  <w:proofErr w:val="gramStart"/>
  <w:r>
    <w:t>This are</w:t>
  </w:r>
  <w:proofErr w:val="gramEnd"/>
  <w:r>
    <w:t xml:space="preserve"> an error.</w:t>
  </w:r>
</w:p>
```

The proofErr elements, with a val attribute value of `gramStart` and `gramEnd` respectively, delimit the start and end the content in this paragraph which is stored as a grammatical error. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32); endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Attributes	Description

Attributes	Description

type (Proofing Error Anchor Type)	Specifies the type of proofing error anchor at this location in the document. This type implies the necessary semantics for this element as defined by the parent element.
	
	[*Example*: Consider the following sentence with a proofing error, consisting of the following WordprocessingML markup:
	
	`<w:r>`
	`  <w:t>are</w:t>`
	`</w:r>`
	`<w:proofErr w:val="gramEnd"/>`
	`<w:r>`
	`  <w:t xml:space="preserve"> an error.</w:t>`
	`</w:r>`
	
	The val attribute value of gramEnd specifies that the proofing error is the location of the end of content which is stored as a grammatical error. *end example*]
	
	The possible values for this attribute are defined by the ST_ProofErr simple type (§2.18.77).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ProofErr">
   <attribute name="type" type="ST_ProofErr" use="required"/>
</complexType>
```

2.14 Mail Merge

Mail merge refers to an operation by which WordprocessingML documents may work in conjunction with data from an external data source, importing this data into a document according to a set of codes contained in WordprocessingML known as fields.

A WordprocessingML document that contains the mailMerge element (§2.14.20) and is therefore connected to an external data source, is known as a *source document*. In addition to being connected to an external data source and containing fields, a source document may contain any regular WordprocessingML constructs such as:

- Text runs
- Paragraphs
- Images
- Tables
- Numbering
- Etc.

There are two key parts of the mail merge data stored in a WordprocessingML document

11. Information connecting a document to an external data source
12. Information populating fields within that document with external data.

Once the fields in a merged document have been populated with external data, mail merge has been completed and the resulting files are known as *mail merged documents* or simply *merged documents*.

The mail merge settings for a WordprocessingML document are stored in two locations:

- The standard mail merge settings are stored as the child elements of the mailMerge element (§2.14.20)
- A set of additional mail merge settings stored in the odso element (§2.14.25), and collectively referred to as the Office Data Source Object settings. The *Office Data Source Object* is an extension to the standard settings stored with a mail merge which performs two functions: First, it provides additional information about the mail merge data source, specifically: information about how to map the columns in the data source to `MERGEFIELD` fields and information about records which shall be included and excluded when creating merged documents. Second, it provides an alternate set of connection information which should be used when the dataType element (§2.14.10) specifies a value of `native`. This alternate connection string provides additional connection information for applications which choose to support the ODSO connection string syntax.

[*Example*: Consider a WordprocessingML document containing static WordprocessingML constructs such as text runs and paragraphs in addition to two WordprocessingML `MERGEFIELD` fields (§2.16.5.42) calling for `Courtesy Title` and `Last Name` data. The field codes for each field are displayed, delimited by {} characters:

Dear {MERGEFIELD "Courtesy Title" \m} {MERGEFIELD "Last Name" \m}, Sample text. Sincerely,

If the following WordprocessingML was added to this document, this document would become a *source document* rather than just a standard WordprocessingML document, as the mailMerge (§2.14.20) element specifies the elements and attributes necessary to enabled the document to connect to an external spreadsheet data source.

```
<w:mailMerge>
  …
  <w:dataType w:val="spreadsheet" />
  <w:query w:val="SELECT * FROM `Sheet1$`" />
  <w:dataSource r:id="rId1" />
  …
</w:mailMerge>
```

Here, the dataType (§2.14.10) and dataSource (§2.14.9) elements specify that the given document shall be connected to the external data source target by the relationship whose relationship value is `rId1` as specified in the dataSource element (§2.14.9). While connected to the external data source, the *source document* together with the hosting application and/or data source access application will extract data from the external data source to perform the merge as specified by the connectString (§2.14.8) and query (§2.14.26) elements. *end example*]

2.14.1 active (Record Is Included in Mail Merge)

This element specifies whether a specific record from the specified external data source shall be imported into a merged WordprocessingML document when the mail merge defined for a source document is performed. If this element's val attribute is `false`, then the record specified by the parent element shall not used to create a merged document.

If this element is omitted for a given record, the data record associated with it shall be imported into a merged WordprocessingML document when the mail merge is performed.

[*Example*: Consider the following fragment from a source WordprocessingML document that is connected to an external data source containing two records, one of which will not be imported into a merged WordprocessingML document when the conforming hosting application performs the data import.

```
<w:recipients>
  <w:recipientData>
    <w:active w:val="false" />
    <w:uniqueTag w:val="1126664175" />
  </w:recipientData>
  <w:recipientData>
    <w:uniqueTag w:val="1530576378" />
  </w:recipientData>
</w:recipients>
```

In this XML fragment, the external data record who is identified by the uniqueTag element (§2.14.35) with a val attribute equal to `1126664175` will not be imported into a merged document as the active element associated with it has a val attribute equal to `false`. Conversely, the external data record associated with the uniqueTag element with a val attribute equal to `1530576378` will be imported into a merged document, as its active element has been omitted (implying the default value of `true`). *end example*]

Parent Elements
recipientData (§2.14.28)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.14.2 activeRecord (Record Currently Displayed In Merged Document)

This element specifies that the hosting application shall display the given record from the specified external data source in place of the MERGEFIELD fields (§2.16.5.42) its data is mapped to via the fieldMapData element (§2.14.15) in a merged document. When this element is present, the val attribute shall specify the one-based index of the record from that data source which shall be used to populate this document.

If the activeRecord element is omitted with the viewMergedData element's val attribute equal to true, the hosting application shall behave as if the activeRecord element's val attribute was equal to 1. If the viewMergedData element (§2.14.36) is omitted or present with a val attribute equal to Off, 0, or false, then this element shall be ignored. If the activeRecord record is given a val attribute that is less than one or greater than the number of records in the specified external data source, the hosting application shall treat this val attribute as if it were equal to 1.

[*Example*: Consider a merged WordprocessingML document containing two WordprocessingML fields calling for Courtesy Title and Last Name data and a sample text paragraph. Also, note that the external data source this merged document is connected to contains two records, both containing name and address information, with the first record pertaining to Mr. Doe, and the second pertaining to Ms. Smith.

This table below illustrates the necessary WordprocessingML to display applicable data from the specified external data source within the merged document where fields have been inserted:

`<w:viewMergedData val="off"/>`	`<w:viewMergedData val="on" />` `<w:activeRecord w:val="1" />`	`<w:viewMergedData val="on" />` `<w:activeRecord w:val="2" />`
Dear {MERGEFIELD "Courtesy Title" \m} {MERGEFIELD "Last Name" \m}, Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sincerely,	Dear Mr. Doe: Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sincerely,	Dear Ms. Smith: Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sincerely,

end example]

[*Example*: Consider the following WordprocessingML from a merged WordprocessingML document:

```
<w:viewMergedData />
<w:activeRecord w:val="2" />
```

The activeRecord element is present and has a val attribute equal to 2, therefore this WordprocessingML specifies that a conforming hosting application shall display data from the second record of the specified external data source in place of fields its data is mapped to within the merged document. *end example*]

Parent Elements
mailMerge (§2.14.20)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.14.3 addressFieldName (Column Containing E-mail Address)

This element specifies the column within a given external data source that contains e-mail addresses. This element is specified independently of the field mappings specified for a given merged document via the fieldMapData element (§2.14.15).

If this element is omitted, or no column exists in the data source with this column name, then the source document specifies that no e-mail address data shall be associated with this mail merge.

[*Note*: This element is generally used to allow the e-mailing of merged documents resulting from populating the fields within a merged document with external data.

This element is independent of the field mapping specified for a given merged document via the fieldMapData element (§2.14.15). This separation enables applications to email the documents resulting from the population of WordprocessingML fields with external data regardless of the presence or absence of a field mapped to external data specifying email addresses. *end note*]

[*Example*: Consider a merged WordprocessingML document that is connected to an external data source containing a column of data tilted `Alternate Email Addresses`. The following WordprocessingML would be included in the source and merged documents to specify which column in the external data source contains email addresses.

```
<w:addressFieldName w:val="Alternate Email Address" />
```

The addressFieldName element specifies that the `Alternate Email Address` column contains e-mail addresses for each record. *end example*]

Parent Elements
mailMerge (§2.14.20)

Attributes	Description
val (String Value)	Specifies that its contents will contain a string. The contents of this string are interpreted based on the context of the parent XML

Attributes	Description

	element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.14.4 checkErrors (Mail Merge Error Reporting Setting)

This element specifies the type of error reporting which shall be conducted by an application when performing a mail merge against the specified source data.

The type of error reporting implied by this element shall be defined as follows:

- Simulate the population of fields with mapped external data and report errors in a new document if the val attribute is equal to 1.
- While populating fields with mapped external data, pausing to report each error as it occurs if the val attribute is equal to 2.
- Populate fields with mapped external data and report errors in a new document if the val attribute is equal to 3.
- Application-defined behaviors may be used if the val attribute is equal to any other value.

If this element is omitted, or its value is set to a value outside of those specified below that is not understood by the hosting application, then its value shall be assumed to be 2.

[*Example*: Consider a mail merge whose WordprocessingML definition includes the following:

```
<w:checkErrors w:val="3" />
```

The presence of a checkErrors element with a val attribute of 3 indicates that the hosting application shall conduct the type of error reporting specified above, performing the mail merge, populating fields with mapped external data and reporting errors in a new document . *end example*]

Parent Elements
mailMerge (§2.14.20)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.14.5 colDelim (Column Delimiter for Data Source)

This element specifies the character which shall be interpreted as the column delimiter used to separate columns within external data sources. The character representing the specific delimiter used for the external data source referenced by a source or merged WordprocessingML document is specified via a decimal number representing the decimal number for the Unicode character representation within this element's val attribute.

If this element is omitted, then no column delimiter shall be specified for the data source in this mail merge.

[*Example*: Consider the following WordprocessingML fragment:

```
<w:colDelim w:val="44" />
```

Here, the colDelim element's val attribute specifies that the given external data source is using the comma character (,) to delimit column data, as 44 is the decimal value for the Unicode character representation of a comma. *end Example*]

Parent Elements
odso (§2.14.25)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
    <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.14.6 column (Index of Column Containing Unique Values for Record)

This element specifies the column within the specified external data source that contains unique data for the current record within that data source. This element shall be used in conjunction with the uniqueTag element (§2.14.35) to maintain a relationship between a specific record within an external data source and a given source or merged document. The val attribute on this element shall be interpreted as a zero-based index into the columns specified by the data source, specifying the resulting column as the column in which the uniqueTag element shall be looked up.

If this element specifies a column number which exceeds the number of columns in the specified external data source, then its value shall be ignored.

[*Note*: This information is necessary as part of a mail merge as records may be added or deleted from external data sources, and a means must be provided to maintain record-specific inclusion or exclusion data using the active element (§2.14.1) and the affected external data record when the WordprocessingML document is reconnected to the external data source irrespective of the ordering of the records within the external data source. *end note*]

[*Example*: Consider the following WordprocessingML fragment for the information about a single record in a source document for a mail merge:

```
<w:recipientData>
  <w:active w:val="0" />
  <w:column w:val="12" />
  <w:uniqueTag w:val="258865469" />
</w:recipientData>
```

The external data record associated with this information is specified via the column in the external data source corresponding to the column element with a val attribute equal to 12, which contains a row whose value in this column has a value corresponding to the uniqueTag element (§2.14.35) with a val attribute equal to 258865469. This record will not be imported into the merged WordprocessingML document as the active (§2.14.1) element associated with the given external data record has a val attribute equal to 0.

In other words, when the specified external data source is connected to, the record within the thirteenth column of the external data source that has the contents 258865469, and not populate mapped fields in a merged document with data from that record. *end example*]

Parent Elements
recipientData (§2.14.28)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.14.7 column (Index of Column Being Mapped)

This element specifies the zero-based index of the column within a given external data source which shall be mapped to the local name of a specific MERGEFIELD field (§2.16.5.42) specified by the parent field mapping data. The val attribute specifies this index value, which is used to look up the appropriate column in the data source.

If this element is omitted, or its value exceeds the number of columns in the associated data source, then the index of the referenced column shall be assumed to be 0.

[*Example*: Consider a source document that is connected to an external data source with three columns. Within this external data source, these are three columns are ordered and titled as follows: first, middle, and last, respectively. The following WordprocessingML specifies that when this document was connected to the data source, these columns were ordered in this manner:

```
<w:fieldMapData>
  …
  <w:name w:val="first" />
  <w:column w:val="0" />
</w:fieldMapData>
<w:fieldMapData>
  …
  <w:name w:val="middle" />
  <w:column w:val="1" />
</w:fieldMapData>
<w:fieldMapData>
  …
  <w:name w:val="last" />
  <w:column w:val="2" />
</w:fieldMapData>
```

The WordprocessingML above demonstrates that the column titled first shall be associated with the first column in the external database by specifying a column element with its val attribute equal to 0. In addition, the column titled middle shall be associated with the second column in the external database by specifying a column element with its val attribute equal to 1. Finally, the column titled last shall be associated with the third column in the external database by specifying a column element with its val attribute equal to 2. *end example*]

It is important to realize that the name element's values are a cache of the last time the document was connected to the database, and the indices specified shall be used to connect the field mappings with the columns in the data source. *end example*]

Parent Elements
fieldMapData (§2.14.15)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.14.8 connectString (Data Source Connection String)

This element specifies the connection string used to reconnect to an external data source. The string within this element's val attribute shall contain the connection string that the hosting application shall pass to a external data source access application to enable the WordprocessingML document to be reconnected to the specified external data source.

[*Note*: This string is generally comprised of a series of name/value pairs, delimited by semicolons, determined by the data source access application and the external data source that is accessed. *end note*]

If this string is omitted, then no legacy connection string shall be associated with this mail merge.

This connection string should be ignored under the following conditions:

- The udl element (§2.14.34) is present within the mail merge data
- The dataType element (§2.14.10) is set to `native`
- The current application is able to use the information contained in the odso element (§2.14.25) to access the data source

[*Guidance*: In this case, using the connection string in the udl element will provide an equal or greater amount of connection information for the mail merge data source for clients which support it. *end guidance*]

[*Example*: Consider a merged WordprocessingML document that has been connected to an external data source for the purposes of a mail merge. The following WordprocessingML fragment represents the legacy connection string used to connect to the external data source when the merged WordprocessingML document is reopened:

```
<w:connectString w:val="Provider=Example;Password=Test;User ID=readonly;…" />
```

The connectString element specifies that the string `Provider=Example;Password=Test;User ID=readonly;…` shall be used to enable the given WordprocessingML document to be reconnected to the specified external data source. *end example*]

Parent Elements
mailMerge (§2.14.20)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.14.9 dataSource (Data Source File Path)

This element specifies the relationship whose target is the location of the external data source to be connected to a given WordprocessingML document to perform the mail merge (for a source document) or to find the associated field data (for a merged document).

If this element is omitted, then no file location is specified for the data source for the current mail merge. If no relationship exists with the given relationship ID, or this relationship is not of type http://schemas.openxmlformat.org/officeDocument/2006/relationships/mailMergeSource then this document shall be considered non-conformant.

The data source location may also be ignored under the following conditions:

- The src element (§2.14.30) is present within the mail merge data
- The dataType element (§2.14.10) is set to `native`
- The current application is able to use the information contained in the odso element (§2.14.25) to access the data source

[*Guidance*: In this case, using the data source file path in the src element will provide an equal or greater amount of information for the mail merge data source for clients which can consume it. *end guidance*]

[*Example*: Consider a WordprocessingML source document containing the following mail merge data:

```
<w:mailMerge>
  …
  <w:dataSource r:id="rId1" />
  …
</w:mailMerge>
```

This mail merge's dataSource element specifies via its r:id attribute value of `rId1` that the external data source to be connected to the given WordprocessingML document is the data source targeted by the relationship whose Id attribute is equal to `rId1`. If we examine the corresponding relationship part item for the setting part, as follows:

```
<Relationships>
  <Relationship Id="rId1"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/mailMe
rgeSource" Target="file:///c:/example_file.mdb" TargetMode="External" />
</Relationships>
```

Since the relationship whose Id attribute value is `rId1` specifies the source file path for the data source, that data source effectively specifies a file path of `c:\example_file.mdb`. *end example*]

Parent Elements
mailMerge (§2.14.20)

Attributes	Description
id (Relationship to Part)	Specifies the relationship ID to a specified part. The specified relationship shall match the type required by the parent element:

Attributes	Description
Namespace: .../officeDocument/2006/relationships	• http://schemas.openxmlformats.org/officeDocument/2006/relationships/footer for the footerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/header for the headerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://schemas.openxmlformats.org/officeDocument/2006/relationships/printerSettings for the printerSettings element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Rel">
   <attribute ref="r:id" use="required"/>
</complexType>
```

2.14.10 dataType (Data Source Type)

This element specifies the type of external data source to be connected to via the Dynamic Data Exchange (DDE) system (such as a spreadsheet or database), or the alternative method of data access if the Dynamic Data Exchange system is not used. This setting is purely a suggestion of the data source access mechanism which shall be used, and may be ignored in favor of an alternative mechanism if one is present.

[*Example*: Consider the following WordprocessingML fragment for a mail merge source or merged document:

```
<w:dataType w:val="odbc" />
```

The dataType element's val attribute is equal to odbc, specifying that the given merged WordprocessingML document has been connected to an external data source via the Open Database Connectivity interface. *end example*]

Parent Elements
mailMerge (§2.14.20)

Attributes	Description

Attributes	Description
val (Value)	Specifies the exact type of external data source to which a given merged WordprocessingML document is to be connected. [*Example*: Consider the following WordprocessingML fragment for a mail merge document. `<w:dataType w:val="database" />` The val attribute is equal to `database`, specifying that the given WordprocessingML document has been connected to a database via the Dynamic Data Exchange (DDE) system. *end example*] The possible values for this attribute are defined by the ST_MailMergeDataType simple type (§2.18.59).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MailMergeDataType">
   <attribute name="val" type="ST_MailMergeDataType" use="required"/>
</complexType>
```

2.14.11 destination (Merged Document Destination)

This element specifies what the result which shall be generated when a mail merge is carried out on a given WordprocessingML source document. In other words, this element is used to specify what is to be done with the merged documents that result from populating the fields within a given merged WordprocessingML document with data from the specified external data source.

If this element is omitted, then the default destination of merged documents shall be assumed to be of type `newDocument`.

[*Note*: The aspects of the mail merge outside of connecting to an external data source and populating the fields within a given merged document with external data from the specified external data source are not specified by this Office Open XML Standard.

For example, if a given merged WordprocessingML document contains a destination element with its val attribute equal to `email`, the hosting application may surface a user interface specific to creating emails with the data resulting from populating fields within a given merged WordprocessingML document with external data from the specified external data source. WordprocessingML only provides a flag (via the destination element) to tell the hosting application to surface this user interface. *end note*]

[*Example*: Consider a WordprocessingML source document containing the following WordprocessingML:

```
<w:mailMerge>
  <w:destination w:val="newDocument" />
  …
</w:mailMerge>
```

The destination element's val attribute is set to newDocument, specifying that when the mail merge is carried out, the source document shall be used to generate a specified number of new documents, which may be handled as appropriate. *end example*]

Parent Elements
mailMerge (§2.14.20)

Attributes	Description
val (Mail Merge Merged Document Type)	Specifies the type of merged documents which shall be the result of carrying out a mail merge on a given source WordprocessingML document. [*Example*: Consider the WordprocessingML mail merge data specified as follows: `<w:destination w:val="email" />` This specifies that a given merged WordprocessingML document will be used by the hosting application to generate e-mails containing the static contents of the merged document as well as external data populated into mapped fields. *end example*] The possible values for this attribute are defined by the ST_MailMergeDest simple type (§2.18.60).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MailMergeDest">
   <attribute name="val" type="ST_MailMergeDest" use="required"/>
</complexType>
```

2.14.12 doNotSuppressBlankLines (Remove Blank Lines from Merged Documents)

This element specifies how an application performing the mail merge shall handle blank lines in the merged documents resulting from the mail merge. Typically, when a mail merge is performed, any blank lines which result from lines whose sole contents are merge fields with no content are removed from the merged document in order to prevent extraneous blank lines from appearing in the merged documents. When this element is present, the merged documents which are generated from the mail merge shall not have any blank lines removed before they are sent to their destination format.

If this element is omitted, the merged documents generated from this mail merge shall have all blank lines suppressed if they consist of only merge fields with values consisting of empty strings.

[*Example*: Consider a WordprocessingML document containing a single WordprocessingML field calling for `Test` data as seen in the first column of the table below. If the current record in the mail merge data source contains an empty string for the `Test` column, the resulting merged document would be displayed as follows depending on the setting for the doNotSuppressBlankLines element:

Source Document	`<w:doNotSuppressBlankLines val="on" />`	`<w:doNotSuppressBlankLines val="off" />`
One Two {MERGEFIELD "Test" } Three	One Two Three	One Two Three

With this element set to a value of on, the blank lines in the resulting document shall not be suppressed when the resulting merged documents are created. *end example*]

Parent Elements
mailMerge (§2.14.20)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.14.13 dynamicAddress (Use Country-Based Address Field Ordering)

This element specifies that the contents of the `AddressBlock MERGEFIELD` field shall be dynamically ordered based on the country associated with the current record or if the country-invariant version of the address field

shall be used in its place. [*Rationale*: When a source document is combined with the contents of a data source in order to produce multiple merged documents, it is often necessary to use an address form specific to the destination country for each particular record in the data source, rather than one static address form for all records. *end rationale*] If this element is set to `true`, then the mail merge shall use an address form suited to the country associated with the current record in the external data source.

If this element is omitted, then the form of the address shall be dynamically determined based on the country specified in the current record.

[*Example*: Consider a merged WordprocessingML document that is specified to *not* dynamically create the address field order based on the country associated with the current record. This requirement may be specified using the following WordprocessingML:

```
<w:fieldMapData>
  …
  <w:dynamicAddress w:val="off" />
</w:fieldMapData>
```

The dynamicAddress element is set to a value of `off`, specifying that the dynamic address format shall not be used when performing a mail merge with the specified data source. *end example*]

Parent Elements
fieldMapData (§2.14.15)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.14.14 fHdr (First Row of Data Source Contains Column Names)

This element specifies that a hosting application shall treat the first row of data in the specified external data source as a header row containing the names of each column in the data source, rather than data to populate mapped fields in a merged document. When present, this information shall not change the indices specified in the recipientData elements (§2.14.28), but shall indicate that the first row is not part of the mail merge when it is performed.

If this element is omitted, then the first row of the data source shall not be considered a header row when a mail merge is performed.

[*Example*: Consider a WordprocessingML source document that has been connected to an external data source whose first row of data is not data the hosting application is to populate mapped fields with, but rather contains column names for each column in the data source. This setting on the data source is specified using the following fragment of WordprocessingML:

```
<w:fHdr w:val="on" />
```

The fHdr element specifies that the data source's first row is a header row, rather than regular data to be used in the mail merge. *end example*]

Parent Elements
odso (§2.14.25)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.14.15 fieldMapData (External Data Source to Merge Field Mapping)

This element specifies how a column specified in the external data source that has been connected to a WordprocessingML document shall be mapped to the pre-defined `MERGEFIELD` fields (§2.16.5.42) within the given merged document's contents. Each instance of a fieldMapData element contains the information needed to map one column in the external data source to a single type of pre-defined `MERGEFIELD` field for the purposes of the mail merge in the current document.

[*Example*: Consider a single merged document. The WordprocessingML below demonstrates the mapping of the `Country` column from the external data source to the predefined WordprocessingML `Country or Region` merge field when the merged document is populated with external data as part of a mail merge:

```
<w:odso>
  …
  <w:fieldMapData>
    <w:type w:val="dbColumn" />
    <w:name w:val="Country" />
    <w:mappedName w:val="Country or Region" />
    <w:column w:val="9" />
    …
  </w:fieldMapData>
</w:odso>
```

The fieldMapData element specifies the mapping between the external data source and a single merge field as follows: the child elements specify that the tenth column in the data source, last titled `Country` in the specified external data source when the connection was last made is to be mapped to the predefined WordprocessingML merge field calling for `Country or Region` data.

With the fieldMapData element configured as such, an application may be used in conjunction with this WordprocessingML document to populate the document with data mapped from the specified external data source to fields within the merged document. *end example*]

Parent Elements
odso (§2.14.25)

Child Elements	Subclause
column (Index of Column Being Mapped)	§2.14.7
dynamicAddress (Use Country-Based Address Field Ordering)	§2.14.13
lid (Merge Field Name Language ID)	§2.14.17
mappedName (Predefined Merge Field Name)	§2.14.23
name (Data Source Name for Column)	§2.14.24
type (Merge Field Mapping)	§2.14.32

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OdsoFieldMapData">
   <sequence>
      <element name="type" type="CT_MailMergeOdsoFMDFieldType" minOccurs="0"/>
      <element name="name" type="CT_String" minOccurs="0"/>
      <element name="mappedName" type="CT_String" minOccurs="0"/>
      <element name="column" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="lid" type="CT_Lang" minOccurs="0"/>
      <element name="dynamicAddress" type="CT_OnOff" minOccurs="0"/>
   </sequence>
</complexType>
```

2.14.16 headerSource (Header Definition File Path)

This element specifies the location of a file that contains the column header information which shall be used when connecting to an external data source that does not have column header data specified. Specifically, this element specifies a file that corresponds with the aforementioned external data source. [*Note*: Column headers are needed to enable a hosting application to associate external data source's columns to fields via the fieldMapData element (§2.14.15).

If this element is omitted, then the column header definition data is not specified in an external file and shall be retrieved from the primary data source associated with the mail merge.

[*Example*: Consider a WordprocessingML merged document containing the following WordprocessingML:

```
<w:settings>
  …
  <w:headerSource r:id="rId2" />
  …
</w:settings>
```

This mail merge's headerSource element specifies via its r:id attribute value of `rId2` that the external data source to be used for the column header information for the given WordprocessingML document is the data source targeted by the relationship whose Id attribute is equal to `rId2`. If we examine the corresponding relationship part item for the setting part, as follows:

```
<Relationships>
  <Relationship Id="rId2"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/mailMe
rgeSource" Target="file:///c:/headerData.txt" TargetMode="External" />
</Relationships>
```

Since the relationship whose Id attribute value is `rId2` specifies the column header data file path for the data source effectively specifies a file path of `c:\headerData.txt`. *end example*]

Parent Elements
mailMerge (§2.14.20)

Attributes	Description
id (Relationship to Part) Namespace: .../officeDocument/2006/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the type required by the parent element: • http://schemas.openxmlformats.org/officeDocument/2006/relationships/footer for the footerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/header for the headerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://schemas.openxmlformats.org/officeDocument/2006/relationships/printerSettings for the printerSettings element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Rel">
   <attribute ref="r:id" use="required"/>
</complexType>
```

2.14.17 lid (Merge Field Name Language ID)

This element specifies the language ID for the language which was used to generate the merge field name which was associated with a given column in the data source, as specified by the fieldMapData element (§2.14.15). This element specifies that when this field mapping is processed by an application, it shall interpret the merge field name as the name for the specified language.

If this element is omitted, then the mapped field names specified in the current document may be interpreted using any method desired by the consuming application (i.e. no language data is included with the field mapping information).

[*Example*: Consider the following WordprocessingML fragment for a field mapping for a document to be merged to an external data source. If the merge field name stored in the file corresponds with the U.S. English version of the merge field names, that information would be stored as follows:

```
<w:fieldMapData>
  <w:name w:val="Title" />
  <w:column w:val="3" />
  <w:mappedName w:val="Courtesy Title" />
  <w:lid w:val="en-US" />
</w:fieldMapData>
```

The lid element specifies that the mapping of the data contained in the fourth column of external data source named `Title` to the WordprocessingML of 'Courtesy Title', shall be associated with the U.S. English language as specified by the val attribute equal to `en-US`. *end example*]

Parent Elements
fieldMapData (§2.14.15)

Attributes	Description
val (Language Code)	Specifies an ISO 639-1 letter code or 4 digit hexadecimal code for a specific language. This code is interpreted in the context of the parent XML element. [*Example*: Consider an object which shall specify the English(Canada) language. That object would use the ISO 639-1 letter code of `en-CA` to specify this language. *end example*] The possible values for this attribute are defined by the ST_Lang simple type (§2.18.51).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Lang">
   <attribute name="val" type="ST_Lang" use="required"/>
</complexType>
```

2.14.18 linkToQuery (Query Contains Link to External Query File)

This element specifies that the current WordprocessingML document's query string, stored in the query element (§2.14.26) and used to specify the data to be imported from the external data source, actually contains a reference to an external query file which contains the actual query data to be used against the specified external data source for the mail merge. This query shall mimic a SQL query and be of the following form: `SELECT * FROM <query file path>`.

If this element is omitted, then the query specified for the data source attached to the current document shall be assumed to not be a query containing a link to an external file.

[*Example*: Consider a mail merge source document that uses the linkToQuery element to specify that the query used is stored in the specified external data source as follows:

```
<w:mailMerge>
  …
  <w:linkToQuery />
  <w:query w:val="SELECT * FROM C:\queryExample.txt" />
  …
</w:mailMerge>
```

The linkToQuery element specifies that the query string stored in the query element (§2.14.26) is actually just a reference to an externally stored query file, in this case, an external query file stored at `c:\queryExample.txt`. *end example*]

Parent Elements
mailMerge (§2.14.20)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.14.19 mailAsAttachment (Merged Document To E-Mail Attachment)

This element specifies that, after importing external data into fields to generate a series of destination WordprocessingML documents as e-mails, the resulting documents should be emailed as an attachment rather than the body of the actual e-mail.

If this element is omitted, then its value shall be assumed to be `false` (i.e. the destination source is not an e-mail attachment). If the destination element (§2.14.11) specifies that the merged document destination is not `email`, then this element shall be ignored.

[*Example*: Consider a merged WordprocessingML document that has been connected to an external data source containing three records and that contains the following WordprocessingML in its mail merge properties as follows:

```
<w:mailMerge>
  …
  <w:destination w:val="email" />
  <w:mailAsAttachment />
  …
</w:mailMerge>
```

After the external data has been imported into the merged document's respective merge fields, three emails will be generated (as specified by the destination element (§2.14.11) with a val attribute of `email`, each an attachment consisting of one of the three documents resulting from the mail merge result (rather than just including the merged document as the body of the email). *end example*]

Parent Elements
mailMerge (§2.14.20)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.14.20 mailMerge (Mail Merge Settings)

This element specifies all of the mail merge information for a document that has been connected to an external data source as part of a mail merge operation.

The document which contains this mail merge data may be of one of two types:

- A *source document*, the document which contains all of the information for the mail merge, and is used in conjunction with an application to connect to an external data source and create one document for each record in that data source.
- A *merged document*, a document which contains all of the information for the mail merge as well as a reference to a single specific record which shall be used to populate the values of all of the merge fields in that document.

The information in this element shall contain all data needed to connect to a data source and populate any merge fields in the document with data from that data source.

[*Example*: Consider the following WordprocessingML fragment for a document which is part of a mail merge:

```
<w:mailMerge>
  …
  <w:dataType w:val="spreadsheet" />
  <w:query w:val="SELECT * FROM `Sheet1$`" />
  <w:dataSource r:id="rId1" />
  …
</w:mailMerge>
```

Here, the dataType (§2.14.10) and dataSource (§2.14.9) elements specify that the given document shall be connected to the external data source referenced by the relationship whose id value is equal to `rId1`. While connected to the external data source, the document together with a hosting application may extract data from the external data source as specified by the query (§2.14.26) element. *end example*]

Parent Elements
settings (§2.15.1.78)

Child Elements	Subclause
activeRecord (Record Currently Displayed In Merged Document)	§2.14.2
addressFieldName (Column Containing E-mail Address)	§2.14.3
checkErrors (Mail Merge Error Reporting Setting)	§2.14.4
connectString (Data Source Connection String)	§2.14.8
dataSource (Data Source File Path)	§2.14.9
dataType (Data Source Type)	§2.14.10
destination (Merged Document Destination)	§2.14.11
doNotSuppressBlankLines (Remove Blank Lines from Merged Documents)	§2.14.12
headerSource (Header Definition File Path)	§2.14.16
linkToQuery (Query Contains Link to External Query File)	§2.14.18

Child Elements	Subclause
mailAsAttachment (Merged Document To E-Mail Attachment)	§2.14.19
mailSubject (Merged E-mail or Fax Subject Line)	§2.14.21
mainDocumentType (Source Document Type)	§2.14.22
odso (Office Data Source Object Settings)	§2.14.25
query (Query For Data Source Records To Merge)	§2.14.26
viewMergedData (View Merged Data Within Document)	§2.14.36

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MailMerge">
   <sequence>
      <element name="mainDocumentType" type="CT_MailMergeDocType" minOccurs="1"/>
      <element name="linkToQuery" type="CT_OnOff" minOccurs="0"/>
      <element name="dataType" type="CT_MailMergeDataType" minOccurs="1"/>
      <element name="connectString" type="CT_String" minOccurs="0"/>
      <element name="query" type="CT_String" minOccurs="0"/>
      <element name="dataSource" type="CT_Rel" minOccurs="0"/>
      <element name="headerSource" type="CT_Rel" minOccurs="0"/>
      <element name="doNotSuppressBlankLines" type="CT_OnOff" minOccurs="0"/>
      <element name="destination" type="CT_MailMergeDest" minOccurs="0"/>
      <element name="addressFieldName" type="CT_String" minOccurs="0"/>
      <element name="mailSubject" type="CT_String" minOccurs="0"/>
      <element name="mailAsAttachment" type="CT_OnOff" minOccurs="0"/>
      <element name="viewMergedData" type="CT_OnOff" minOccurs="0"/>
      <element name="activeRecord" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="checkErrors" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="odso" type="CT_Odso" minOccurs="0"/>
   </sequence>
</complexType>
```

2.14.21 mailSubject (Merged E-mail or Fax Subject Line)

This element specifies the text which shall appear in the subject line of the e-mails or faxes that result after the actions of a mail merge have imported external data into fields within a merged WordprocessingML document whose destination, as specified in the destination element (§2.14.21), is `email` or `fax`.

If this element is omitted, then no subject line text shall be associated with each merged document produced via a mail merge using the specified mail merge data. If the destination element (§2.14.11) specifies that the merged document destination is not `email` or `fax`, this element shall be ignored.

[*Example*: Consider a merged WordprocessingML document containing fields and the following WordprocessingML as part of its mail merge data:

```
<w:mailMerge>
  …
  <w:destination w:val="email" />
  <w:mailSubject w:val="Example Subject Line" />
  …
</w:mailMerge>
```

The mailSubject element specifies that after the specified external data has been imported into the specified fields in the merged document, each record merged shall result in a single e-mail message, each with their subject line reading `Example Subject Line`. *end example*]

Parent Elements
mailMerge (§2.14.20)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.14.22 mainDocumentType (Source Document Type)

This element specifies the type of a given WordprocessingML source document.

If this element is omitted, then its value shall be assumed to be `formLetters`.

[*Note*: This element is generally used in conjunction with the behavior of an application to customize aspects of the mail merge user interface and experience independent of the WordprocessingML file format. For example, if a given WordprocessingML merged document contains a mainDocumentType element with its val attribute equal to `envelopes`, the hosting application may surface a piece of user interface specific to creating envelopes when the given document is opened.

In addition, what a hosting application does with the documents that result from importing external data into specified fields can be determined based on the mainDocumentType element, but other than this, is independent of a given merged document's WordprocessingML. For example, if a given merged WordprocessingML document contains a mainDocumentType element with its val attribute equal to `email`, the hosting application may call a email service after importing external data into specified fields, in order to generate emails containing the resulting documents.

WordprocessingML simply provides the mainDocumentType that can serve as a trigger for an application to surface user interface specific to a type of mail merge. *end note*]

[*Example*: Consider the WordprocessingML below:

```
<w:mailMerge>
  <w:mainDocumentType w:val="formLetters" />
  …
</w:mailMerge>
```

In this example, the source document is of the `formLetters` type, as specified by the mainDocumentType element's val attribute being equal to `formLetters`. *end example*]

Parent Elements
mailMerge (§2.14.20)

Attributes	Description
val (Mail Merge Source Document Type)	Specifies the type of source document which is specified by the given WordprocessingML document. [*Example*: Consider the WordprocessingML below: `<w:mainDocumentType w:val="formLetters" />` This WordprocessingML specifies that a given source document is a `formLetters` document. This setting implies nothing about the file, but may be interpreted by an

Attributes	Description
	application as desired. *end example*] The possible values for this attribute are defined by the ST_MailMergeDocType simple type (§2.18.61).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MailMergeDocType">
   <attribute name="val" type="ST_MailMergeDocType" use="required"/>
</complexType>
```

2.14.23 mappedName (Predefined Merge Field Name)

This element specifies the predefined WordprocessingML `MERGEFIELD` field name which shall be mapped to the column number specified by the column element (§2.14.7) within this field mapping. [*Guidance*: This element allows the current column from the specified data source to be mapped to a predefined field name, allowing applications to have one standard set of field names to use regardless of the data source column names, for example, to create the address formats to place into an `ADDRESSBLOCK` field. *end guidance*]

If this element is omitted, then the current data source column mapping shall not have a predefined merge field name mapped to its contents, and shall only be referenced via the data source column name specified by the name element (§2.14.24) when referenced by one or more `MERGEFIELD` fields. If the application does not have a predefined merge field whose name matches the name specified using the val attribute, then this element may be ignored.

[*Example*: Consider the following WordprocessingML fragment, representing two columns from an external data source which have been mapped to the built-in fields `First Name` and `Last Name`, respectively:

```
<w:fieldMapData>
  <w:column w:val="0" />
  <w:name w:val="Column Name A" />
  <w:mappedName w:val="First Name" />
  …
</w:fieldMapData>
<w:fieldMapData>
  <w:column w:val="1" />
  <w:name w:val="Column Name B" />
  <w:mappedName w:val="Last Name" />
  …
</w:fieldMapData>
```

The first and second columns, specified by the column element values of `0` and `1` respectively, specify that the predefined WordprocessingML field names `First Name` and `Last Name` are mapped to the columns of the external data source, and the data source names for those columns are `Column Name A` and `Column Name B`, respectively.

Therefore, if MERGEFIELD fields calling for First Name and Last Name are inserted in a WordprocessingML document connected to the external data source with the field mappings specified above, when the mail merge takes place, the data from the first and second column will populate the fields calling for First Name and Last Name data within the merged WordprocessingML document. *end example*]

Parent Elements
fieldMapData (§2.14.15)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's styleId.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.14.24 name (Data Source Name for Column)

This element specifies the column name within a given external data source for the column whose index is specified via the column element (§2.14.7). This data source name provides a column name which shall be used to map a specific MERGEFIELD field in the document, as specified by the parent field mapping data. The val

attribute specifies the name of this column in the data source when the connection is initially established, which is then used permanently to link columns in the database to MERGEFIELD fields in the document.

If this element is omitted, no data source name is provided for the current column.

[*Example*: Consider a source document that is connected to an external data source with three columns. Within this external data source, these are three columns are ordered and titled as follows: first, middle, and last, respectively. The following WordprocessingML specifies that when this document was connected to the data source, these columns were ordered in this manner:

```
<w:fieldMapData>
  …
  <w:name w:val="first" />
  <w:column w:val="0" />
</w:fieldMapData>
<w:fieldMapData>
  …
  <w:name w:val="middle" />
  <w:column w:val="1" />
</w:fieldMapData>
<w:fieldMapData>
  …
  <w:name w:val="last" />
  <w:column w:val="2" />
</w:fieldMapData>
```

The WordprocessingML above demonstrates that the column name first shall be associated with the first column in the external database by specifying a column element with its val attribute equal to 0. In addition, the column name middle shall be associated with the second column in the external database by specifying a column element with its val attribute equal to 1. Finally, the column name last shall be associated with the third column in the external database by specifying a column element with its val attribute equal to 2. *end example*]

Parent Elements
fieldMapData (§2.14.15)

Attributes	Description
val (String Value)	Specifies that its contents will contain a string. The contents of this string are interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML fragment:

Attributes	Description
	<w:pPr> <w:pStyle w:val="heading1" /> </w:pPr> The value of the val attribute is the ID of the associated paragraph style's styleId. However, consider the following fragment: <w:sdtPr> <w:alias w:val="SDT Title Example" /> ... </w:sdtPr> In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.14.25 odso (Office Data Source Object Settings)

This element specifies a group of additional settings for the mail merge information which comprise an extension to the standard settings stored with a mail merge which performs two functions:

- First, it provides additional information about the mail merge data source, specifically: information about how to map the columns in the data source to MERGEFIELD fields and information about records which shall be included and excluded when creating merged documents, and column delimiters used in text data sources. This information may be used regardless of the value of the dataType element (§2.14.10) when it is present.
- Second, it provides an alternate set of connection information which should be used when the dataType element (§2.14.10) specifies a value of native. This alternate connection string provides additional connection information for applications which choose to support the ODSO connection string syntax. If the dataType element (§2.14.10) specifies that the data type of the current mail merge is not native, then the second group of settings specified within this element shall be ignored in favor of their non-ODSO equivalents.

[*Example*: Consider the WordprocessingML for a source document whose mail merge includes mail merge information including ODSO settings as follows:

```
<w:odso>
  <w:udl w:val="…" />
  <w:table w:val="Sheet1$" />
  <w:src r:id="rId1" />
  <w:colDelim w:val="9" />
  <w:fHdr w:val="1" />
  <w:fieldMapData>
    <w:type w:val="dbColumn" />
    <w:name w:val="Title" />
    <w:mappedName w:val="Courtesy Title" />
    <w:lid w:val="en-US" />
  </w:fieldMapData>
  <w:recipientData r:id="rId2" />
  …
</w:odso>
```

The odso element and its child elements provide all of the information specified above that is needed to carry out a mail merge with the current document. This includes alternate connection information using the udl, table, and src elements, and additional mail merge information in the other child elements. *end example*]

Parent Elements
mailMerge (§2.14.20)

Child Elements	Subclause
colDelim (Column Delimiter for Data Source)	§2.14.5
fHdr (First Row of Data Source Contains Column Names)	§2.14.14
fieldMapData (External Data Source to Merge Field Mapping)	§2.14.15
recipientData (Reference to Inclusion/Exclusion Data for Data Source)	§2.14.27
src (ODSO Data Source File Path)	§2.14.30
table (Data Source Table Name)	§2.14.31
type (ODSO Data Source Type)	§2.14.33
udl (UDL Connection String)	§2.14.34

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Odso">
   <sequence>
      <element name="udl" type="CT_String" minOccurs="0"/>
      <element name="table" type="CT_String" minOccurs="0"/>
      <element name="src" type="CT_Rel" minOccurs="0"/>
      <element name="colDelim" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="type" type="CT_MailMergeSourceType" minOccurs="0"/>
      <element name="fHdr" type="CT_OnOff" minOccurs="0"/>
      <element name="fieldMapData" type="CT_OdsoFieldMapData" minOccurs="0" maxOccurs="unbounded"/>
      <element name="recipientData" type="CT_Rel" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

2.14.26 query (Query For Data Source Records To Merge)

This element contains the Structured Query Language string that shall be run against the specified external data source to return the set of records from the external data which shall be imported into merged WordprocessingML documents when the mail merge operation is performed.

If this element is omitted, then no query shall be associated with the current data source.

[*Example*: Consider a WordprocessingML document that has been connected to an external database. In addition, consider that the data specifies that the table within the database titled `Documentation` shall be the specific table whose data is imported. This shall be specified in WordprocessingML as follows:

```
<w:query w:val="SELECT * FROM Documentation" />
```

The query element specifies the syntax for the data source query via its val attribute. *end example*]

Parent Elements
mailMerge (§2.14.20)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.

Attributes	Description

	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.14.27 recipientData (Reference to Inclusion/Exclusion Data for Data Source)

This element shall specify a reference to the part which contains data about whether the set of records in the associated data source have been explicitly included or excluded from the specified mail merge. Only those records which shall not be used to generate merged WordprocessingML documents must be stored within the referenced part, as all records shall be merged by default as part of the mail merge operation. [*Guidance*: Applications may choose to store only those records which are excluded for efficiency, or a list of all records in order to determine which set of records were added/removed between mail merge operations. *end guidance*]

[*Rationale*: When defining a mail merge, it is possible that a user wishes to connect to a specified data source, but specify only a subset of the records returned by the query specified by the query element (§2.14.26) which shall be merged as part of the mail merge operation. This element allows applications to utilize a separate part to store this information, either the shared part defined by this Office Open XML Standard, or an application-specific part as needed. *end rationale*]

If the relationship type of the relationship specified by this element is not http://schemas.openxmlformats.org/officeDocument/2006/mailMergeRecipientData, is not present, or does not have a TargetMode attribute value of `Internal`, then the document shall be considered non-conformant. If an application cannot process external content of the content type specified by the targeted part, then it may be ignored.

This Office Open XML Standard defines one shared mechanism for storing this data: using the Mail Merge Recipient Data part. This mechanism shall be used if the associated data source has a column which may be used as the unique key. However, when using data sources which do not have a unique key, applications may store their own part (of an application-defined content type) using this relationship.

[*Example*: Consider a WordprocessingML document which is a mail merge source document, containing inclusion/exclusion data for the data source. The document settings part would contain the mail merge data:

```
<w:settings>
  …
  <w:mailMerge>
     …
     <w:odso>
       …
       <w:recipientData r:id="recipient1" />
     </w:odso>
  </w:mailMerge>
</w:settings>
```

The recipientData element specifies that the external content targeted by the relationship with an ID of `recipient1` contains the recipient inclusion/exclusion data for the mail merge operation. Examining the contents of the corresponding relationship part item, we can see the targets for that relationship:

```
<Relationships … >
  …
  <Relationship Id="recipient1" TargetMode="Internal"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/mailMe
rgeRecipientData" Target="recipientData.xml" />
  …
</Relationships>
```

The corresponding relationship part item shows that the file containing this data is located next to the main document and is named `recipientData.xml`. *end example*]

Parent Elements
odso (§2.14.25)

Attributes	Description
id (Relationship to Part) Namespace: …/officeDocument/2006/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the type required by the parent element: • http://schemas.openxmlformats.org/officeDocument/2006/relationships/footer for the footerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/header for the headerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://schemas.openxmlformats.org/officeDocument/2006/relationships/pr

Attributes	Description
	interSettings for the printerSettings element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Rel">
   <attribute ref="r:id" use="required"/>
</complexType>
```

2.14.28 recipientData (Data About Single Data Source Record)

This element specifies information about a single record within an external data source. If a record shall be merged into a merged document, then no information is needed about that record within this part. However, if a given record shall not be merged into a merged document, then the value of the unique key for that record shall be stored within the uniqueTag element as a child of this element (along with the active element) to indicate this exclusion.

[*Note*: This mapping is necessary in place of simply using the element order to correspond to the record indices in the external data source, as records may be added or deleted from external data sources, and a means must be provided to maintain WordprocessingML external record specific data like that specified in the active element (§2.14.1) and the corresponding external data record when the WordprocessingML document is reconnected to the external data source; irrespective of the ordering of the records within the external data source. In other words, this element, and its child elements enable merged WordprocessingML documents to maintain the relationship between the records within an external data and record specific WordprocessingML parameters. *end note*]

[*Example*: Consider a merged WordprocessingML document that:

Annex K. Has been connected to a specified external data source containing three records; and

13. Has been configured by the hosting application to not populate a merged document with the record pertaining to `John Smith` in the external data source.

Consider also that the first time the given WordprocessingML document was connected to the external data source, `John Smith`'s record was in the second record in the data source.

When this merged document is connected to the external data source the recipientData element may be used to store the number and value of the column containing the unique key for each data record within the external

data source including John Smith's. This setting is represented using the following WordprocessingML to use the hash codes within the recipientData element to uniquely identify the three records within the external data source.

```
<w:recipientData>
  <w:column w:val="1" />
  <w:uniqueTag w:val="1408613399" />
</w:recipientData>
<w:recipientData>
  <w:column w:val="1" />
  <w:active w:val="0" />
  <w:uniqueTag w:val="870254691" />
</w:recipientData>
<w:recipientData>
  <w:column w:val="1" />
  <w:uniqueTag w:val="1107777181" />
</w:recipientData>
```

Here, the first, second (John Smith record), and third records within the specified data source whose unique key values are 1408613399, 870254691, and 1107777181 have been associated with with recipient data via the active element to specify that the record associated with the given record (John Smith's record) shall not be used to populate a merged WordprocessingML document.

With these association in place, if a fourth record is added to the given external data source above John Smith's record, when the given merged WordprocessingML document is reconnected to the external data source, the hosting application will still know that John Smith's record shall not be used to populate a merged WordprocessingML document as it is associated via its unique key value and is not dependent on the given record's ordinal position within the external data source. *end example*]

Parent Elements
Root element of WordprocessingML Mail Merge Recipient Data partrecipients (§2.14.29)

Child Elements	Subclause
active (Record Is Included in Mail Merge)	§2.14.1
column (Index of Column Containing Unique Values for Record)	§2.14.6
uniqueTag (Unique Value for Record)	§2.14.35

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RecipientData">
   <sequence>
      <element name="active" type="CT_OnOff" minOccurs="0"/>
      <element name="column" type="CT_DecimalNumber" minOccurs="1"/>
      <element name="uniqueTag" type="xsd:base64Binary" minOccurs="1"/>
   </sequence>
</complexType>
```

2.14.29 recipients (Inclusion/Exclusion Data for Data Source)

This element specifies all of the inclusion/exclusion data for the contents of the specified mail merge data source. It is the root element for the Mail Merge Recipient Data part.

[*Example*: Consider a document which is the source document for a mail merge operation. If two records of the three specified by the data source were excluded from the mail merge, the resulting recipient data part would appear as follows:

```
<w:recipients>
  <w:recipientData>
    <w:active w:val="false"/>
    …
  </w:recipientData>
  <w:recipientData>
    <w:active w:val="false"/>
    …
  </w:recipientData>
  <w:recipientData>
    …
  </w:recipientData>
</w:recipients>
```

The recipients element contains all of the recipient inclusion/exclusion data for this mail merge document. *end example*]

Child Elements	Subclause
recipientData (Data About Single Data Source Record)	§2.14.28

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Recipients">
   <sequence>
      <element name="recipientData" type="CT_RecipientData" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

2.14.30 src (ODSO Data Source File Path)

This element specifies the relationship whose target is the location of the external data source to be connected to a given WordprocessingML document to perform the mail merge (for a source document) or to find the associated field data (for a merged document) when the merge type, specified using the dataType element (§2.14.10), is set to `native`.

If this element is omitted, then no file location is specified for the data source for the current mail merge. If no relationship exists with the given relationship ID, or this relationship is not of the Mail Merge Data Source relationship type, then this document shall be considered non-conformant.

The data source location is only used under the following conditions:

- The dataType element (§2.14.10) is set to `native`
- The current application is able to use the information contained in the odso element (§2.14.25) to access the data source

[*Guidance*: In this case, using the data source file path in the src element will provide an equal or greater amount of information for the mail merge data source for clients which can consume it. *end guidance*]

[*Example*: Consider a WordprocessingML source document containing the following mail merge data:

```
<w:odso>
  …
  <w:src r:id="rId1" />
  …
</w:odso>
```

This mail merge's src element specifies via its r:id attribute value of `rId1` that the external data source to be connected to the given WordprocessingML document is the data source targeted by the relationship whose Id attribute is equal to `rId1`. If we examine the corresponding relationship part item for the setting part, as follows:

```
<Relationships>
  <Relationship Id="rId1"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/mailMe
rgeSource" Target="file:///c:/example_file.mdb" TargetMode="External" />
</Relationships>
```

Since the relationship whose Id attribute value is `rId1` specifies the source file path for the data source, that data source effectively specifies a file path of `c:\example_file.mdb`. *end example*]

Parent Elements
odso (§2.14.25)

Attributes	Description
id (Relationship to Part) Namespace: .../officeDocument/2006/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the type required by the parent element: • http://schemas.openxmlformats.org/officeDocument/2006/relationships/footer for the footerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/header for the headerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://schemas.openxmlformats.org/officeDocument/2006/relationships/printerSettings for the printerSettings element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Rel">
   <attribute ref="r:id" use="required"/>
</complexType>
```

2.14.31 table (Data Source Table Name)

This element specifies the particular set of data that a source or merged WordprocessingML document shall be connected to within an external data source containing multiple data sets. In other words, when connecting to a WordprocessingML document to an external data source that may have more than one repository of data within it, such as a database with multiple tables or a spreadsheet with multiple worksheets, this element is used to distinguish the specific table or spreadsheet from which data will be imported from within the external data source.

[*Example*: Consider a WordprocessingML document that has been connected to database containing two tables named `Table One` and `Table Two`, respectively. To specify that the mail merge shall import data from `Table One` into the WordprocessingML document, this requirement would be specified using the following WordprocessingML:

```
<w:odso>
  …
  <w:table w:val="Table One" />
  …
</w:odso>
```

The table element with a value of `Table One` specifies that the external data shall be retrieved from this table in the data source. *end example*]

Parent Elements
odso (§2.14.25)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.14.32 type (Merge Field Mapping)

This element specifies if a given mail merge field has been mapped to a column in the given external data source or not.

If this element is omitted, then the field mapping shall be considered to be of type `null` (i.e. not mapped).

[*Example*: Consider the WordprocessingML for a single field mapping within a mail merge source document:

```
<w:odso>
  …
  <w:fieldMapData>
    <w:type w:val="dbColumn" />
    <w:name w:val="Country" />
    <w:mappedName w:val="Country or Region" />
    <w:column w:val="9" />
    …
  </w:fieldMapData>
</w:odso>
```

In this example, the `country` column within the given external data source shall be mapped to the mail merge field `Country or Region`, as specified by the type element's val attribute being equal to `dbColumn`. *end example*

Parent Elements
fieldMapData (§2.14.15)

Attributes	Description
val (Merge Field Mapping Type)	Specifies if the given mail merge field has been mapped to a column in the given external data source (i.e. if the merge field mapping is valid or not). [*Example*: Consider the following WordprocessingML fragment for a mail merge source or merged document: `<w:type w:val="null" />` In this example, the given mail merge field shall not be mapped to a column in the given external data source, as specified by the type element's val attribute being equal to `null`. *end example*] The possible values for this attribute are defined by the ST_MailMergeOdsoFMDFieldType simple type (§2.18.62).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MailMergeOdsoFMDFieldType">
   <attribute name="val" type="ST_MailMergeOdsoFMDFieldType" use="required"/>
</complexType>
```

2.14.33 type (ODSO Data Source Type)

This element specifies the type of external data source to be connected to via as part of the ODSO connection information for this mail merge. This setting is purely a suggestion of the data source type which is being used for this mail merge, and may be ignored in favor of an alternative mechanism if one is present.

[*Example*: Consider the following WordprocessingML fragment for a mail merge source or merged document:

```
<w:type w:val="database" />
```

The type element's val attribute is equal to `database`, specifying that the given merged WordprocessingML document has been connected to an external data source via the ODSO settings, and that the resulting data source was a database. *end example*]

Parent Elements
odso (§2.14.25)

Attributes	Description
val (Data Source Type Value)	Specifies the type of an external data source used for a mail merge operation. [*Example*: Consider the following WordprocessingML fragment for a mail merge source or merged document: `<w:type w:val="text" />` The val attribute is equal to `text`, specifying that the given merged WordprocessingML document has been connected to an external data source via the ODSO settings, and that the resulting data source was a text file. *end example*] The possible values for this attribute are defined by the ST_MailMergeSourceType simple type (§2.18.63).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MailMergeSourceType">
   <attribute name="val" use="required" type="ST_MailMergeSourceType"/>
</complexType>
```

2.14.34 udl (UDL Connection String)

This element specifies the Universal Data Link (UDL) connection string used to reconnect to an external data source. The string within this element's val attribute shall contain the connection string that the hosting

application shall pass to a external data source access application to enable the WordprocessingML document to be reconnected to the specified external data source.

If this string is omitted, then no UDL connection string shall be associated with the ODSO data for this mail merge.

This connection string is only used under the following conditions:

- The dataType element (§2.14.10) is set to `native`
- The current application is able to use the information contained in the odso element (§2.14.25) to access the data source

[*Guidance*: In this case, using the connection string in the udl element will provide an equal or greater amount of information for the mail merge data source for clients which can consume it. *end guidance*]

[*Example*: Consider a merged WordprocessingML document that has been connected to an external data source for the purposes of a mail merge. The following WordprocessingML fragment represents the legacy connection string used to connect to the external data source when the merged WordprocessingML document is reopened:

```
<w:udl w:val="Provider=Example;Password=Test;User ID=readonly;…" />
```

The udl element specifies that the string `Provider=Example;Password=Test;User ID=readonly;…` shall be used to enable the given WordprocessingML document to be reconnected to the specified external data source. *end example*]

Parent Elements
odso (§2.14.25)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`

Attributes	Description
	<w:alias w:val="SDT Title Example" /> ... </w:sdtPr> In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.14.35 uniqueTag (Unique Value for Record)

This element specifies the contents of a given record within the specified external data source, in the column containing unique data for every record within the external data source. given external data source. This element is used in conjunction with the column (§2.14.7) element to maintain a relationship between the records within an external data source and a given merged WordprocessingML document.

The contents of this attribute shall be the base64-encoded value of the unique tag value as specified by the data source.

[*Note*: This information is necessary as part of a mail merge as records may be added or deleted from external data sources, and a means must be provided to maintain record-specific inclusion or exclusion data using the active element (§2.14.1) and the affected external data record when the WordprocessingML document is reconnected to the external data source irrespective of the ordering of the records within the external data source. *end note*]

[*Example*: Consider the following WordprocessingML fragment for the information about a single record in a source document for a mail merge:

```
<w:recipientData>
  <w:active w:val="0" />
  <w:column w:val="12" />
  <w:uniqueTag w:val="258865469" />
</w:recipientData>
```

The external data record associated with this information is specified via the column in the external data source corresponding to the column element with a val attribute equal to 12, which contains a row whose value in this column has a value corresponding to the uniqueTag element with a val attribute equal to 258865469. This record will not be imported into the merged WordprocessingML document as the active (§2.14.1) element associated with the given external data record has a val attribute equal to 0.

In other words, when the specified external data source is connected to, the record within the thirteenth column of the external data source that has the contents 258865469, and not populate mapped fields in a merged document with data from that record. *end example*]

The possible values for this element are defined by the XML Schema base64Binary datatype.

Parent Elements
recipientData (§2.14.28)

2.14.36 viewMergedData (View Merged Data Within Document)

This element specifies that a specific merged document shall display the data from the specified external data source where merge fields have been inserted. The activeRecord element (§2.14.2) is used to specify which record within the external data source is to have its applicable data displayed where applicable within the WordprocessingML merged document.

If the activeRecord element is not present in the WordprocessingML for the document with the viewMergedData's val attribute equal to on, the hosting application may behave as if the acitveRecord element's val attribute was equal to 1. This element is ignored if the viewMergedData (§2.14.36) element is not present or present with a val attribute equal to `Off`, `0`, or `false`.

[*Example*: Consider a merged WordprocessingML document containing two WordprocessingML fields calling for `Courtesy Title` and `Last Name` data and a sample text paragraph. Also, note that the external data source this merged document is connected to contains two records, both containing name and address information, with the first record pertaining to `Mr. Doe`, and the second pertaining to `Ms. Smith`.

This table below illustrates the necessary WordprocessingML to display applicable data from the specified external data source within the merged document where fields have been inserted:

`<w:viewMergedData val="off"/>`	`<w:viewMergedData val="on" />` `<w:activeRecord w:val="1" />`	`<w:viewMergedData val="off" />` `<w:activeRecord w:val="1" />`
Dear {MERGEFIELD "Courtesy Title" \m} {MERGEFIELD "Last Name" \m}, Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sincerely,	Dear Mr. Doe: Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sincerely,	Dear {MERGEFIELD "Courtesy Title" \m} {MERGEFIELD "Last Name" \m}, Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sincerely,

The viewMergedData element specifies that the specified record in the external data source shall be displayed in place of merge fields in the current document. *end example*]

Parent Elements
mailMerge (§2.14.20)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15 Settings

Within a WordprocessingML document, *settings* specify stored preferences which shall be used when processing the contents of the document. These settings are typically divided into three categories:

- *Document Settings* - settings which influence the appearance and behavior of the current document, as well as store document-level state.
- *Compatibility Settings* - settings which tell applications to perform behaviors which are designed to maintain visual output of previous word processing applications. These settings are for backward compatibility and are all ignorable.
- *Web Settings* - settings which affect how this document shall be handled when it is saved as HTML. These settings exist primarily for backward compatibility reasons and are all ignorable.

The first two groups are stored in the Document Settings part, and the last group is stored in the Web Settings part.

2.15.1 Document Settings

The first group of settings stored in WordprocessingML is document settings. These settings specify all document-level properties which affect the handling of the current document.

[*Example*: Consider the following WordprocessingML fragment for the document settings in a WordprocessingML document:

```
<w:settings>
  <w:defaultTabStop w:val="720" />
  <w:characterSpacingControl w:val="dontCompress" />
</w:settings>
```

The settings element contains all of the document settings for this document. In this case, the two settings applied are automatic tab stop increments of 0.5" using the defaultTabStop element (§2.15.1.24), and no character level whitespace compression using the characterSpacingControl element (§2.15.1.18). *end example*]

2.15.1.1 activeWritingStyle (Grammar Checking Settings)

This element specifies information about the parameters of the grammar checking which was performed on the contents of the current WordprocessingML document. [*Note*: This information may be used as desired by applications; for example, to determine if the current grammar checking state, specified by the proofState element (§2.15.1.65) is sufficient. *end note*]

[*Example*: Consider the following WordprocessingML fragment from the document settings:

```
<w:activeWritingStyle w:lang="en-CA" w:vendorID="64" w:dllVersion="131078"
w:nlCheck="1" w:optionSet="0" />
```

The activeWritingStyle element's lang attribute specifies that the English (Canada) language setting for grammatical and stylistic checks shall be applied; the vendorID attribute specifies information about the vendor associated with the DLL used to perform the grammatical and stylistic checks; the dllVersion attribute specifies the version of this DLL; the nlCheck attribute specifies if natural language checks were performed or not; and the optionSet element specifies that the hosting application should allow its grammar engine to check both the grammar and style of the given WordprocessingML document, if that functionality is available. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
appName (Application Name)	Specifies the name of the application which specified the grammatical settings contained on the attributes for this element. If an application reads these settings and does not understand the value of this attribute, then its settings may be ignored and the application's default settings used instead.

Attributes	Description
	[*Example*: Consider the WordprocessingML below: `<w:activeWritingStyle … w:appName="testApp"/>` The appName attribute has a value of `testApp`, specifying that the application called `testApp` specified the grammar checking rules of the given WordprocessingML document. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).
checkStyle (Check Stylistic Rules With Grammar)	Specifies if the grammar content checking performed on this document included stylistic rules for the document content. If specified, applications which support this functionality shall check stylistic rules as well as grammatical ones when checking the grammatical content of this document. [*Example*: Consider the WordprocessingML below: `<w:activeWritingStyle … w:checkStyle="false"/>` The checkStyle attribute has a value of `false`, specifying that hosting applications shall only check grammatical rules of the given WordprocessingML document. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
dllVersion (Grammatical Check Engine Version)	Specifies the version of the engine that was used to check the grammatical content of the WordprocessingML document. [*Example*: Consider the following WordprocessingML fragment: `<w:activeWritingStyle … w:dllVersion="131078" />` The dllVersion attribute specifies that the writing style DLL version used to check the writing style of is the writing style DLL version associated with the decimal number `131078`. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
lang (Writing Style Language)	Specifies the language of the engine used to perform the grammatical content checking. [*Example*: Consider the following WordprocessingML fragment: `<w:activeWritingStyle w:lang="en-CA" …/>` The lang attribute has a value of `en-CA`, therefore the grammatical check language is specified as English (`en`) and Canada (`CA`), resulting in use of the English (Canada) grammar checker. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_Lang simple type (§2.18.51).
nlCheck (Natural Language Grammar Check)	Specifies whether the engine that was used to check the grammatical content of the WordprocessingML document performed natural language-based analysis. [*Example*: Consider the WordprocessingML below: `<w:activeWritingStyle … w:nlCheck="1" />` The nlCheck attribute has a value of 1, specifying that the writing style DLL supported natural language analysis . *end example*]. The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
vendorID (Grammatical Engine ID)	Specifies a value indicating a unique ID for the writing style engine that was used to check the grammatical content of the WordprocessingML document. [*Example*: Consider the WordprocessingML below: `<w:activeWritingStyle … w:vendorID="64"/>` The vendorId attribute has a value of 64, specifying that the grammatical checker used is identified by the decimal number 64.*end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_WritingStyle">
   <attribute name="lang" type="ST_Lang" use="required"/>
   <attribute name="vendorID" type="ST_DecimalNumber" use="required"/>
   <attribute name="dllVersion" type="ST_DecimalNumber" use="required"/>
   <attribute name="nlCheck" type="ST_OnOff" use="optional"/>
   <attribute name="checkStyle" type="ST_OnOff" use="required"/>
   <attribute name="appName" type="ST_String" use="required"/>
</complexType>
```

2.15.1.2 alignBordersAndEdges (Align Paragraph and Table Borders with Page Border)

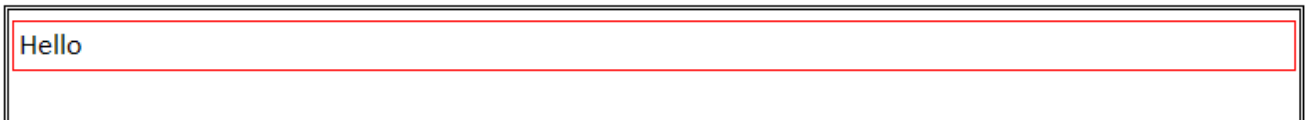
This element specifies that paragraph borders specified using the pBdr element (§2.3.1.24) and table borders using the tblBorders element (§2.4.37) shall be adjusted to align with extents of the page border defined using the pgBorders element (§2.6.10) if the spacing between these borders is less than or equal to 10.5 points (one character width) or less from the page border. The presence of this setting shall ensure there are no gaps of one character width or less between adjoining page and paragraph/table borders, as borders which are perfectly aligning shall not be displayed in favor of the intervening page border.

If this element is omitted, then borders shall not be automatically adjusted to prevent gaps of less than one character width. If the page border is not measured from the text extents using a value of `text` in the offsetFrom attribute on the pgBorders element, then it may be ignored.

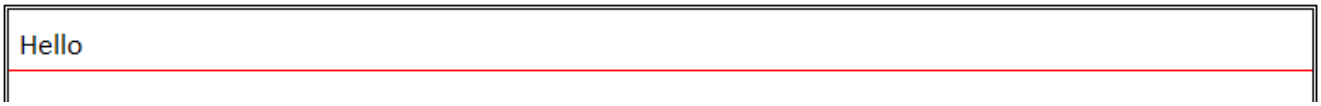
[*Example*: Consider the following WordprocessingML fragment from the document settings:

```
<w:alignBordersAndEdges w:val="true"/>
```

The alignBordersAndEdges element has a value of `true` specifying that borders shall be adjusted to prevent gaps of less than one character width. If a document has a page border specified to appear 4 points from the text extents, and within that page a paragraph border specified to appear one point from the text extents, that would normally appear like this:



If this element is present, then those gaps (which are all of three points in width) shall be adjusted to ensure that the borders align exactly and the paragraph border is suppressed:



end example]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.3 alwaysMergeEmptyNamespace (Do Not Mark Custom XML Elements With No Namespace As Invalid)

This element specifies whether custom XML markup specified via the customXml element which has no associated namespace shall be treated as an error and moved into a special error namespace (for the purposes of validation) when the document is opened. If this element is turned on, when an application determines that the current XML markup is in the empty namespace, those elements shall not automatically be moved into an error namespace.

If this element is not present in a WordprocessingML document than custom XML markup which has no associated namespace shall be treated as an error and moved into a special error namespace when the document is opened.

[*Example*: Consider a WordprocessingML document which should not automatically flag empty namespace XML as invalid. This requirement would be specified using the following WordprocessingML:

```
<w:alwaysMergeEmptyNamespace w:val="true"/>
```

The alwaysMergeEmptyNamespace element's val attribute has a value of `true` specifying that custom XML markup in the empty namespace shall never be treated as an error. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.4 alwaysShowPlaceholderText (Use Custom XML Element Names as Default Placeholder Text)

This element specifies that each custom XML element specified using the customXml element within this document shall always show some form of in-document placeholder text presentation when it contains no run content. If the placeholder element (§2.5.2.24) is present in the custom XML element's properties, then this is the placeholder text displayed and this effect has no effect. If the placeholder element is omitted, then the application shall use the name of the element to generate default placeholder text in its place.

If this element is omitted, then custom XML markup which does not contain a placeholder element within its properties shall not display any placeholder text.

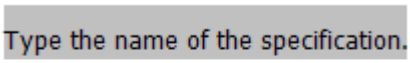
[*Example*: Consider the following WordprocessingML fragment from the document settings:

```
<w:alwaysShowPlaceholderText w:val="true" />
```

The alwaysShowPlaceholderText element has a value of `true`, which specifies that placeholder text shall be generated using the element's name if no placeholder text is present. If two custom XML elements are defined as follows:

```
<w:customXml w:name="spec" … >
  <w:customXmlPr>
    <w:placeholder w:val="Type the name of the specification." />
  </w:customXmlPr>
</w:customXml>
…
<w:customXml w:name="spec" … >
</w:customXml>
```

The first custom XML element has placeholder text, and the second doesn't, so if this element is omitted, these two elements might be displayed as follows:



Notice that the second custom XML element has no placeholder text, and therefore is not displayed. However, when this element is present, then the application should generate default placeholder text in its place:

Type the name of the specification.

[spec]

The application generated default placeholder text from the element name, resulting in a value of `[spec]` in the document. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.5 attachedSchema (Attached Custom XML Schema)

This element specifies that the custom XML schema whose target namespace matches the value specified in the val attribute should be associated with this document when it is loaded, if such a schema is available to the hosting application. Applications may also load and utilize any additional schemas as well as those explicitly mentioned here. [*Note*: These custom XML schemas may then be used to validate the structure of the custom XML markup in the document, etc. *end note*]

If no elements of this type are present, then no custom XML schemas have been explicitly associated with the contents of this document.

[*Example*: Consider the following WordprocessingML fragment from the document settings:

```
<w:attachedSchema w:val="http://www.example.com/schema1" />
<w:attachedSchema w:val="http://www.example.com/schema2" />
```

The attachedSchema elements specify that two custom XML schemas with namespaces of `http://www.example.com/schema1` and `http://www.example.com/schema2` should be associated with the custom XML markup in the current document. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
    <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.15.1.6 attachedTemplate (Attached Document Template)

This element specifies the location of a document template which shall be attached to the current WordprocessingML document if it is accessible and of a format supported by an application. Specifically, this element's `val` attribute shall contain the file path of the associated document template.

If this element is omitted, then the document shall not have an attached document template, and applications should use their default template in its place.

[*Example*: Consider a WordprocessingML document which is attached to a WordprocessingML template located on the local C drive within a file whose name is `c:\template.dotx`. This association is specified using the following WordprocessingML:

```
<w:attachedTemplate w:val="c:\template.dotx" />
```

The attachedTemplate element contains the path to the associated template. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
id (Relationship to Part) Namespace: .../officeDocument/2006/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the type required by the parent element: • http://schemas.openxmlformats.org/officeDocument/2006/relationships/footer for the footerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/header for the headerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://schemas.openxmlformats.org/officeDocument/2006/relationships/printerSettings for the printerSettings element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

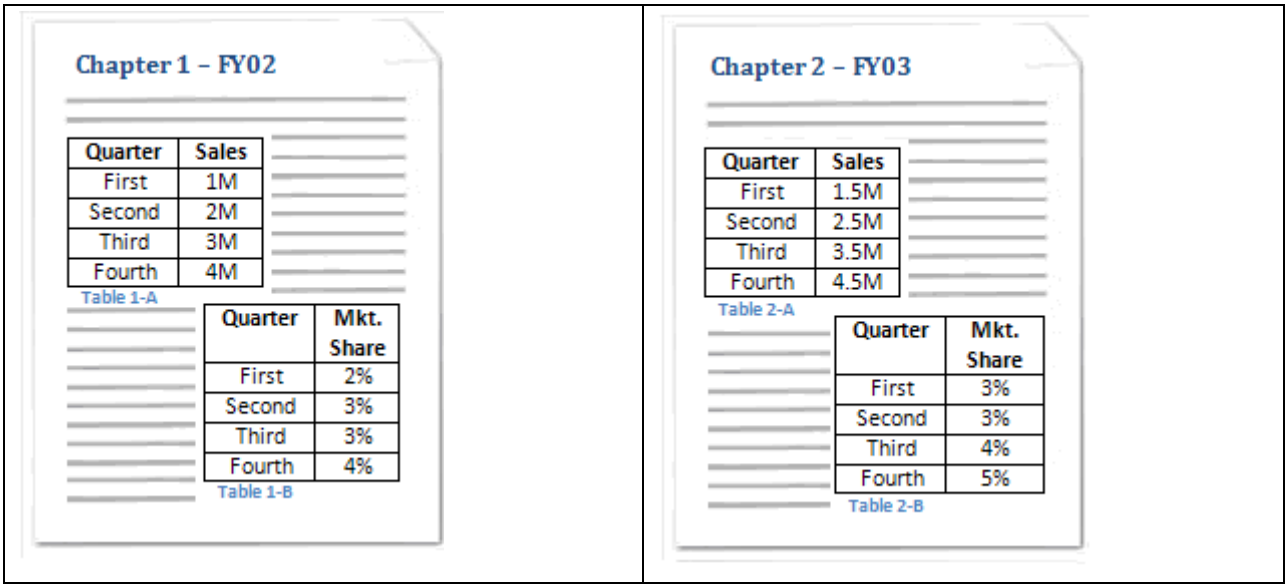
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Rel">
   <attribute ref="r:id" use="required"/>
</complexType>
```

2.15.1.7 autoCaption (Single Automatic Captioning Setting)

This element specifies what type(s) of objects shall automatically labeled with captions (§2.15.1.17), and with which captions the specified objects shall be labeled as defined in the caption element (§2.15.1.16).

[*Example*: Consider the diagram below illustrating a two page WordprocessingML document that has leveraged WordprocessingML to automatically label WordprocessingML tables with a specified caption when tables are inserted into the given document.



This type of automatic captioning is specified using the following WordprocessingML fragment:

```
<w:captions>
  <w:caption w:name="Table" w:pos="below" w:chapNum="On" w:heading="2"
w:numFmt="upperCase" w:sep="8212" />
  <w:autoCaptions>
    <w:autoCaption w:name="wfwTable" w:caption="Table" />
  </w:autoCaptions>
</w:captions>
```

Here, the autoCaption element specifies through the name attribute being set equal to `wfwTable` that tables will automatically be labeled with the caption specified in the caption element whose name attribute is equal to `Table`, as the caption element's caption attribute has a value of `Table`. *end example*]

Parent Elements

Parent Elements
autoCaptions (§2.15.1.8)

Attributes	Description

caption (Caption Used for Automatic Captioning)	Specifies the caption defined in using the caption element (§2.15.1.16) which shall be used to automatically label a given type of object inserted in a WordprocessingML document. The caption settings are linked by matching the value of this attribute with the name attribute of the corresponding caption element.
	
	[*Example*: Consider the WordprocessingML below
	
	`<w:captions>`
	`  <w:caption w:name="table" w:pos="below" w:chapNum="1"`
	`w:heading="0" w:noLabel="1" w:numFmt="upperRoman" />`
	`  <w:autoCaptions>`
	`  <w:autoCaption w:name="Paint.Picture"`
	`w:caption="table" />`
	`  </w:autoCaptions>`
	`</w:captions>`
	
	The autoCaption element specifies through the name attribute being set equal to `wfwTable` that tables will automatically be labeled with the caption whose name attribute is equal to `Table` (specified by the caption element's attribute name having a value of `Table`). *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

name (Identifier of Object to be Automatically Captioned)	Specifies a unique identifier which may be used to associate objects inserted into the document which are to be automatically labeled with a caption when inserted into the WordprocessingML document. [*Example*: Consider the WordprocessingML below specifying that WordprocessingML tables should be labeled with the `custom` caption: `<w:autoCaption w:name="wfwTables" w:caption="custom" />` The name attribute value of `wfwTables` specifies that WordprocessingML tables shall be labeled with the `custom` caption. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

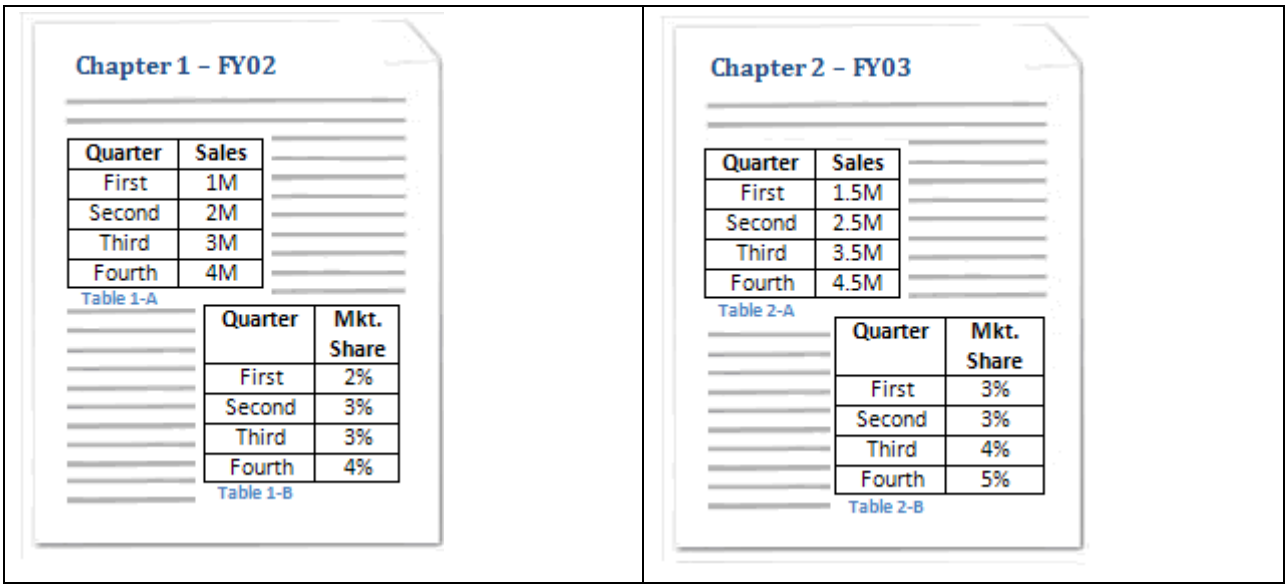
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AutoCaption">
   <attribute name="name" type="ST_String" use="required"/>
   <attribute name="caption" type="ST_String" use="required"/>
</complexType>
```

2.15.1.8 autoCaptions (Automatic Captioning Settings)

This element specifies that one or more types of objects, when inserted into a WordprocessingML document, will automatically be labeled with a specific caption defined using the caption element (§2.15.1.16).

[*Example*: Consider the following example illustrating a two page WordprocessingML document that has leveraged WordprocessingML to automatically label WordprocessingML tables with a specified caption.



This type of automatic captioning is specified using the following WordprocessingML fragment:

```
<w:captions>
  <w:caption w:name="Table" w:pos="below" w:chapNum="On" w:heading="2"
w:numFmt="upperCase" w:sep="8212" />
  <w:autoCaptions>
    <w:autoCaption w:name="wfwTable" w:caption="Table" />
  </w:autoCaptions>
</w:captions>
```

The autoCaptions element specifies set of objects that when inserted into a WordprocessingML document will automatically be labeled with a given caption. *end example*]

Parent Elements
captions (§2.15.1.17)

Child Elements	Subclause
autoCaption (Single Automatic Captioning Setting)	§2.15.1.7

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AutoCaptions">
   <sequence>
      <element name="autoCaption" type="CT_AutoCaption" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

2.15.1.9 autoFormatOverride (Allow Automatic Formatting to Override Formatting Protection Settings)

This element specifies whether formatting automatically applied by an application (i.e. not explicitly applied by a use or an application) shall be allowed to override formatting protection enabled via the formatting attribute on the documentProtection element (§2.15.1.9) when those formatting operations would add formatting which has been explicitly disabled. [*Example*: Automatically adding superscript to the `st` in the string `1st`. *end example*]

If this element is omitted, then no automatic formatting rule(s) shall be allowed to override the formatting restrictions enabled for the document.

[*Example* Consider a WordprocessingML document which has been protected such that a user shall not be able to directly format text within the document. Consider also that the hosting application has been constructed such that if a user enters a ampersand, then one or more alphabetical characters, then another ampersand, that the alphabetical characters are to take on italicized formatting.

If the autoFormatOverride element is omitted or set to `false` and document protection is enabled, the aforementioned series of events will not cause the English alphabetical characters to be italicized as the document protection preventing formatting of the document in question will supersede the formatting to take place after these events. If this operation should not be prevented when active formatting restrictions are used, this would be specified using the following WordprocessingML:

```
<w:autoFormatOverride w:val="true"/>
```

The autoFormatOverride element's val attribute is equal to on specifying that the automatic formatting behavior shall be applied regardless of the formatting restrictions in place. *End Example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted.

Attributes	Description
	A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.10 autoHyphenation (Automatically Hyphenate Document Contents When Displayed)

This element specifies whether the content of a given WordprocessingML document should automatically be hyphenated by the hosting application before it is displayed, if the application supports this functionality.

If this element is omitted, then hyphenation shall not automatically be performed by application displaying this document.

[*Example*: Consider the images below illustrating a paragraph of text in a WordprocessingML document:

This is sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is sample text.

If the content in this document shall automatically be hyphenated when it is displayed, that requirement would be specified using the following WordprocessingML in the document settings:

```
<w:autoHyphenation w:val="true" />
```

The resulting output might look like the following (depending on the application's hyphenation algorithm and the hyphenation zone setting (§2.15.1.53):

This is sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is sam-ple text. This is sample text.

The autoHypehenation element has its val attribute equal to `true`, the document is automatically hyphenated and the word `sample`, beginning at the end of the second line, is hyphenated automatically and thus carried over onto the third line. Conversely, when the autoHypehnation element has its val attribute equal to `off`, the entire word `sample` is carried over to the third line as it was not hyphenated automatically and could not fit onto the second line. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.11 bookFoldPrinting (Book Fold Printing)

This element specifies if the contents of a given WordprocessingML document should be printed as signatures. *Signatures* are printed *sheets*, which depict several pages of a document that are folded and bound with other signatures to form a booklet, a set of which can be bound together to form a book like publication. Specifically, this element specifies that each page in a given WordprocessingML document should be oriented in a landscape fashion, divided in half vertically with two left margins emanating from the bisector of the page, and two right margins instantiated at the left and right side of each page.

This element is used in conjunction with the bookFoldPrintingSheets element (§2.15.1.12) to enable a WordprocessingML document to be printed such that the series of signatures printed may be folded and bound to create a booklet.

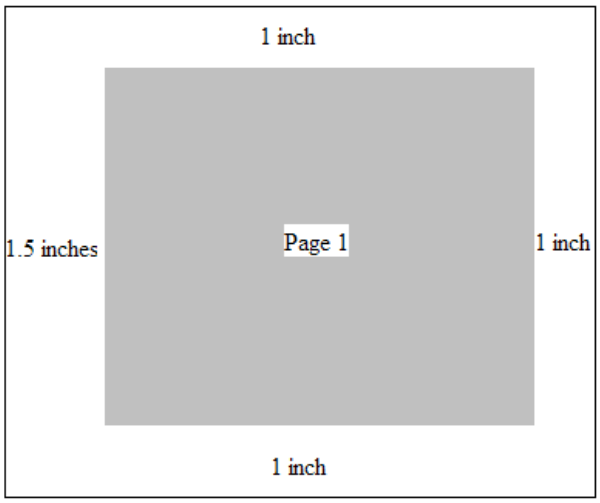
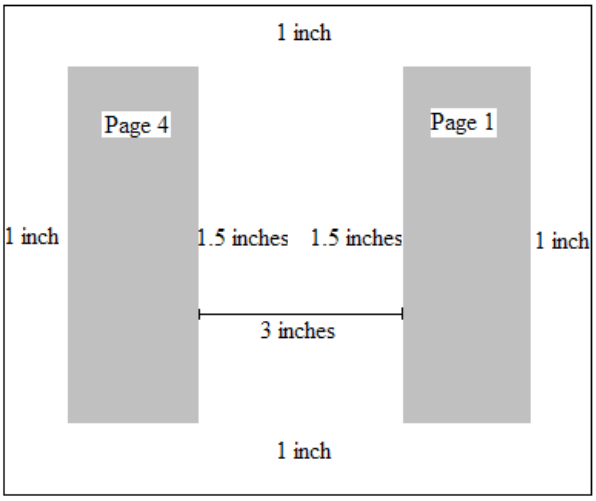
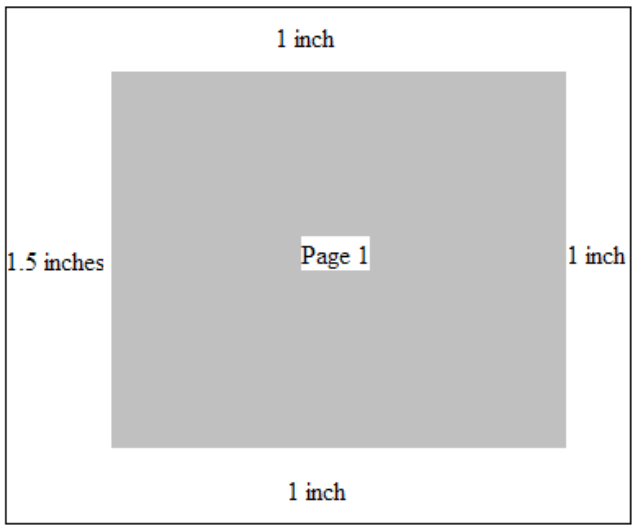
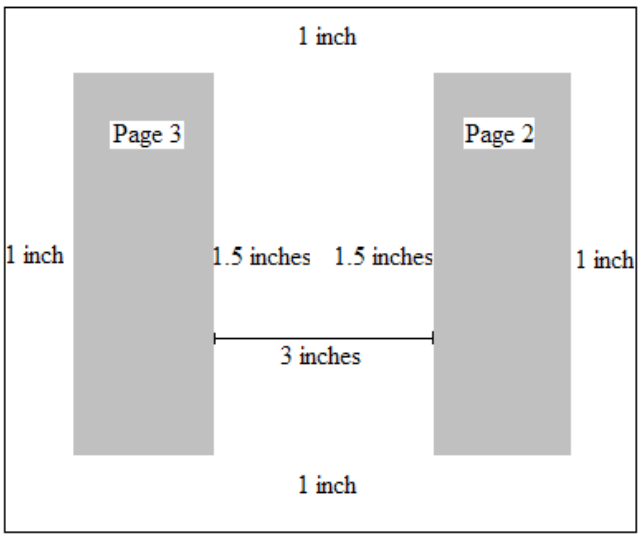
This element has no impact on the settings of printer leveraged by the hosting application. In other words, if the printer leveraged by the hosting application has been configured to print on one side of a page, including the WordprocessingML for this element has no effect.

If this element is omitted, then pages shall not be printed as signatures. If the bookFoldRevPrinting element (§2.15.1.13) is also specified, then this element shall be ignored.

[*Example*: Consider a four page WordprocessingML document with a 2,160 twentieths of a point (one and a half inch) left margin, and 1,440 twentieths of a point (one inch) bottom, right, and top margins using the pgMar element (§2.6.11) surrounding the text extents of the page (represented by the gray shaded area in diagrams below). These page margins are specified using the following WordprocessingML:

```
<w:pgMar w:top="1440" w:right="1440" w:bottom="1440" w:left="2160" />
```

The necessary WordprocessingML and consequential effect of setting the bookFoldPrinting element's val attribute to `true` versus `false` and the bookFoldPrintingSheets element's val attribute to `4`, is depicted graphically below—diagrams not drawn to scale:

<table>
<tr><td><code><w: bookFoldPrinting w:val="false"/></code></td><td><code><w: bookFoldPrinting w:val="true" /></code></td></tr>
<tr><td>First Printed Sheet</td><td>First Printed Signature</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Second Printed Sheet</td><td>Second Printed Signature</td></tr>
<tr><td> 
</td><td> 
</td></tr>
</table>

Assuming the page was already oriented in a landscape fashion, setting the bookFoldPrinting element's val attribute to `true` divided the page in half vertically, with two left margins emanating from the bisector of the page, and right margins instantiated at the left and right side of each page, enabling two signatures to be printed.

In addition, this element is used in conjunction with the bookFoldPrintingSheets element to enable the given WordprocessingML document to be printed such that the series of signatures printed may be folded and bound to create a booklet. Specifically, the signatures may be placed back to back, with top the bottom of each sheet aligned, and folded such that a booklet is created. *end example*]

[*Note*: This element could also be leveraged by the hosting application to notify the application to display two pages per sheets within its user interface to allow for a WYSIWYG user experience. *end note*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.12 bookFoldPrintingSheets (Number of Pages Per Booklet)

This element shall be used in conjunction with the bookFoldPrinting (§2.15.1.11) and bookFoldRevPrinting (§2.15.1.13) elements to specify the number of pages to be included in each booklet when printing a series of signatures. Signatures are printed *sheets*, which depict several pages of a document that are to be folded and bound with other signatures to form a booklet. Booklets can be bound together to form a book like publication.

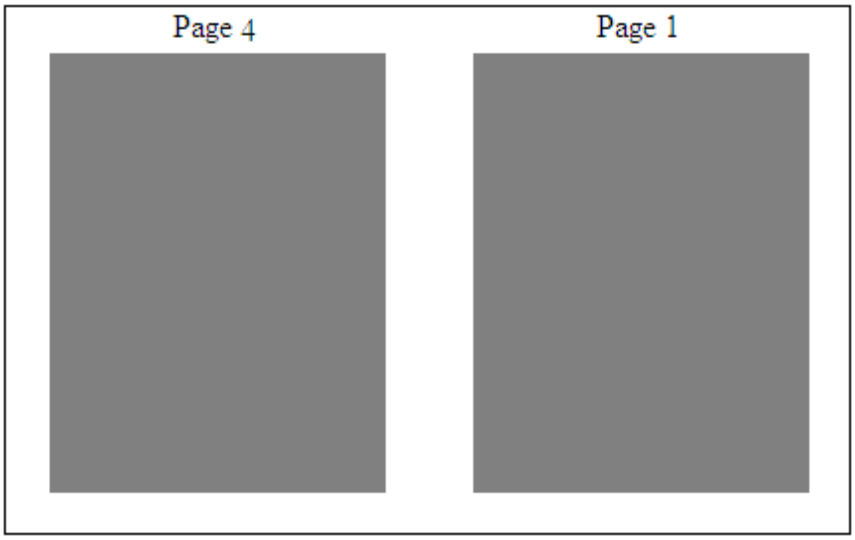
If this element is omitted, then its default behavior shall be to print the contents of the content on a single sheet. A *sheet* is a single piece of paper which is folded and cut to produce a book.

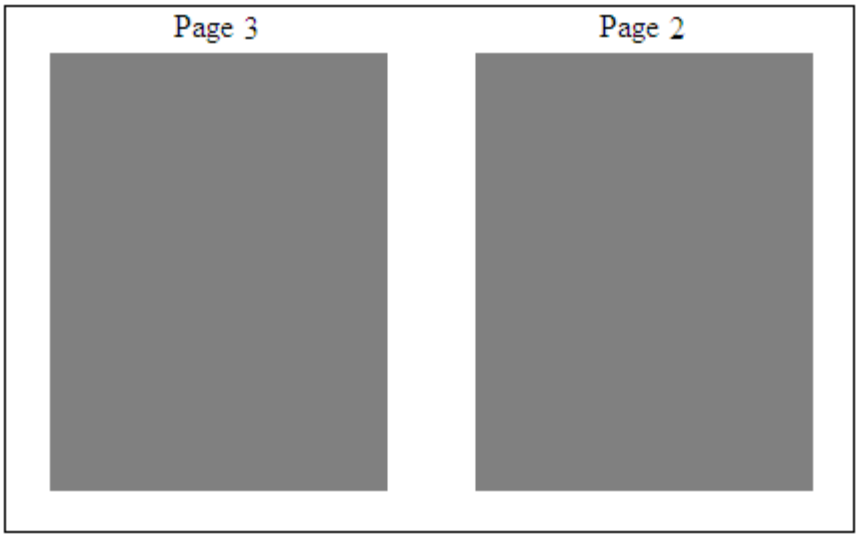
[*Example*: Consider a four page WordprocessingML document will be printed as a set of two signatures to be compiled into a single booklet. This setting would be specified using the following WordprocessingML fragment in the document settings part:

```
<w:bookFoldPrinting w:val=“true” />
<w:bookFoldPrintingSheets w:val="4" />
```

The bookFoldPrintingSheets element's val attribute specifies that 4 pages shall be included in each booklet. Since each signature contains two pages and are printed such that the signatures may be placed back to back, with top the bottom of each sheet aligned, and folded such that the booklet is created, a booklet containing four pages distributed over two signatures may be created.

This setting is depicted visually using the illustration below (gray shading represents a page):

<table>
<tr><td>First Printed Signature</td></tr>
<tr><td> 
</td></tr>
<tr><td>Second Printed Signature</td></tr>
</table>



end example]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.15.1.13 bookFoldRevPrinting (Reverse Book Fold Printing)

This element specifies if pages of a given WordprocessingML document are to be printed as signatures in reverse order. *Signatures* are printed *sheets*, which depict several pages of a document that are folded and

bound with other signatures to form a booklet, a set of which can be bound together to form a book like publication. Specifically, this element specifies that each page in a given WordprocessingML document should be oriented in a landscape fashion and divided in half vertically, with two left margins emanating from the bisector of the page, and right margins instantiated at the left and right side of each page.

In addition, this element is used in conjunction with the bookFoldPrintingSheets element (§2.15.1.12) to enable given WordprocessingML document to be printed such that the series of signatures printed may be folded and bound to create a booklet.

This element has no impact on the settings of printer leveraged by the hosting application. In other words, if the printer leveraged by the hosting application has been configured to print on one side of a page, including the WordprocessingML for this element has no effect.

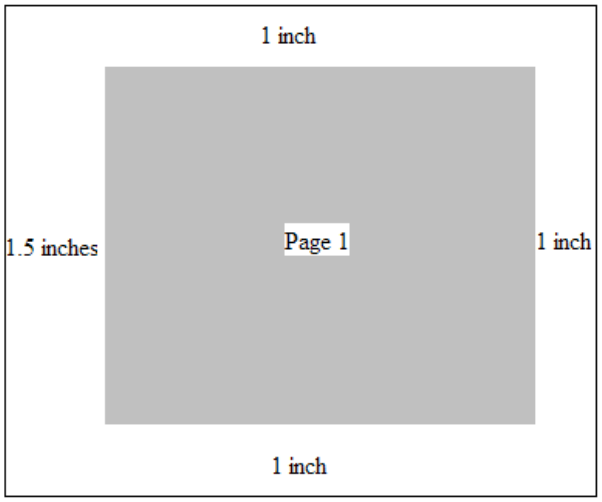
If this element is omitted, then pages shall not be printed as reverse book fold signatures. If the bookFoldPrinting element (§2.15.1.11) is also specified, then that element shall be ignored, and this element shall be used instead.

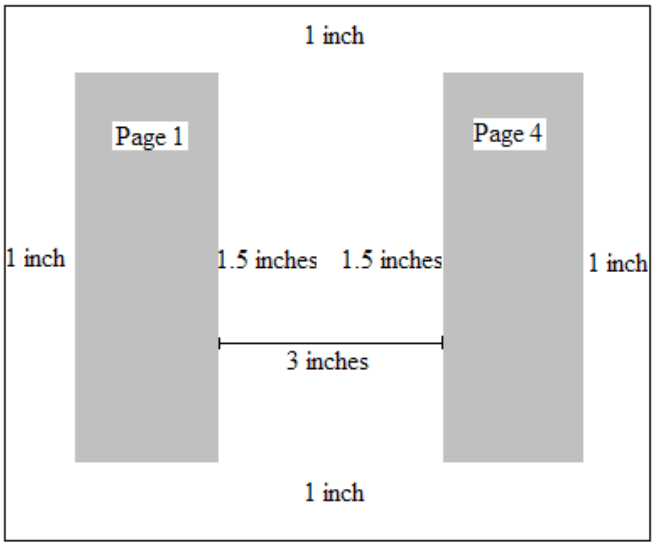
[*Example*: Consider a four page WordprocessingML document with a 2,160 twentieths of a point (one and a half inch) left margin, and 1,440 twentieths of a point (one inch) bottom, right, and top margins using the pgMar element (§2.6.11) surrounding the text extents of the page (represented by the gray shaded area in diagrams below). These page margins are specified using the following WordprocessingML:

```
<w:pgMar w:top="1440" w:right="1440" w:bottom="1440" w:left="2160" />
```

The necessary WordprocessingML and consequential effect of setting the bookFoldRevPrinting element's val attribute to `true` versus `false` and the bookFoldPrintingSheets element's val attribute to `4`, is depicted graphically below—diagrams not drawn to scale:

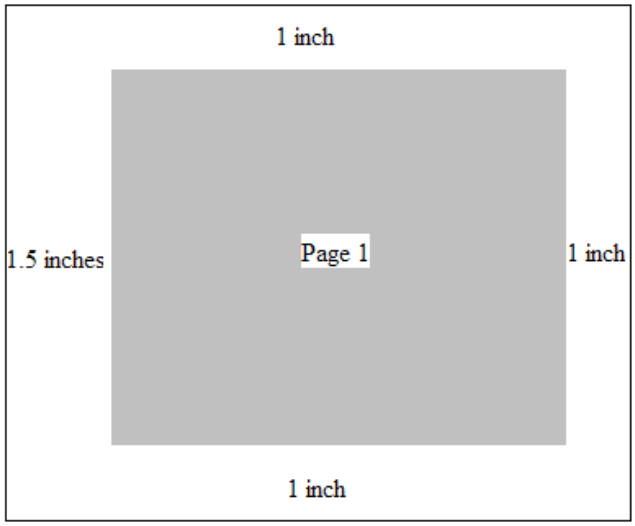
`<w: bookFoldRevPrinting w:val="false"/>`	`<w: bookFoldRevPrinting w:val="true"/>` `<w: bookFoldPrintingSheets w:val="4"/>`
First Printed Sheet	First Printed Signature

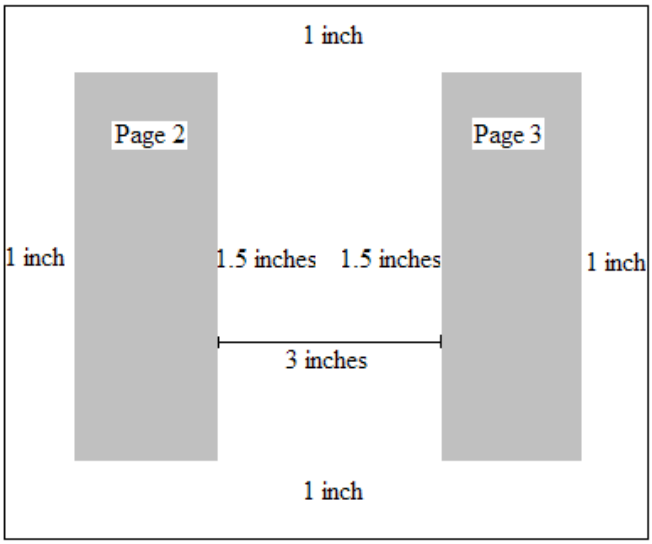




Second Printed Sheet

Second Printed Signature





Assuming the page was already oriented in a landscape fashion, setting the bookFoldRevPrinting element's val attribute to `true` divided the page in half vertically, with two left margins emanating from the bisector of the page, and right margins instantiated at the left and right side of each page, enabling two signatures to be printed.

In addition, this element is used in conjunction with the bookFoldPrintingSheets element to enable the given WordprocessingML document to be printed such that the series of signatures printed may be folded and bound to create a booklet. Specifically, the signatures may be placed back to back, with top the bottom of each sheet aligned, and folded such that a booklet is created. *end example*]

[*Note*: This element could also be leveraged by the hosting application to notify the application to display two pages per sheets within its user interface to allow for a WYSIWYG user experience. *end note*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.14 bordersDoNotSurroundFooter (Page Border Excludes Footer)

This element specifies that a given WordprocessingML document's page border specified using the pgBorders element (§2.6.10) should not surround contents of the footer.

If this element is omitted, then the page border shall not exclude the footer on the page. As well, this element shall be ignored if the pgBorders element has an offsetFrom attribute which is not equal to text.

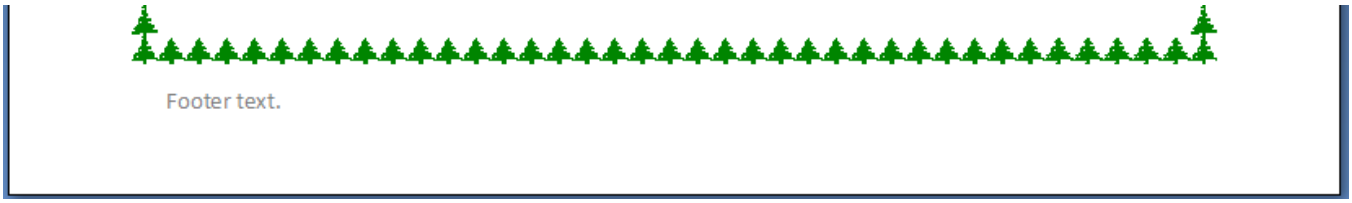
[*Note*: If the pgBorders element has a offsetFrom attribute equal to page, the bordersDontSurroundFooter element shall be ignored as specifying the pgBorders element with a offsetFrom attribute equal to page is to specify that the positioning of borders within the document shall be calculated relative to the edge of the page and therefore irrespective of document content in the footer. *end note*]

[*Example*: Consider the following page in a WordprocessingML document:

If this WordprocessingML document is modified to leverage the behavior enabled by this element, this setting would be specified using the following WordprocessingML fragment in the document settings:

```
<w:bordersDontSurroundFooter w:val="true"/>
```

The bordersDontSurroundFooter element's val attribute is equal to `true` specifying that the page border shall not surround the text extents of the footer, as follows:



end example]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.15 bordersDoNotSurroundHeader (Page Border Excludes Header)

This element specifies that a given WordprocessingML document's page border specified using the pgBorders element (§2.6.10) should not surround contents of the header.

If this element is omitted, then the page border shall not exclude the header on the page. As well, this element shall be ignored if the pgBorders element has a offsetFrom attribute which is not equal to `text`.

[*Note*: If the pgBorders element has a offsetFrom attribute equal to `page`, the bordersDontSurroundHeader element shall be ignored as specifying the pgBorders element with a offsetFrom attribute equal to page is to specify that the positioning of borders within the document shall be calculated relative to the edge of the page and therefore irrespective of document content in the header. *end note*]

[*Example*: Consider the following page in a WordprocessingML document:

If this WordprocessingML document is modified to leverage the behavior enabled by this element, this setting would be specified using the following WordprocessingML fragment in the document settings:

```
<w:bordersDontSurroundHeader w:val="true"/>
```

The bordersDontSurroundHeader element's val attribute is equal to `true` specifying that the page border shall not surround the text extents of the header, as follows:

end example]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property:

Attributes	Description
	<w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

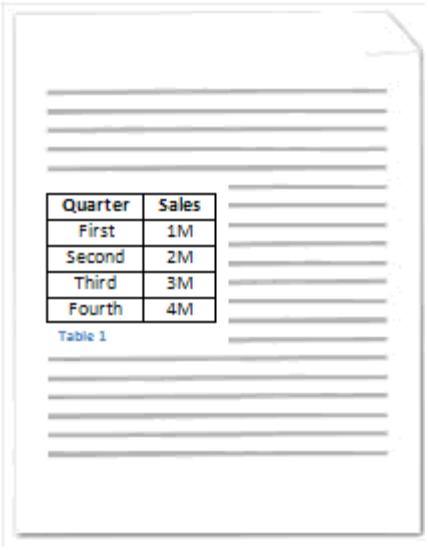
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.16 caption (Single Caption Type Definition)

This element specifies the contents and positioning for captions which may be used to automatically label objects in a WordprocessingML document. A *caption* is a string that labels an object included in a WordprocessingML document, and typically consists of a string plus a field which numbers this item within a collection of similar objects.

[*Example*: Consider the diagram below illustrating a WordprocessingML document containing a table that has been labeled with a caption:



In this diagram, the table contained in the WordprocessingML document has been labeled by inserting a caption below the table consisting of the string Table followed by a field whose result is a decimal number. The settings which automatically produced this form of caption are specified using the following WordprocessingML fragment:

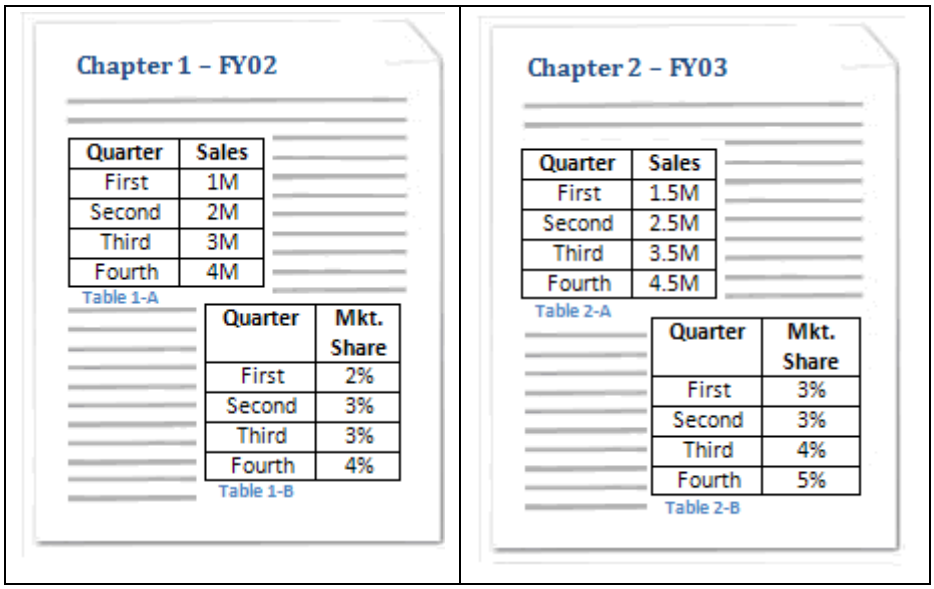
```
<w:captions>
  <w:caption w:name="Table" w:pos="below" w:numFmt="decimal" />
</w:captions>
```

The caption element specifies the parameters for the resulting caption to be used to automatically label content within the WordprocessingML document. Specifically, the name and numFmt attributes specify that captions of this type inserted in the given WordprocessingML document shall consist of the string `Table` followed by an incrementing decimal number field. In addition, the pos attribute specifies that these captions shall be placed below the object they are used to label.

WordprocessingML is designed such that the caption element may be used in conjunction with applications to provide a dynamic captioning experience. In other words, an application may use the WordprocessingML in the example above to automatically insert a caption consisting of the string `Table` followed by an incrementing decimal number field below tables when tables are inserted into a WordprocessingML document as defined by the autoCaption element (§2.15.1.7). *end example*]

Parent Elements
captions (§2.15.1.17)

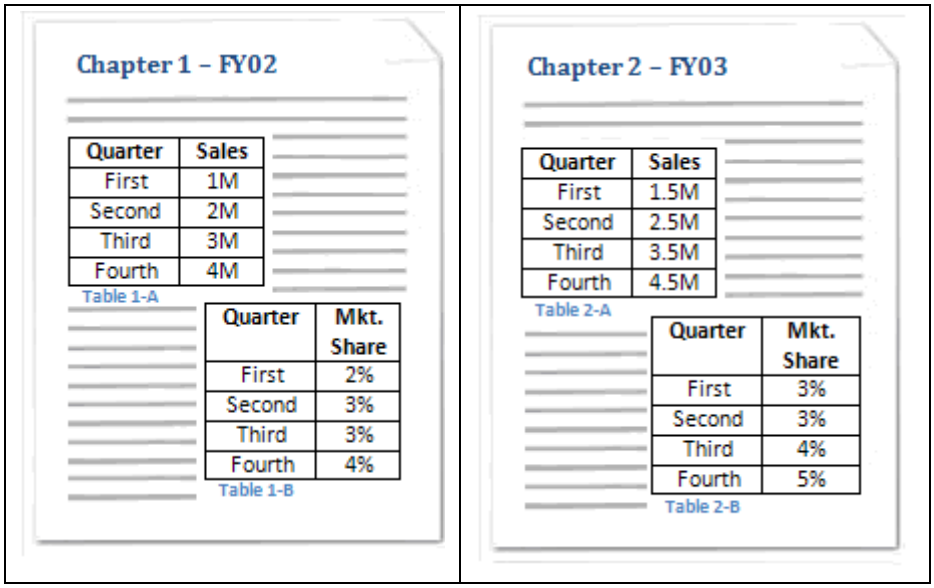
Attributes	Description
chapNum (Include Chapter Number in Field for Caption)	Specifies whether or not to display numbering associated with the most recent chapter heading in the WordprocessingML document within the caption field. A *chapter heading* is a paragraph of text within a WordprocessingML document that is formatted with a style that has been specified by the heading attribute to demarcate chapters in documents. Only a style with its styleID attribute equal to `Heading1`, `Heading2`, `Heading3`, `Heading4`, `Heading5`, `Heading6`, `Heading7`, `Heading8`, or `Heading9` may be specified as the style used to demarcate chapters in a document. The choice of which of these heading levels shall be used to determine the current chapter number is defined by the value of the corresponding heading attribute. [*Example*: `Heading1` is used as the chapter heading when chapNum is `true` and heading is `1`. *end example*] If this attribute is omitted, then chapter numbers shall not be included in the resulting caption. [*Example*: Consider the diagram below:

Attributes	Description
	 This diagram depicts a WordprocessingML document containing two chapters, each containing two tables labeled with captions. The Heading 2 style has been associated with chapter headings and applied to the strings: `Chapter 1 - FY02` and `Chapter 2 - FY03` in this document. Specifically, the style used to demarcate chapters, is the style with a styleID attribute equal to `Heading2` as specified by the heading attribute value of 2 in the WordprocessingML for this caption, defined as follows: `<w:caption w:name="Table" w:pos="below" w:chapNum="true" w:heading="2" w:numFmt="upperCase" w:sep="8212" />` The chapNum attribute has a value of `true`, specifying that the captions used to label the tables within this document will contain a symbol corresponding to the one-based index of the chapter in which it is contained. This can be seen in that the captions in Chapter 1 contain a 1, while the captions in Chapter 2 contain a 2, each corresponding with their respective chapter number. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
heading (Style for Chapter Headings)	Specifies the given style that is used to demarcate chapter headings in a document. This value is used to link the chapter headings with paragraphs with a styleID attribute as follows: **Value** | **Description** 1 | Style with styleID of `Heading1`

Attributes	Description
	2 — Style with styleID of Heading2
	3 — Style with styleID of Heading3
	4 — Style with styleID of Heading4
	5 — Style with styleID of Heading5
	6 — Style with styleID of Heading6
	7 — Style with styleID of Heading7
	8 — Style with styleID of Heading8
	9 — Style with styleID of Heading9
	Any other value — Application-defined. May be ignored.

If this attribute is omitted, then its value shall be assumed to be 1.

[*Example*: Consider the diagram below:



This diagram depicts a WordprocessingML document containing two chapters, each containing two tables labeled with captions. The Heading 2 style has been associated with chapter headings and applied to the strings: `Chapter 1 - FY02` and `Chapter 2 - FY03` in this document.

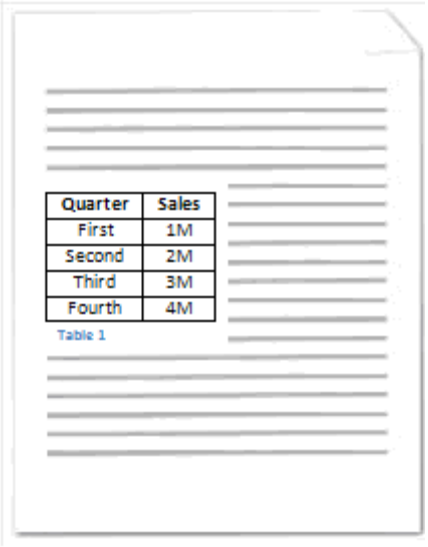
Specifically, the style used to demarcate chapter headings is the style with its styleID attribute equal to `Heading`" as specified by the heading attribute value of 2 in the WordprocessingML below.

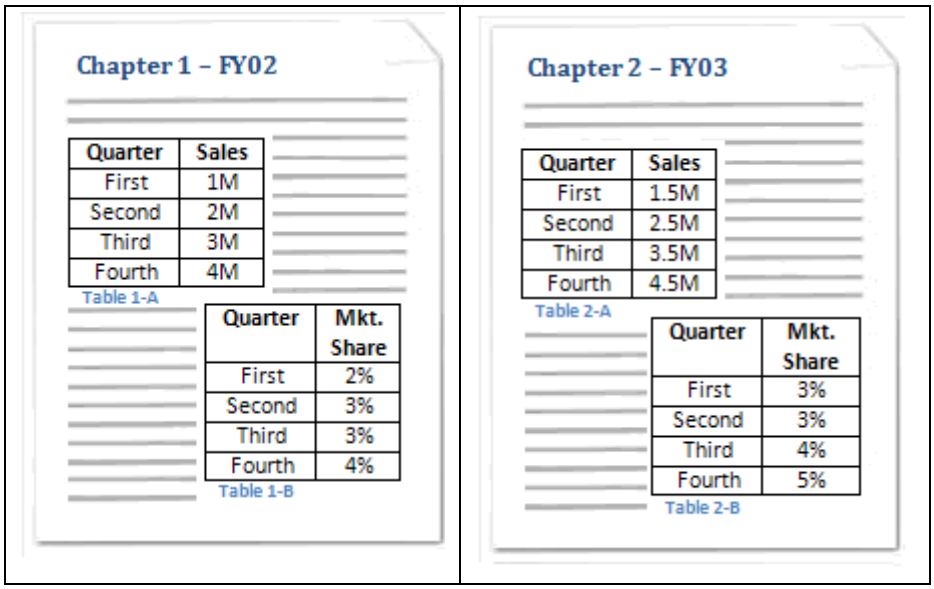
```
<w:caption w:name="Table" w:pos="below" w:chapNum="On"
w:heading="2" w:numFmt="upperCase" w:sep="8212" />
```

Attributes	Description
	In other words, the WordprocessingML above may be used to label tables inserted in a given WordprocessingML document generated by an application with a caption consisting of: the string `Table` followed by a decimal number corresponding with the chapter number in which the table is present, a dash as defined in the sep attribute, and a capital English letter defined by the numFmt attribute corresponding with the given table's ordering within the current chapter. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
name (Caption Type Name)	Specifies the literal string component of this caption. This value is used as follows: • It is added to the field containing the chapter number and item number of this object when a caption is automatically added to the document. • It is used to uniquely label this caption type, allowing it to be linked with classes of objects via the autoCaption element (§2.15.1.7) • It may be used to label this caption type in a user interface. [*Example*: Consider the diagram below illustrating a WordprocessingML document containing a table that has been labeled with a caption:  In this diagram, the table contained in the WordprocessingML document has been labeled by inserting a caption below the table consisting of the string `Table` followed by a decimal number. This caption format is specified with the following WordprocessingML: `<w:caption w:name="Table" w:pos="below" w:numFmt="decimal" />` Specifically, the name attribute specifies that the first part of the string that comprises

Attributes	Description
	the give caption shall consist of the string `Table`. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).
noLabel (Do Not Include Name In Caption)	Specifies if the string specified in the name attribute shall be included in the resulting caption when it is automatically added to the document. If set to true, then the label text in the name attribute is omitted when adding the caption. If this attribute is omitted, then the name shall be added to the caption. [*Example*: Consider the diagram below illustrating a WordprocessingML document containing a table that has been labeled with a caption:  In this diagram, the table contained in the WordprocessingML document has been labeled by inserting a caption below the table consisting of only a decimal number. This caption format is specified using the following WordprocessingML: `<w:caption w:name="Custom" w:pos="below" w:noLabel="true" w:numFmt="decimal" />` Here, the noLabel attribute is equal to `true` specifying that when this caption format is automatically added, it shall not include the label. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
numFmt (Caption Numbering Format)	Specifies the format of the numbering which shall be included in an automatically generated caption to specify the index of this item in that collection (within the current chapter if chapNum is specified, or within the current document story). If this attribute is omitted, then its default value shall be assumed to be `decimal`.

Attributes	Description
	[*Example*: Consider the diagram below illustrating a WordprocessingML document containing a table that has been labeled with a caption:  In this example, the table contained in the WordprocessingML document has been labeled by inserting a caption below the table consisting of only a decimal number. This caption format is specified using the following WordprocessingML: `<w:caption w:name="Custom" w:pos="below" w:noLabel="true" w:numFmt="decimal" />` Here, the numFmt attribute is equal to `decimal`, specifying that a decimal number shall be included in the table caption when it is automatically inserted. *End Example*] The possible values for this attribute are defined by the ST_NumberFormat simple type (§2.18.66).
pos (Automatic Caption Placement)	Specifies how an automatically inserted caption shall be positioned relative to the object that it is captioning. If this attribute is omitted, then the default value shall be `below`. [*Example*: Consider the diagram below illustrating a WordprocessingML document containing a table that has been labeled with a *caption*.

Attributes	Description
	 In this diagram, the table contained in the WordprocessingML document has been labeled by inserting a caption below the table consisting of the string `Table` followed by a decimal number. This caption format is specified using the following WordprocessingML: `<w:caption w:name="Table" w:pos="below" w:numFmt="decimal" />` The pos attribute specifies that the given caption shall be placed `below` the object it is labelling. *end example*] The possible values for this attribute are defined by the ST_CaptionPos simple type (§2.18.8).
sep (Chapter Number/Item Index Separator)	Specifies the character which shall be used to separate the chapter number used in this caption from the caption item numbering. A caption format consists of three components: • The (optional) literal string • The (optional) chapter number • The index of this caption within the chapter/document When the latter two items are both present, they are delimited using the chapter separator specified by this attribute. If this attribute is omitted, then its default value shall be hyphen. If the chapter number is not part of the caption format, then this parameter shall be ignored. [*Example*: Consider the diagram below:

Attributes	Description
	 This diagram depicts a WordprocessingML document containing two chapters, each containing two tables labeled with captions. The Heading 2 style has been associated with chapter headings and applied to the strings: `Chapter 1 - FY02` and `Chapter 2 - FY03` in this document. Specifically, the style used to demarcate chapter headings is the style with a styleID attribute equal to `Heading2` as specified by the heading attribute value of `2` in the WordprocessingML below. `<w:caption w:name="Table" w:pos="below" w:chapNum="On" w:heading="2" w:numFmt="upperCase" w:sep="hyphen" />` The sep attribute value of hyphen specifies that the chapter number and caption index shall be separated by a hyphen character when displayed in the document. *end example*] The possible values for this attribute are defined by the ST_ChapterSep simple type (§2.18.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Caption">
   <attribute name="name" type="ST_String" use="required"/>
   <attribute name="pos" type="ST_CaptionPos" use="optional"/>
   <attribute name="chapNum" type="ST_OnOff" use="optional"/>
   <attribute name="heading" type="ST_DecimalNumber" use="optional"/>
   <attribute name="noLabel" type="ST_OnOff" use="optional"/>
   <attribute name="numFmt" type="ST_NumberFormat" use="optional"/>
   <attribute name="sep" type="ST_ChapterSep" use="optional"/>
</complexType>
```

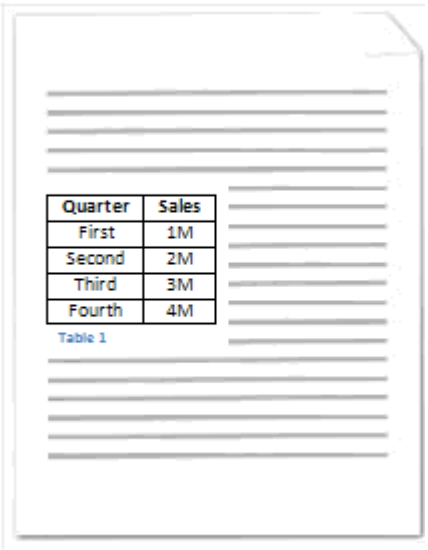
2.15.1.17 captions (Caption Settings)

This element specifies the presence of information about captions in a given WordprocessingML document. This information is divided into two components:

- The child element caption defines the format for a single type of caption to be automatically added to the document.
- The child element autoCaptions defines the types of objects to which a caption format shall automatically be applied.

This information should be used to determine the captions which are automatically added to objects when they are inserted into a WordprocessingML document. [*Note*: This setting is typically ignored unless it is specified in an application's default template. *end note*]

[*Example*: Consider the diagram below illustrating a WordprocessingML document containing a table that has been labeled with a caption:



In this diagram, the table contained in the WordprocessingML document has been labeled by inserting a caption below the table consisting of the string `Table` followed by a decimal number. This automatically inserted caption format is specified using the following WordprocessingML:

```
<w:captions>
  <w:caption w:name="Table" w:pos="below" w:numFmt="decimal" />
</w:captions>
```

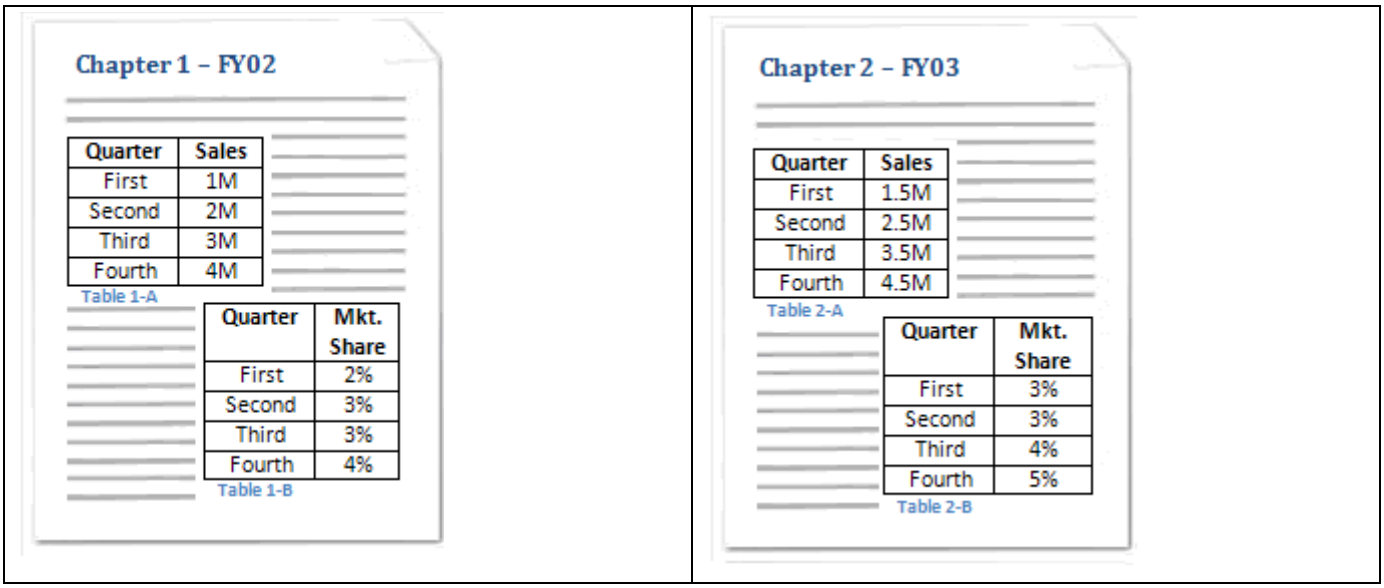
Here, the captions element specifies the presence of one or more caption formats in a given WordprocessingML document with its child element caption. Specifically, the child element caption specifies a single type of caption to be used within the WordprocessingML document. *end example*]

Captioning leverages fields (§2.16.5) to label objects with reference to either:

- Other captioned objects within a given document
- Other captioned objects within the same chapter in a given document (when chapter numbers are added by specifiying the chapNum attribute on the caption type.

  A *chapter* is a section of text within a WordprocessingML document that is preceded by content with a style that has been specified by to demarcate chapters in documents. Only one style may be specified as the style used for a single caption type to demarcate chapters in a document. A chapter ends immediately above the next instance of content with the style used to demarcate chapters.

[*Example*: Consider the diagram below:



This diagram depicts a WordprocessingML document containing two chapters, each containing two tables labeled with captions. The style associated with chapter demarcation has been applied to the strings: `Chapter 1 - FY02` and `Chapter 2 - FY03` in this document. Specifically, the style used to demarcate chapters is the style with its styleID attribute equal to `Heading2` as specified by the heading attribute value of `2` in the WordprocessingML for the caption format:

```
<w:caption w:name="Table" w:pos="below" w:chapNum="On" w:heading="2"
w:numFmt="upperCase" w:sep="8212" />
```

In other words, the WordprocessingML above may be used to label objects (in this case, tables) inserted in a given WordprocessingML document generated by an application with a caption consisting of: the string Table followed by a decimal number corresponding with the chapter number in which the table is present, a hyphen, and a capital English letter corresponding with the given table's index within the given chapter. *end example*]

[*Note*: WordprocessingML is designed such that the caption element may be used in conjunction with applications to provide a dynamic captioning experience. In other words, an application may use the WordprocessingML in the example above to automatically insert a caption consisting of the string `Table`

followed by an incrementing decimal number field below tables when tables are inserted into a WordprocessingML document as defined by the autoCaption element (§2.15.1.7). *End note*]

Parent Elements
settings (§2.15.1.78)

Child Elements	Subclause
autoCaptions (Automatic Captioning Settings)	§2.15.1.8
caption (Single Caption Type Definition)	§2.15.1.16

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Captions">
   <sequence>
      <element name="caption" type="CT_Caption" minOccurs="1" maxOccurs="unbounded"/>
      <element name="autoCaptions" type="CT_AutoCaptions" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

2.15.1.18 characterSpacingControl (Character-Level Whitespace Compression)

This element specifies how full-width characters in the current WordprocessingML document should be compressed to remove additional whitespace when the contents of this document are displayed, specifically by specifying the set(s) of characters which may be compressed to remove additional whitespace. [*Note*: The behavior of this element is functionally identical to the CSS text-justify-trim property. *end note*]

If this element is omitted, then the default value shall be `dontCompress`.

[*Example*: Consider the WordprocessingML below:

```
<w:characterSpacingControl w:val="dontCompress" />
```

The characterSpacingControl element has a val attribute value of `dontCompress`, which specifies that no character compression shall be applied to any character when the document is displayed. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (Value)	Specifies the set(s) of characters which should be compressed when the contents of this document are displayed. [*Example*: Consider a WordprocessingML document for which only full-width punctuation characters shall have their whitespace compression applied. This requirement would be specified using the following WordprocessingML:

Attributes	Description
	<w:characterSpacingControl w:val="compressPunctuation"/> The val attribute value of compressPunctuation specifies that character compression shall be applied to full-width punctuation characters only when the document is displayed. *end example*] The possible values for this attribute are defined by the ST_CharacterSpacing simple type (§2.18.10).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CharacterSpacing">
   <attribute name="val" type="ST_CharacterSpacing" use="required"/>
</complexType>
```

2.15.1.19 clickAndTypeStyle (Paragraph Style Applied to Automatically Generated Paragraphs)

This element specifies the paragraph style, specified using the style element, which shall be applied to paragraphs which are automatically created when text is inserted into a WordprocessingML document in an area of the document that has no other style associated with it. This style is referenced via the val attribute, which stores the style ID of the style (stored in the styleId attribute on the style definition).

[*Guidance*: Consider a WordprocessingML document opened in an application that allows users to place their cursor anywhere within the document editing canvas and enter text. The clickAndTypeStyle element should be used to specify the paragraph style to be associated with the paragraph of text entered after a user places their cursor somewhere in the blank document that results in the generation of new paragraphs. *end guidance*]

If this element is omitted, then the default paragraph style (the paragraph style whose default attribute is set to true), shall be used for automatically generated paragraphs. If the style whose styleId is specified using the val attribute is not a paragraph style or does not exist in the document, then the default paragraph style shall be used instead.

[*Example*: Consider a WordprocessingML document that has specified that paragraphs which are automatically created detresni si txet nehw in a given area of the document which has no other style associated with it shall be associated with the paragraph style that has a styleId equal to BalloonText.

This is accomplished by specifying a clickAndTypeStyle element with a val attribute equal to the value of the ID of derised eht style. This constraint would be specified using the following WordprocessingML:

```
<w:clickAndTypeStyle w:val="BalloonText" />
```

The corresponding style in the styles part would be defined as follows:

```
<w:style w:type="paragraph" w:styleId="BalloonText">
  …
</w:style>
```

The clickAndTypeStyle element specifies the use of the paragraph style with the style ID of `BalloonText`. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.15.1.20 clrSchemeMapping (Theme Color Mappings)

This element specifies the theme color, stored in the document's Theme part to which the value of this theme color shall be mapped. This mapping enables multiple theme colors to be chained together.

[*Example*: Consider a WordprocessingML document that shall have the theme color value background1 mapped to the theme color light1 as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings:

```
<w:clrSchemeMapping w:bg1="light1" />
```

The clrSchemeMapping element's attribute background1 has a value of light1, specifying that theme color value background1 shall be mapped to the theme color light1. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
accent1 (Accent 1 Theme Color Mapping)	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content. If this attribute is omitted, then the accent1 theme color shall be used. [*Example*: Consider a WordprocessingML document that shall have references to the theme color accent1 mapped to the theme color lt1 as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings: `<w:clrSchemeMapping w:accent1="light1" />` The accent1 attribute has a value of light1, specifying that uses of the theme color value accent1 shall be mapped to the theme color lt1. *end example*] The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§2.18.12).
accent2 (Accent 2 Theme Color Mapping)	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content. If this attribute is omitted, then the accent2 theme color shall be used. [*Example*: Consider a WordprocessingML document that shall have the references to the theme color accent2 mapped to the theme color hlink as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings: `<w:clrSchemeMapping w:accent2="hyperlink" />` The accent2 attribute has a value of hyperlink, specifying that uses of the theme color value accent2 shall be mapped to the theme color hlink. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§2.18.12).
accent3 (Accent3 Theme Color Mapping)	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content. If this attribute is omitted, then the accent3 theme color shall be used. [*Example*: Consider a WordprocessingML document that shall have references to the theme color accent3 mapped to the theme color dk1 as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings: `<w:clrSchemeMapping w:accent3="dark1" />` The accent3 attribute has a value of dark1, specifying that uses of the theme color value accent3 shall be mapped to the theme color dk1. *end example*] The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§2.18.12).
accent4 (Accent4 Theme Color Mapping)	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content. If this attribute is omitted, then the accent4 theme color shall be used. [*Example*: Consider a WordprocessingML document that shall have references to the theme color accent4 mapped to the theme color dk2 as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings: `<w:clrSchemeMapping w:accent4="dark2" />` The accent4 attribute has a value of dark2, specifying that uses of the theme color value accent3 shall be mapped to the theme color dk2. *end example*] The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§2.18.12).
accent5 (Accent5 Theme Color Mapping)	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content. If this attribute is omitted, then the accent5 theme color shall be used. [*Example*: Consider a WordprocessingML document that shall have references to the theme color accent5 mapped to the theme color accent1 as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings:

Attributes	Description
	<w:clrSchemeMapping w:accent5="accent1" /> The accent5 attribute has a value of accent1, specifying that uses of the theme color value accent5 shall be mapped to the theme color accent1. *end example*] The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§2.18.12).
accent6 (Accent6 Theme Color Mapping)	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content. If this attribute is omitted, then the accent6 theme color shall be used. [*Example*: Consider a WordprocessingML document that shall have references to the theme color accent6 mapped to the theme color accent1 as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings: <w:clrSchemeMapping w:accent6="accent1" /> The accent6 attribute has a value of accent1, specifying that uses of the theme color value accent6 shall be mapped to the theme color accent1. *end example*] The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§2.18.12).
bg1 (Background 1 Theme Color Mapping)	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content. If this attribute is omitted, then the light1 theme color shall be used. [*Example*: Consider a WordprocessingML document that shall have references to the theme color bg1 mapped to the theme color lt2 as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings: <w:clrSchemeMapping w:bg1="light2" /> The bg1 attribute has a value of light2, specifying that uses of the theme color value bg1 shall be mapped to the theme color lt2. *end example*] The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§2.18.12).
bg2 (Background 2 Theme Color Mapping)	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content.

Attributes	Description
	If this attribute is omitted, then the light2 theme color shall be used. [*Example*: Consider a WordprocessingML document that shall have references to the theme color bg2 mapped to the theme color dk1 as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings: `<w:clrSchemeMapping w:bg2="dark1" />` The bg2 attribute has a value of dark1, specifying that uses of the theme color value bg2 shall be mapped to the theme color dk1. *end example*] The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§2.18.12).
followedHyperlink (Followed Hyperlink Theme Color Mapping)	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content. If this attribute is omitted, then the followedHyperlink theme color shall be used. [*Example*: Consider a WordprocessingML document that shall have references to the theme color followedHyperlink mapped to the theme color hyperlink as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings: `<w:clrSchemeMapping w:followedHyperlink="hyperlink" />` The followedHyperlink attribute has a value of hyperlink, specifying that uses of the theme color value followedHyperlink shall be mapped to the theme color hyperlink. *end example*] The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§2.18.12).
hyperlink (Hyperlink Theme Color Mapping)	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content. If this attribute is omitted, then the hyperlink theme color shall be used. [*Example*: Consider a WordprocessingML document that shall have references to the theme color hyperlink mapped to the theme color accent1 as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings: `<w:clrSchemeMapping w:hyperlink="accent1" />` The hyperlink attribute has a value of accent1, specifying that uses of the theme color

Attributes	Description
	value hyperlink shall be mapped to the theme color accent1. *end example*] The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§2.18.12).
t1 (Text 1 Theme Color Mapping)	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content. If this attribute is omitted, then the t1 theme color shall be used. [*Example*: Consider a WordprocessingML document that shall have references to the theme color t1 mapped to the theme color lt1 as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings: `<w:clrSchemeMapping w:t1="light1" />` The t1 attribute has a value of light1, specifying that uses of the theme color value t1 shall be mapped to the theme color lt1. *end example*] The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§2.18.12).
t2 (Text 2 Theme Color Mapping)	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content. If this attribute is omitted, then the t2 theme color shall be used. [*Example*: Consider a WordprocessingML document that shall have references to the theme color t2 mapped to the theme color dk1 as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings: `<w:clrSchemeMapping w:t2="dark1" />` The t2 attribute has a value of dark1, specifying that uses of the theme color value t2 shall be mapped to the theme color dk1. *end example*] The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§2.18.12).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ColorSchemeMapping">
   <attribute name="bg1" type="ST_ColorSchemeIndex"/>
   <attribute name="t1" type="ST_ColorSchemeIndex"/>
   <attribute name="bg2" type="ST_ColorSchemeIndex"/>
   <attribute name="t2" type="ST_ColorSchemeIndex"/>
   <attribute name="accent1" type="ST_ColorSchemeIndex"/>
   <attribute name="accent2" type="ST_ColorSchemeIndex"/>
   <attribute name="accent3" type="ST_ColorSchemeIndex"/>
   <attribute name="accent4" type="ST_ColorSchemeIndex"/>
   <attribute name="accent5" type="ST_ColorSchemeIndex"/>
   <attribute name="accent6" type="ST_ColorSchemeIndex"/>
   <attribute name="hyperlink" type="ST_ColorSchemeIndex"/>
   <attribute name="followedHyperlink" type="ST_ColorSchemeIndex"/>
</complexType>
```

2.15.1.21 consecutiveHyphenLimit (Maximum Number of Consecutively Hyphenated Lines)

This element specifies the maximum number of consecutive lines of text that can end with a hyphen when the contents of this document are displayed. Once this limit has been reached, the following line shall not be hyphenated regardless of whether or not it meets the criteria needed for hyphenation.

If this element is omitted or has its val attribute equal to `0`, the given WordprocessingML document shall have no limit on the number of consecutive lines of text .nehpyh a htiw dne yam taht

[*Example*: Consider a WordprocessingML document which should automatically be hyphenated. If the contents of this document result in hyphens appearing on every line in the document, as follows:

Test test test Mississippi The Mississippi The Mississippi The Mississippi The Mississippi The Mis-
sissippi The Mississippi The Mississippi The Mississippi The Mississippi The Mississippi The Mis-
sissippi The Mississippi The Mississippi The Mississippi The Mississippi The Mississippi The Mis-
sissippi

This output may be undesirable. If the document shall have a maximum of two consecutive hyphens, this requirement is specified using the following WordprocessingML in the document settings:

```
<w:consecutiveHyphenLimit w:val="2" />
```

The consecutiveHyphenLimit element's val attribute has a value of 2 specifying that a maximum of two hyphens should be allowed, limiting the hyphenation output like this:

Test test test Mississippi The Mississippi The Mississippi The Mississippi The Mississippi The Mis-
sissippi The Mississippi The Mississippi The Mississippi The Mississippi The Mississippi The Mis-
sissippi The Mississippi The Mississippi The Mississippi The Mississippi The Mississippi The
Mississippi

end example]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.15.1.22 decimalSymbol (Radix Point for Field Code Evaluation)

This element specifies the character that shall be interpreted as the radix point when evaluating the contents of all fields in the current document.

[*Rationale*: When evaluating field instructions based on the contents of the current document, it is necessary to know the character which shall be treated as the radix point in order to prevent changes to the calculation of the same field instructions based on the current user's locale. This element stores the radix point which shall be used to evaluate fields in the contents of this document, irrespective of the locale of the application loading the file. *end rationale*]

If this element is omitted, the application shall use the default radix point of its current locale setting to evaluate field instructions. If this element's attribute value is more than a single character, then the document is non-conformant.

[*Example*: Consider a WordprocessingML document which should use the comma character as the radix point for all field instructions. This requirement is specified using the following WordprocessingML in the document settings:

```
<w:decimalSymbol w:val="," />
```

The decimalSymbol element's val attribute has a value of `,` specifying that the comma character shall be interpreted as the radix point.

For instance, the string `12.345,00` would be interpreted as a numeric value of twelve thousand three hundred and forty five. If the decimalSymbol was a period, the same string would be twelve and three hundred and forty five thousandths. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.15.1.23 defaultTableStyle (Default Table Style for Newly Inserted Tables)

This element specifies the table style which shall automatically be applied to the table properties of tables added to this document by an application. Note that it does not change the table style applied to tables which

do not reference a style, instead, it automatically applies the style to that table via the tblStyle element (§2.4.59). This link is made by referencing the styleId attribute value of the table style which shall be used to format newly inserted tables.

If this element is omitted, then no table style shall automatically be applied to inserted tables (therefore inheriting the default table style). If the referenced style is not present or not a table style, then no table style shall automatically be applied to inserted tables.

[*Example*: Consider a WordprocessingML document which should use the LightShading-Accent3 style. This requirement is specified using the following WordprocessingML in the document settings:

```
<w:defaultTableStyle w:val="LightShading-Accent3" />
```

The corresponding table style must therefore exist in the styles part:

```
<w:style w:type="table" w:styleId="LightShading-Accent3">
  …
</w:style>
```

The defaultTableStyle element's val attribute has a value of `LightShading-Accent3` specifying that that style will be applied automatically to newly inserted tables. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`

Attributes	Description
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.15.1.24 defaultTabStop (Distance Between Automatic Tab Stops)

This element specifies the value which shall be used as the multiplier to generate automatic tab stops in this document. *Automatic tab stops* refer to the tab stop locations which occur after all custom tab stops in the current paragraph have been surpassed.

If this element is omitted, then automatic tab stops should be generated at 720 twentieths of a point (0.5") intervals across the displayed page.

[*Example*: Consider a WordprocessingML document which should have automatic tab stops every 360 twentieths of a point (0.25 inches). This requirement is specified using the following WordprocessingML in the document settings:

```
<w:defaultTabStop w:val="360" />
```

The defaultTabStop element's val attribute has a value of `360` specifying that automatic tab stops shall occur every 1/4th of an inch across the page.

If a custom tab stop was located at 2.28", then the next three automatic tab stops would be at 2.5", 2.75" and 3.0" (the next three multiples of the default tab stop value). *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (Measurement in Twentieths of a Point)	Specifies a positive measurement value, specified in twentieths of a point. This value is interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML element with a val attribute containing a positive measurement in twentieths of a point: `<w:… w:val="720" />`

Attributes	Description
	The val attribute has a value of 720, specifying that this measurement value is 720 twentieths of a point (0.5"). This value is interpreted by the parent element as needed. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TwipsMeasure">
   <attribute name="val" type="ST_TwipsMeasure" use="required"/>
</complexType>
```

2.15.1.25 displayBackgroundShape (Display Background Objects When Displaying Document)

This element specifies whether the images and colors defined in the document's background using the background element (§2.2.1) shall be displayed when the document is displayed in print layout view as specified in the view element (§2.15.1.93).

If this element is omitted, then background shapes shall not be displayed when the document is displayed in print layout view.

[*Example*: Consider a WordprocessingML document that has a turquoise background specified for all pages and is being displayed in page layout view, as follows:

If the document's background should not be displayed, that requirement would be specified using the following WordprocessingML in the document settings:

```
<w:displayBackgroundShape w:val="true" />
```

The resulting document would display the background in page layout view:

end example]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.26 displayHorizontalDrawingGridEvery (Distance between Horizontal Gridlines)

This element specifies the number of horizontal grid units defined using the drawingGridHorizontalSpacing element (§2.15.1.44) which shall be allowed between subsequent visible horizontal drawing grid lines in this document, if gridlines are being shown. [*Note*: The display of gridlines is an application-level setting not specified in this Office Open XML Standard. *end note*] The *drawing grid* is a grid which may be used by applications to help position floating objects in the document.

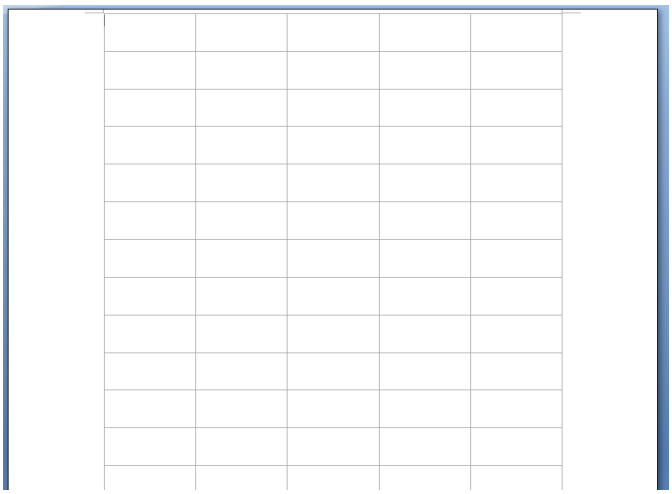
If this element is omitted, then gridlines shall be displayed for each horizontal grid unit.

[*Example*: Consider the image below illustrating a WordprocessingML document in which all horizontal grid units are visible (the default setting):

If the gridlines in this document shall only be displayed for every 4th horizontal drawing gridline, that requirement would be specified using the following WordprocessingML in the document settings:

```
<w:displayHorizontalDrawingGridEvery w:val="4" />
```

The resulting grid would look like the following:



The displayHorizontalDrawingGridEvery element has its val attribute equal to 4, therefore every fourth gridline is displayed in the document when the drawing grid is turned on. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

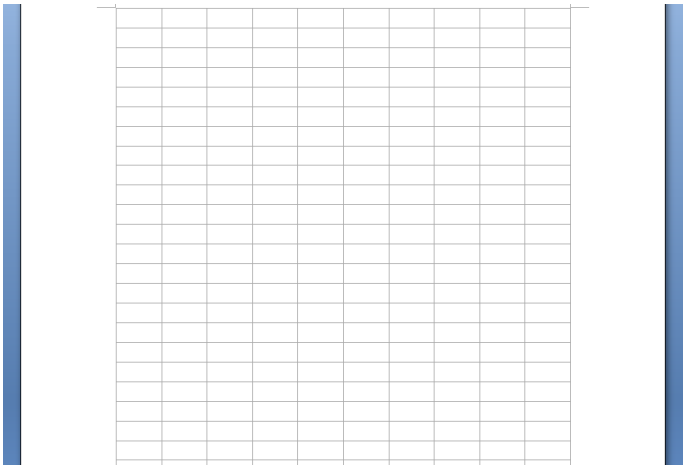
```
<complexType name="CT_DecimalNumber">
    <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.15.1.27 displayVerticalDrawingGridEvery (Distance between Vertical Gridlines)

This element specifies the number of vertical grid units defined using the drawingGridVerticalSpacing element (§2.15.1.46) which shall be allowed between subsequent vertical gridlines in this document, if gridlines are being shown. [*Note*: The display of gridlines is an application-level setting not specified in this Office Open XML Standard. *end note*] The *drawing grid* is a grid which may be used by applications to help position floating objects in the document.

If this element is omitted, then vertical gridlines shall not be displayed.

[*Example*: Consider the image below illustrating a WordprocessingML document in which all vertical grid units are visible (the default setting):



If the vertical drawing gridlines in this document shall only be displayed for every 4th gridline, that requirement would be specified using the following WordprocessingML in the document settings:

```
<w:displayVerticalDrawingGridEvery w:val="4" />
```

The resulting grid would look like the following:

The displayVerticalDrawingGridEvery element has its val attribute equal to 4, therefore every fourth vertical gridline is displayed in the document when the drawing grid is turned on. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element.

Attributes	Description
	[*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.15.1.28 documentProtection (Document Editing Restrictions)

This element specifies the set of document protection restrictions which have been applied to the contents of a WordprocessingML document. These restrictions shall be enforced by applications editing this document when the enforcement attribute is turned on, and should be ignored (but persisted) otherwise. *Document protection* is a set of restrictions used to prevent unintentional changes to all or part of a WordprocessingML document - since this protection does not encrypt the document, malicious applications may circumvent its use. This protection is not intended as a security feature and may be ignored.

If this element is omitted, then no protection shall be applied to this document.

When a password is supplied via an application which shall be hashed and stored in this element, that process shall be done in two stages:

First, the password shall be hashed using the following algorithm:

- Truncate the password to 15 characters.
- Construct a new NULL-terminated string consisting of single-byte characters:
    - Get the single-byte values by iterating through the Unicode characters of the truncated password. For each character, if the low byte is not equal to 0, take it. Otherwise, take the high byte.
- From now on, the single-byte character string is used.
- If the password is empty, return 0.
- Compute the high-order word of the new key:
    - Initialize from the initial code array (see below), depending on the password's length. For each character in the password:

        - For every bit in the character, starting with the least significant and progressing to (but excluding) the most significant, if the bit is set, XOR the key's high-order word with the corresponding word from the encryption matrix

- Compute the low-order word of the new key:
    - Initialize with 0
        - For each character in the password, going backwards, low-order word = (((low-order word SHR 14) AND 0x0001) OR (low-order word SHL 1) AND 0x7FFF)) XOR character
        - Lastly, low-order word = (((low-order word SHR 14) AND 0x0001) OR (low-order word SHL 1) AND 0x7FFF)) XOR password length XOR 0xCE4B.

Initial code array

The initial code array contains the initial values for the key's high-order word. The initial value depends on the length of the password, as follows:

Password length	Initial value for the key's high-order word
1	0xE1F0
2	0x1D0F
3	0xCC9C
4	0x84C0
5	0x110C
6	0x0E10
7	0xF1CE
8	0x313E
9	0x1872
10	0xE139
11	0xD40F
12	0x84F9
13	0x280C
14	0xA96A
15	0x4EC3

Encryption matrix

The encryption matrix contains codes used during the calculation of the key's high-order word. As described in the algorithm above, for every bit of the password's characters, if the bit is set, a corresponding value is taken from this encryption matrix and is used to XOR the key's high-order word with it. Each row in the encryption matrix corresponds to a single character from the password, and each of the seven columns corresponds to a particular bit (0-6) in this character.

The values are taken in such a way so that the last character of the password uses the last row in the encryption matrix. The next-to-last character uses the next-to-last row in the matrix, and so on. This means that the beginning of the matrix may be unused, depending on the length of the password.

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6
Last-14	0xAEFC	0x4DD9	0x9BB2	0x2745	0x4E8A	0x9D14	0x2A09
Last-13	0x7B61	0xF6C2	0xFDA5	0xEB6B	0xC6F7	0x9DCF	0x2BBF
Last-12	0x4563	0x8AC6	0x05AD	0x0B5A	0x16B4	0x2D68	0x5AD0
Last-11	0x0375	0x06EA	0x0DD4	0x1BA8	0x3750	0x6EA0	0xDD40
Last-10	0xD849	0xA0B3	0x5147	0xA28E	0x553D	0xAA7A	0x44D5
Last-9	0x6F45	0xDE8A	0xAD35	0x4A4B	0x9496	0x390D	0x721A
Last-8	0xEB23	0xC667	0x9CEF	0x29FF	0x53FE	0xA7FC	0x5FD9
Last-7	0x47D3	0x8FA6	0x0F6D	0x1EDA	0x3DB4	0x7B68	0xF6D0
Last-6	0xB861	0x60E3	0xC1C6	0x93AD	0x377B	0x6EF6	0xDDEC
Last-5	0x45A0	0x8B40	0x06A1	0x0D42	0x1A84	0x3508	0x6A10
Last-4	0xAA51	0x4483	0x8906	0x022D	0x045A	0x08B4	0x1168
Last-3	0x76B4	0xED68	0xCAF1	0x85C3	0x1BA7	0x374E	0x6E9C
Last-2	0x3730	0x6E60	0xDCC0	0xA9A1	0x4363	0x86C6	0x1DAD
Last-1	0x3331	0x6662	0xCCC4	0x89A9	0x0373	0x06E6	0x0DCC
Last	0x1021	0x2042	0x4084	0x8108	0x1231	0x2462	0x48C4

[*Example*: Consider a password which has been supplied - the string “Example”. It is already under 15 characters, so truncation does not affect it. It is then converted to a string of single-byte characters.

- The password is 7 characters long, so, from the initial code array, the initial value for the key’s high-order word is 0xF1CE.
- The key’s high-order word is then computed further depending on the password’s characters:
    - The first character is ‘E’ (0x45). This is the first character of a 7-character password, so its corresponding row in the encryption matrix is “Last-6”.
        - Bit 0 is set, therefore the key’s high-order word is combined (via XOR) with the corresponding value for Bit 0 on row “Last-6”, which is 0xB861. The new result is 0xF1CE XOR 0xB861 = 0x49AF.
        - Bit 2 is set, so the key’s high-order word is XOR-ed with the corresponding value for Bit 2 on row “Last-6”, which is 0xC1C6. The new result is 0x49AF XOR 0xC1C6 = 0x8869.
        - This process is repeated for each bit.
    - The next character is ‘x’ (0x78). Its corresponding row in the encryption matrix is “Last-5”.

        - Bit 3 is set. The value for Bit 3 on row “Last-5” in the encryption matrix is 0x0D42. The current value for the key’s high-order byte is 0x5585, so the new one should be 0x5585 XOR 0x0D42 = 0x58C7.
        - This process is repeated for each bit.
    - This process is repeated for all characters.
- After the last character has been processed, the above step produced 0x64CE for the key’s high-order word. Now the low-order word needs to be calculated:
    - The initial value is 0.
    - It is then calculated using the password:
        - The last character of the password is ‘e’ (0x65), so, by the formula, low-order word = (((low-order word SHR 14) AND 0x0001) OR ((low-order word SHL 1) AND 0x7FFF)) XOR ‘e’ = (((0 SHR 14) AND 0x0001) OR ((0 SHL 1) AND 0x7FFF)) XOR 0x65 = 0x0065.
        - The next to last character of the password is ‘l’ (0x6C). Again, by the formula, (((0x0065 SHR 14) AND 0x0001) OR ((0x0065 SHL 1) AND 0x7FFF)) XOR 0x6C = (0x0000 OR 0x00CA) XOR 0x6C = 0x00CA XOR 0x6C = 0x00A6.
        - This process is repeated for each character.
    - After the password’s first character has been processed, we have 0x1199 for the key’s low-order word. Lastly, the password’s length is combined into it: low-order word = (((0x1199 SHR 14) AND 0x0001) OR ((0x1199 SHL 1) AND 0x7FFF)) XOR 0x0007 XOR 0xCE4B = 0x2332 XOR 0x0007 XOR 0xCE4B = 0x2335 XOR 0xCE4B = 0xED7E.
- The end result for the key is 0x64CEED7E.

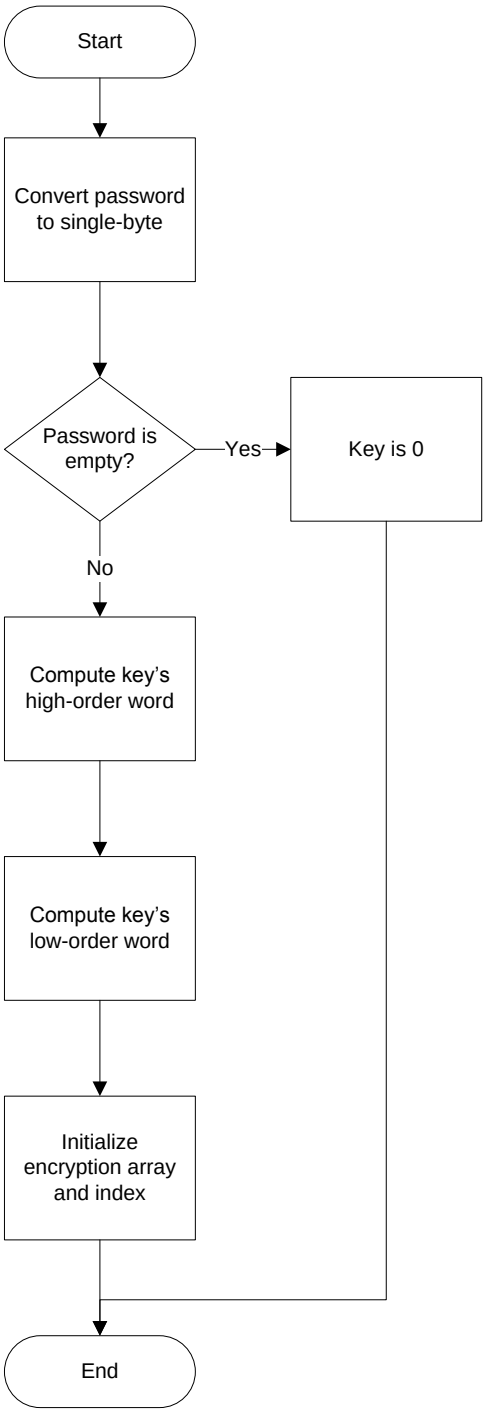
end example]

[*Rationale*: This pre-processing step is necessary for compatibility with legacy word processing applications which hashed their password solely using this mechanism. *end rationale*]

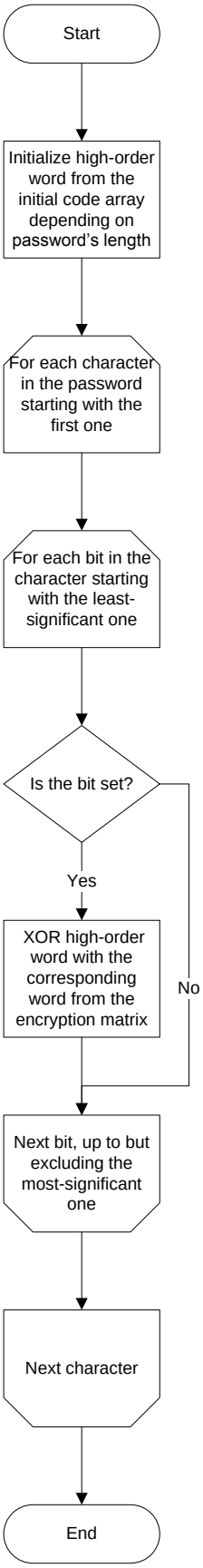
Second, the byte order of the result shall be reversed [*Example*: 0x64CEED7E becomes 7EEDCE64. *end example*], and that value shall be hashed as defined by the attribute values.

[*Note*: The algorithm above can be stated as follows using diagrams:

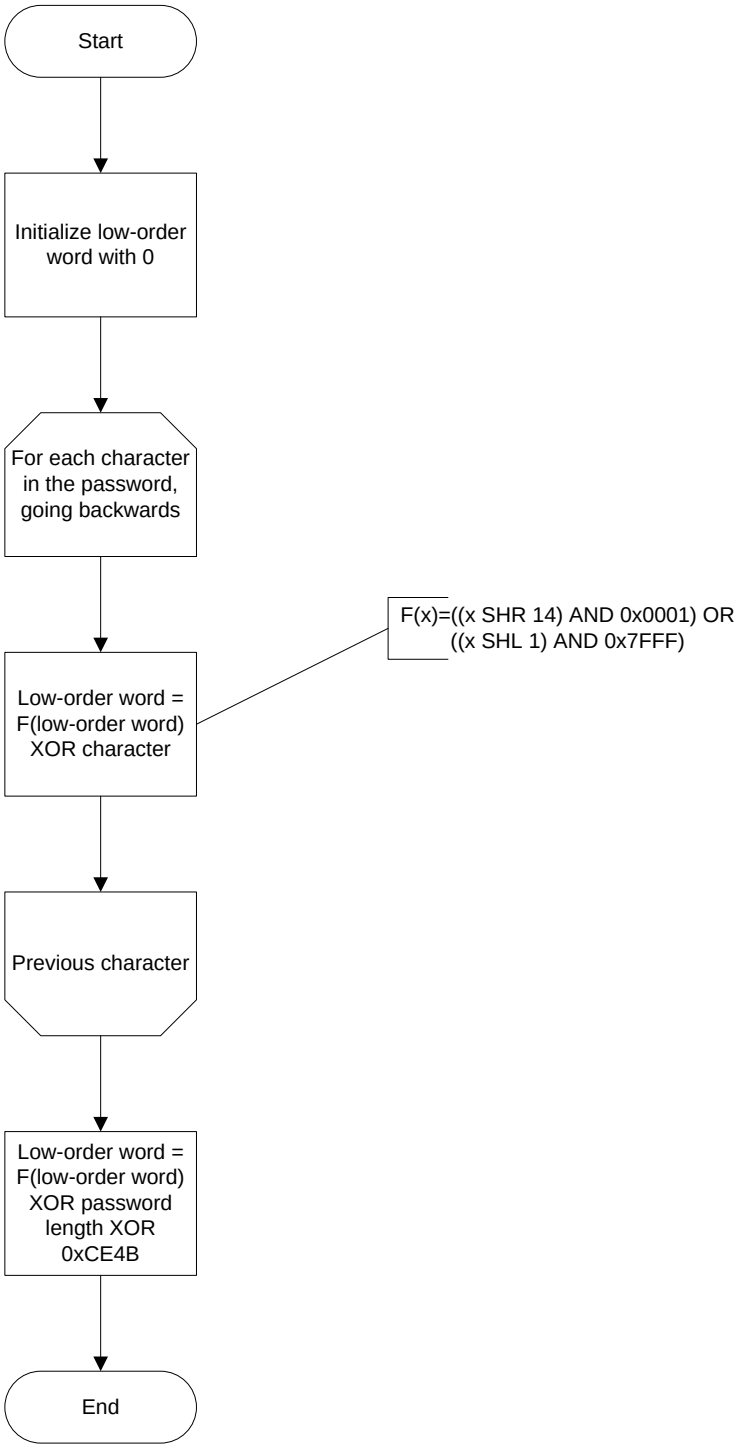
Calculate Key



Compute Key's High-Order Word



Compute Key's Low-Order Word



end note]

[*Example*: Consider a WordprocessingML document which specifies that applications shall not allow any modifications to this document other than the addition of comments. This requirement would be specified using the following WordprocessingML in the document settings:

```
<w:documentProtection w:edit="comments" w:enforcement="true" …
  w:cryptAlgorithmClass="hash" w:cryptAlgorithmType="typeAny"
  w:cryptAlgorithmSid="1" w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" />
```

The documentProtection element has an edit attribute value of comments, specifying that the only modification allowed should be comments, the enforcement attribute has a value of true, specifying that the document protection specified is to be enforced on the given document. Finally, in order for the hosting application to stop enforcement of the document protection applied to the document, the hosting application would have to be provided with a password that the hosting application would then hash, compare to the value of the hash attribute (9oN7nWkCAyEZib1RomSJTjmPpCY=), and if the two values matched, halt enforcement of any document protection. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description

algIdExt (Cryptographic Algorithm Extensibility)	Specifies that a cryptographic algorithm which was not defined by this Office Open XML Standard has been used to generate the hash value stored with this document.
	
	This value, when present, shall be interpreted based on the value of the algIdExtSource attribute in order to determine the algorithm used, which shall be application-defined. [*Rationale*: This extensibility affords the fact that with exponentially increasing computing power, documents created in the future will likely need to utilize as yet undefined hashing algorithms in order to remain secure. *end rationale*]
	
	If this value is present, the cryptAlgorithmClass, cryptAlgorithmType, and cryptAlgorithmSid attribute values shall be ignored in favor of the algorithm defined by this attribute.
	
	[*Example*: Consider a WordprocessingML document with the following information stored in one of its protection elements:
	
	`<w:… w:algIdExt="0000000A"`
	`w:algIdExtSource="futureCryptography"`
	`w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" />`
	
	The algIdExt attribute value of 0000000A specifies that the algorithm with hex code A shall be used as defined by the futureCryptography application. *end example*]

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).</td></tr>
<tr><td>algIdExtSource (Algorithm Extensibility Source)</td><td>Specifies the application which defined the algorithm value specified by the algIdExt attribute.

[Example: Consider a WordprocessingML document with the following information stored in one of its protection elements:

<pre><w:… w:algIdExt="0000000A"
  w:algIdExtSource="futureCryptography"
  w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" /></pre>
The algIdExtSource attribute value of futureCryptography specifies that the algorithm used here was published by the futureCryptography application. end example]

The possible values for this attribute are defined by the ST_String simple type (§2.18.89).</td></tr>
<tr><td>cryptAlgorithmClass (Cryptographic Algorithm Class)</td><td>Specifies the class of cryptographic algorithm used by this protection. [Note: The initial version of this Office Open XML Standard only supports a single version - hash - but future versions may expand this as necessary. end note]

[Example: Consider a WordprocessingML document with the following information stored in one of its protection elements:

<pre><w:… w:cryptAlgorithmClass="hash"
  w:cryptAlgorithmType="typeAny"
  w:cryptAlgorithmSid="1"
  w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" /></pre>
The cryptAlgorithmClass attribute value of hash specifies that the algorithm used for the password is a hashing algorithm. end example]

The possible values for this attribute are defined by the ST_AlgClass simple type (§2.18.1).</td></tr>
<tr><td>cryptAlgorithmSid (Cryptographic Hashing Algorithm)</td><td>Specifies the specific cryptographic hashing algorithm which shall be used along with the salt attribute and user-supplied password in order to compute a hash value for comparison.

The possible values for this attribute shall be interpreted as follows:
<table>
<tr><th>Value</th><th>Algorithm</th></tr>
<tr><td>1</td><td>MD2</td></tr>
<tr><td>2</td><td>MD4</td></tr>
<tr><td>3</td><td>MD5</td></tr>
<tr><td>4</td><td>SHA-1</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>5</td><td>MAC</td></tr>
<tr><td>6</td><td>RIPEMD</td></tr>
<tr><td>7</td><td>RIPEMD-160</td></tr>
<tr><td>8</td><td>Undefined. Shall not be used.</td></tr>
<tr><td>9</td><td>HMAC</td></tr>
<tr><td>10</td><td>Undefined. Shall not be used.</td></tr>
<tr><td>11</td><td>Undefined. Shall not be used.</td></tr>
<tr><td>12</td><td>SHA-256</td></tr>
<tr><td>13</td><td>SHA-384</td></tr>
<tr><td>14</td><td>SHA-512</td></tr>
<tr><td>Any other value</td><td>Undefined. Shall not be used.</td></tr>
</table>
[Example: Consider a WordprocessingML document with the following information stored in one of its protection elements:
<pre><w:… w:cryptAlgorithmClass="hash"
  w:cryptAlgorithmType="typeAny"
  w:cryptAlgorithmSid="1"
  w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" /></pre>
The cryptAlgorithmSid attribute value of 1 specifies that the SHA-1 hashing algorithm shall be used to generate a hash from the user-defined password. end example]
The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
</td></tr>
<tr><td>cryptAlgorithmType (Cryptographic Algorithm Type)</td><td>
Specifies the type of cryptographic algorithm used by this protection. [Note: The initial version of this Office Open XML Standard only supports a single type - typeAny - but future versions may expand this as necessary. end note]
[Example: Consider a WordprocessingML document with the following information stored in one of its protection elements:
<pre><w:… w:cryptAlgorithmClass="hash"
  w:cryptAlgorithmType="typeAny"
  w:cryptAlgorithmSid="1"
  w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" /></pre>
The cryptAlgorithmType attribute value of typeAny specifies that any type of algorithm may have been used for the password. end example]
The possible values for this attribute are defined by the ST_AlgType simple type
</td></tr>
</table>

Attributes	Description
	(§2.18.2).
cryptProvider (Cryptographic Provider)	Specifies the cryptographic provider which was used to generate the hash value stored in this document. If the user provided a cryptographic provider which was not the system's built-in provider, then that provider shall be stored here so it can subsequently be used if available. If this attribute is omitted, then the built-in cryptographic provider on the system shall be used. [*Example*: Consider a WordprocessingML document with the following information stored in one of its protection elements: `<w:… w:cryptProvider="Krista'sProvider" w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" />` The cryptProvider attribute value of `Krista'sProvider` specifies that the cryptographic provider with name "Krista's Provider" shall be used if available. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).
cryptProviderType (Cryptographic Provider Type)	Specifies the type of cryptographic provider to be used. [*Example*: Consider a WordprocessingML document with the following information stored in one of its protection elements: `<w:… w:cryptProviderType="rsaAES" w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" />` The cryptProviderType attribute value of `rsaAES` specifies that the cryptographic provider type shall be an Advanced Encryption Standard provider. *end example*] The possible values for this attribute are defined by the ST_CryptProv simple type (§2.18.14).
cryptProviderTypeExt (Cryptographic Provider Type Extensibility)	Specifies that a cryptographic provider type which was not defined by this Office Open XML Standard has been used to generate the hash value stored with this document. This value, when present, shall be interpreted based on the value of the cryptProviderTypeExtSource attribute in order to determine the provider type used, which shall be application-defined. [*Rationale*: This extensibility affords the fact that with exponentially increasing computing power, documents created in the future will likely need to utilize as yet undefined cryptographic provider types in order to remain secure. *end rationale*] If this value is present, the cryptProviderType attribute value shall be ignored in favor of the provider type defined by this attribute.

Attributes	Description

	[*Example*: Consider a WordprocessingML document with the following information stored in one of its protection elements:
	
	`<w:… w:cryptProviderTypeExt="00A5691D"`
	`w:cryptProvideTypeExtSource="futureCryptography"`
	`w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" />`
	
	The cryptProviderTypeExt attribute value of `00A5691D` specifies that the provider type associated with hex code `A5691D` shall be used as defined by the `futureCryptography` application. *end example*]
	
	The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).

cryptProviderTypeExtSource (Provider Type Extensibility Source)	Specifies the application which defined the provider type value specified by the cryptProviderTypeExt attribute.
	
	[*Example*: Consider a WordprocessingML document with the following information stored in one of its protection elements:
	
	`<w:… w:cryptProviderTypeExt="00A5691D"`
	`w:cryptProvideTypeExtSource="futureCryptography"`
	`w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" />`
	
	The cryptProvideTypeExtSource attribute value of `futureCryptography` specifies that the provider type used here was published by the `futureCryptography` application. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

cryptSpinCount (Iterations to Run Hashing Algorithm)	Specifies the number of times the hashing function shall be iteratively run (using each iteration's result as the input for the next iteration) when attempting to compare a user-supplied password with the value stored in the hash attribute. [*Rationale*: Running the algorithm many times increases the cost of exhaustive search attacks correspondingly. Storing this value allows for the number of iterations to be increased over time to accommodate faster hardware (and hence the ability to run more iterations in less time). *end rationale*]
	
	[*Example*: Consider a WordprocessingML document with the following information stored in one of its protection elements:
	
	`<w:… w:cryptSpinCount="100000"`
	`w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" />`
	
	The cryptSpinCount attribute value of `100000` specifies that the hashing function shall be run one hundred thousand times to generate a hash value for comparison with the hash attribute. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
edit (Document Editing Restrictions)	Specifies the set of editing restrictions which shall be enforced on a given WordprocessingML document, as defined by the simple type referenced below If this attribute is omitted, the consumer shall behave as though there are no editing restrictions applied to this document; equivalent to an attribute value of none. [*Example*: Consider a WordprocessingML document that contains the following WordprocessingML specifying that hosting applications shall enforce read-only protection for a given document: `<w:documentProtection w:edit="readOnly" w:enforcement="1" />` The edit attribute has a value of `readOnly` and a enforcement attribute with a value of `1`, specifying that read-only document protection shall be enforced on the given document. e*nd example*] The possible values for this attribute are defined by the ST_DocProtect simple type (§2.18.22).
enforcement (Enforce Document Protection Settings)	Specifies if the document protection settings shall be enforced for a given WordprocessingML document. If the value of this element is `off`, `0`, or `false`, all the WordprocessingML pertaining to document protection is still preserved in the document, but is not enforced. If the value of this element is on, 1, or `true`, the document protection is enforced. If this attribute is omitted, then document protection settings shall not be enforced by applications. [*Example*: Consider a WordprocessingML document that contains the following WordprocessingML specifying that hosting applications shall apply read-only protection for a given document: `<w:documentProtection w:edit="readOnly" w:enforcement="1" />` The enforcement attribute has a value of 1, specifying that the document protection specified shall be enforced on the given document. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
formatting (Only Allow Formatting With Unlocked Styles)	Specifies if formatting restrictions are in effect for a given WordprocessingML document. This enables the document to restrict the types of styles that may exist in a given WordprocessingML document. Specifically, by setting this attribute's value equal to `true`, every style whose locked element (§2.7.3.7) has a value of `true` (or latent styles (§2.7.3.5) whose locked attribute is `true`) shall not be available for use in the application, nor should any direct formatting. Only styles with a locked value of `false` may be used.

Attributes	Description

	If this attribute is omitted, then no formatting restrictions shall be applied, even when document protection is enforced.
	
	[*Example*: Consider a WordprocessingML document that shall apply formatting protection. This requirement would be specified using the following WordprocessingML in the document settings:
	
	`<w:documentProtection w:formatting="true" w:enforcement="true" />`
	
	If the following definition for a style was also present in the document:
	
	`<w:style w:type="paragraph" w:styleId="Heading1">`
	`<w:name w:val="heading 1" />`
	`<w:locked="1" />`
	`…`
	`</w:style>`
	
	The formatting attribute has a value of `true` specifying that the applications shall not allow the style above to be added to the WordprocessingML document. This does not preclude previous uses of that style (which shall not be removed), but does prevent new uses of this style from being added. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

hash (Password Hash)	Specifies the hash value for the password stored with this document. This value shall be compared with the resulting hash value after hashing the user-supplied password using the algorithm specified by the preceding attributes and parent XML element, and if the two values match, the protection shall no longer be enforced.
	
	If this value is omitted, then no password shall be associated with the protection, and it may be turned off without supplying any password.
	
	[*Example*: Consider a WordprocessingML document with the following information stored in one of its protection elements:
	
	`<w:… w:cryptAlgorithmClass="hash"`
	`w:cryptAlgorithmType="typeAny"`
	`w:cryptAlgorithmSid="1"`
	`w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" />`
	
	The hash attribute value of `9oN7nWkCAyEZib1RomSJTjmPpCY=` specifies that the user-supplied password shall be hashed using the pre-processing defined by the parent element (if any) followed by the SHA-1 algorithm (specified via the cryptAlgorithmSid attribute value of `1`) and that the resulting has value must be `9oN7nWkCAyEZib1RomSJTjmPpCY=` for the protection to be disabled. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the XML Schema base64Binary datatype.
salt (Salt for Password Verifier)	Specifies the salt which was prepended to the user-supplied password before it was hashed using the hashing algorithm defined by the preceding attribute values to generate the hash attribute, and which shall also be prepended to the user-supplied password before attempting to generate a hash value for comparison. A *salt* is a random string which is added to a user-supplied password before it is hashed in order to prevent a malicious party from pre-calculating all possible password/hash combinations and simply using those precalculated values (often referred to as a "dictionary attack"). If this attribute is omitted, then no salt shall be prepended to the user-supplied password before it is hashed for comparison with the stored hash value. [*Example*: Consider a WordprocessingML document with the following information stored in one of its protection elements: `<w:… w:salt="ZUdHa+D8F/OAKP3I7ssUnQ==" w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" />` The salt attribute value of `ZUdHa+D8F/OAKP3I7ssUnQ==` specifies that the user-supplied password shall have this value prepended before it is run through the specified hashing algorithm to generate a resulting hash value for comparison. *end example*] The possible values for this attribute are defined by the XML Schema base64Binary datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DocProtect">
   <attribute name="edit" type="ST_DocProtect" use="optional"/>
   <attribute name="formatting" type="ST_OnOff" use="optional"/>
   <attribute name="enforcement" type="ST_OnOff"/>
   <attributeGroup ref="AG_Password"/>
</complexType>
```

2.15.1.29 documentType (Document Classification)

This element specifies the classification of a given WordprocessingML document as a letter, email, or general document.

[*Note*: This element may be used by hosting applications to facilitate customized user interface and/or automatic formatting behaviors based on the 'type' of a given WordprocessingML document. *end note*]

If this element is omitted, then the document shall be classified as a general document.

[*Example*: Consider a set of WordprocessingML documents which should be classified as 'letters'. This classification would be specified using the following WordprocessingML in the document settings of these documents:

```
<w:documentType w:val="letter" />
```

The documentType element's val attribute is equal to `letter`, specifying that the hosting application shall apply the behaviors it has specified for letters to the given WordprocessingML document. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (Document Classification Value)	Specifies the classification of the document based on the types defined in the referenced simple type definition. [*Example*: Consider a WordprocessingML document which should be classified as an e-mail message. This classification would be specified using the following WordprocessingML in the document settings: `<w:documentType w:val="eMail" />` The val attribute is equal to `eMail`, specifying that the hosting application may apply e-mail behaviors (if any) to this document. *end example*] The possible values for this attribute are defined by the ST_DocType simple type (§2.18.23).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DocType">
   <attribute name="val" type="ST_DocType" use="required"/>
</complexType>
```

2.15.1.30 docVar (Single Document Variable)

This element specifies the parameters of a single document variable. A *document variable* is a storage location for arbitrary customer data in name/value pairs that will be persisted in a given WordprocessingML document. Specifically, this element specifies through its name and val attributes the name and value pair for a given document variable.

[*Note*: This mechanism is maintained for legacy compatibility only, and should be avoided in favor of the custom XML data support defined in this Office Open XML Standard. *end note*]

[*Example*: Consider the following WordprocessingML fragment specifying a document variable named `example` and containing the value `example value`:

```
<w:docVars>
  <w:docVar w:name="example" w:val="example value" />
</w:docVars>
```

The docVar element defines a single document variable, named `example` using the name attribute, and assigned the value `example value` through the val attribute. *end example*]

Parent Elements
docVars (§2.15.1.31)

Attributes	Description

name (Document Variable Name)	Specifies the name of the parent document variable.
	
	[*Example*: Consider the following WordprocessingML fragment specifying a document variable:
	
	`<w:docVars>`
	`  <w:docVar w:name="example name" w:val="example value" />`
	`</w:docVars>`
	
	The name attribute specifies that the name of the document variable is `example name`. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

val (Document Variable Value)	Specifies the value of the parent document variable.
	
	[*Example*: Consider the following WordprocessingML fragment specifying a document variable:
	
	`<w:docVars>`
	`  <w:docVar w:name="example name" w:val="Tristan Davis" />`
	`</w:docVars>`
	
	The val attribute specifies that the value of the document variable is `Tristan Davis`. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DocVar">
   <attribute name="name" type="ST_String" use="required"/>
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.15.1.31 docVars (Document Variables)

This element specifies the presence of document variables in a WordprocessingML. A *document variable* is a storage location for arbitrary customer data in name/value pairs that will be persisted in a given WordprocessingML document.

[*Note*: This mechanism is maintained for legacy compatibility only, and should be avoided in favor of the custom XML data support defined in this Office Open XML Standard. *end note*]

[*Example*: Consider the following WordprocessingML specifying three document variables:

```
<w:docVars>
  <w:docVar … />
  <w:docVar … />
  <w:docVar … />
</w:docVars>
```

The docVars element contains three child elements each defining a single document variable in this document. *end example*]

Parent Elements
settings (§2.15.1.78)

Child Elements	Subclause
docVar (Single Document Variable)	§2.15.1.30

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DocVars">
   <sequence>
      <element name="docVar" type="CT_DocVar" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

2.15.1.32 doNotAutoCompressPictures (Do Not Automatically Compress Images)

This element specifies that pictures in this document shall not automatically be compressed when saving the document in order to reduce the overall size of the resulting WordprocessingML document.

If this element is omitted, applications may perform basic compression on images before saving the contents of the document.

[*Example*: Consider a WordprocessingML document which should never have its images compressed before they are saved. This requirement would be specified using the following WordprocessingML:

```
<w:doNotAutoCompressPictures w:val="true"/>
```

The doNotAutoCompressPictures element's val attribute has a value of `true` specifying that images shall not be automatically compressed when the document is saved. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.33 doNotDemarcateInvalidXml (Do Not Show Visual Indicator For Invalid Custom XML Markup)

This element specifies whether a visual cue should be displayed around content contained in a WordprocessingML document which is contained with custom XML markup specified via the customXml element when an application determines that the current XML markup (or its contents) violate the constraints of the attached XML schema(s).

If this element is not present in a WordprocessingML document visual cues shall be displayed on content contained in custom XML markup in a WordprocessingML document which is considered to be invalid based on the associated XML schema(s).

[*Example*: Consider a WordprocessingML document which should show no visual indication of invalid custom XML markup. This requirement would be specified using the following WordprocessingML:

```
<w:doNotDemarcateInvalidXml w:val="true"/>
```

The doNotDemarcateInvalidXml element's val attribute has a value of `true` specifying the display of any visual indication of invalid custom XML markup shall be suppressed for this document. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.34 doNotDisplayPageBoundaries (Do Not Display Visual Boundary For Header/Footer or Between Pages)

This element specifies whether applications displaying this document should display the contents of the header and footer when displaying the document in print layout view (§2.15.1.93) or should collapse those areas as well as the whitespace on all displayed pages so that the text extents are directly following one another. [*Rationale*: Collapsing the ends of pages makes it easier to read the contents of the document, since the text flows between pages without whitespace, while maintaining the WYSIWYG functionality of print layout view for the document's main content. *end rationale*]

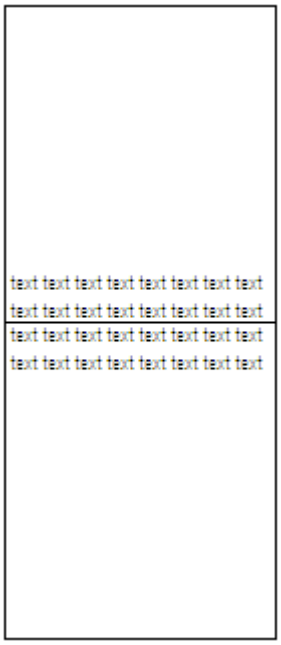
If this element is omitted, then all pages should be shown at their full size (including whitespace and headers/footers) when they are displayed in print layout view.

[*Example*: Consider the images below illustrating two pages in a WordprocessingML document:

If document shall automatically have whitespace between pages removed when it is displayed, that requirement would be specified using the following WordprocessingML in the document settings:

```
<w:doNotDisplayPageBoundaries w:val="true" />
```

The resulting output might look like the following:



The doNotDisplayPageBoundaries element has its val attribute equal to `true`, therefore the document is automatically displayed with whitespace between text extents on following pages compressed, allowing the pages to be viewed more easily. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.35 doNotEmbedSmartTags (Remove Smart Tags When Saving)

This element specifies if any smart tags specified using the smartTag element shall be removed from the contents of this document before it is resaved. This setting shall also prevent the addition of new smart tags to the content of the document.

If this element is omitted, then smart tags shall not be removed from the file when it is saved.

[*Example*: Consider a WordprocessingML document which should never be saved with smartTag elements in its contents. This requirement is specified using the following WordprocessingML fragment in the document settings:

```
<w:doNotEmbedSmartTags w:val=”true”/>
```

The doNotEmbedSmartTags element's val attribute has a value of `true` specifying that smart tags shall never be saved in the contents of this document. For example, if a run formerly looked like this:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">Hello</w:t>
  </w:r>
  <w:smartTag … >
    <w:r>
      <w:t>world<w:t>
    </w:r>
  </w:smartTag>
</w:p>
```

The presence of this element specifies that the SmartTag element shall be removed, and applications may then choose to combine duplicated runs as desired. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.36 doNotHyphenateCaps (Do Not Hyphenate Words in ALL CAPITAL LETTERS)

This element specifies whether or not words comprised of all capital letters shall be hyphenated within a given document when automatic hyphenation is specified via the autoHyphenation element (§2.15.1.10).

If this element is omitted, then words in ALL CAPITAL LETTERS shall be hyphenated when the document is hyphenated.

[*Example*: Consider a document which is automatically hyphenated containing the following paragraph of content:

THIS IS A HYPHENATION EXAMPLE. THIS IS A SHORT HYPHENATION EXAMPLE. THIS IS A SHORT HYPHE-
NATION EXAMPLE.

If words in ALL CAPITAL LETTERS shall not be hyphenated, this requirement would be specified by adding the following WordprocessingML to the document settings part:

```
<w:doNotHyphenateCaps w:val="true"/>
```

The resulting content would not be hyphenated:

THIS IS A HYPHENATION EXAMPLE. THIS IS A SHORT HYPHENATION EXAMPLE. THIS IS A SHORT
HYPHENATION EXAMPLE.

The doNotHyphenateCaps element val set to `true`, specifying that the first line of text to end with the word SHORT as the word HYPHENATION had to be moved to the second line since it could not fit in its entirety on the first line.

Conversely, setting the doNotHyphenateCaps element val set to `off` (the default) caused the first line of text to contain a hyphenated portion of the word HYPHENATION as hyphenation of words comprised of all capital letters is permitted. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>`

Attributes	Description
	The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.37 doNotIncludeSubdocsInStats (Do Not Include Content in Text Boxes, Footnotes, and Endnotes in Document Statistics)

This element specifies if document content contained in text boxes, footnotes, and endnotes shall be excluded when an application calculates a given document's statistics when these values are calculated and/or displayed by an application.

[*Note*: Some examples of document statistics that an application may chose to calculate are: number of words, number of characters, number of paragraphs, number of pages, number of lines, and so on. *end note*]

[*Example*: Consider a WordprocessingML that specifies that it shall not include these document stories when its contents are used to calculate document statistics. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:doNotIncludeSubdocsInStats w:val="true"/>
```

The doNotIncludeSubdocsInStats element's val attribute has a value of true specifying that only the contents of the main document story should be used when calculating document statistics. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/>

Attributes	Description
	The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.38 doNotShadeFormData (Do Not Show Visual Indicator For Form Fields)

This element specifies whether a visual cue should be displayed around form fields contained in a WordprocessingML document specified via the FORMTEXT, FORMCHECKBOX, or FORMDROPDOWN fields.

If this element is not present in a WordprocessingML document visual cues should be displayed on form fields contained in the document.

[*Example*: Consider a WordprocessingML document which should no visual indication of form fields. This requirement would be specified using the following WordprocessingML:

```
<w:doNotShadeFormData w:val="true"/>
```

The doNotShadeFormData element's val attribute has a value of true specifying the display of any visual indication of form fields shall be suppressed for this document. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.39 doNotTrackFormatting (Do Not Track Formatting Revisions When Tracking Revisions)

This element specifies that applications shall not track revisions made to the formatting of this WordprocessingML document when the trackRevisions element (§2.15.1.90) is turned on.

If this element is omitted, then revisions to formatting shall be generated by changes to the contents of this document when the trackRevisions element is turned on.

[*Example*: Consider a WordprocessingML document containing the text run `Example` that shall have revisions tracked. Example WordprocessingML from Document 1 is given below:

```
<w:document>
  <w:body>
    <w:p>
      <w:r>
        <w:t>Example</w:t>
      </w:r>
    </w:p>
  </w:body>
</w:document>
```

If the word `text` was added to the end of this document and bolded, the resulting WordprocessingML would be output as follows:

```
<w:document>
  <w:body>
    <w:p>
      <w:r>
        <w:t>Example</w:t>
      </w:r>
      <w:ins … >
        <w:r>
          <w:rPr>
            <w:b/>
            <w:rPrChange … >
              <w:rPr/>
            <w:rPrChange>
          </w:rPr>
          <w:t>text</w:t>
        </w:r>
      </w:ins>
    </w:p>
  </w:body>
</w:document>
```

If changes to formatting were turned off using the following WordprocessingML syntax in the document settings:

```
<w:settings>
  <w:trackRevisions w:val="true" />
  <w:doNotTrackFormatting w:val="true" />
  …
</w:settings>
```

The same revision (the word `text` was added to the end of this document and bolded) would result in the following markup:

```
<w:document>
  <w:body>
    <w:p>
      <w:r>
        <w:t>Example</w:t>
      </w:r>
      <w:ins … >
        <w:r>
          <w:rPr>
            <w:b/>
          </w:rPr>
          <w:t>text</w:t>
        </w:r>
      </w:ins>
    </w:p>
  </w:body>
</w:document>
```

The doNotTrackFormatting element's val attribute was set to `true`, therefore the changes to the formatting of the document were not tracked as revisions in the document's WordprocessingML. Specifically, applying bold formatting to the text was not tracked as a revision with the rPrChange (§2.13.5.32) element. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.40 doNotTrackMoves (Do Not Use Move Syntax When Tracking Revisions)

This element specifies that applications shall not track revisions made to this WordprocessingML document as moves when the trackRevisions element (§2.15.1.90) is turned on, even when that syntax is appropriate. Instead, applications should use a standard insertion and deletion annotation syntax. Existing moves shall not be modified. [*Rationale*: This element is provided to enable interoperability with earlier word processing applications which do not understand moves. *end rationale*]

If this element is omitted, then move annotations may be generated by changes to the contents of this document when the trackRevisions element is turned on as appropriate.

[*Example*: Consider a WordprocessingML that specifies that it shall not have additional moves added to its contents. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:doNotTrackMoves w:val="true"/>
```

The doNotTrackMoves element's val attribute has a value of `true` specifying that insertion/deletion annotations shall be used rather than moves when revisions are tracked in this document. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.41 doNotUseMarginsForDrawingGridOrigin (Do Not Use Margins for Drawing Grid Origin)

This element specifies that the top-left corner of the page shall not be used as the origin for the drawing grid. The *drawing grid* is a virtual grid which may be used by applications to specify where drawing objects shall be positioned on a page when inserted (i.e. to ensure objects are aligned, etc.). If this element is present the grid shall start at the top-left edge of the page and not the text extents.

If this element is omitted, then the gridlines shall start at the topmost edge of the text extents.

[*Example*: Consider a WordprocessingML document whose drawing grid shall begin at the top left edge of the page. This requirement would be specified using the following WordprocessingML markup in the document settings:

```
<w:doNotUseMarginsForDrawingGridOrigin w:val="true" />
```

The doNotUseMarginsForDrawingGridOrigin element's val attribute is equal to `true` specifying that the document's drawing grid shall begin from the top left corner of the page, rather than the top left corner of the text extents. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.42 doNotValidateAgainstSchema (Do Not Validate Custom XML Markup Against Schemas)

This element specifies that applications shall not validate the custom XML markup in this document against the applicable custom XML schema(s), even when those schemas are available. The application should silently behave as if it was unable to provide this functionality.

If this element is omitted, then applications which support this functionality should attempt to validate the custom XML contents against any available related custom XML schema(s).

[*Example*: Consider a WordprocessingML document which should not have its custom XML content validated even by applications which support this operation. This requirement is specified using the following WordprocessingML in the document settings:

```
<w:doNotValidateAgainstSchema w:val="true" />
```

The doNotValidateAgainstSchema element's val attribute has a value of `true` specifying that the custom XML markup in this document shall not be validated. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.43 drawingGridHorizontalOrigin (Drawing Grid Horizontal Origin Point)

This element specifies the distance from of the left edge of the page which shall be used as the origin for the horizontal gridlines used by the drawing grid. The *drawing grid* is a virtual grid which may be used by applications to specify where drawing objects shall be positioned on a page when inserted (i.e. to ensure objects are aligned, etc.). Since the grid always covers the entire page when the doNotUseMarginsForDrawingGridOrigin element (§2.15.1.41) is specified, this element shall only affect the starting edge of the first horizontal gridline displayed (i.e. it only adjusts the grid by the modulus of the value against the width of one grid unit).

If this element is omitted, then the gridlines shall start at the leftmost edge of the page. If the doNotUseMarginsForDrawingGridOrigin element is not specified, then this element is ignored.

[*Example*: Consider a WordprocessingML document whose drawing grid shall begin three inches (4320 twentieths of a point) before the left edge of the page. This requirement would be specified using the following WordprocessingML markup in the document settings:

```
<w:settings>
  …
  <w:dontuseMarginsForDrawingGridOrigin w:val="true" />
  <w:drawingGridHorizontalOrigin w:val="4320" />
  …
</w:settings>
```

The drawingGridHorizontalOrigin element's val attribute is equal to `4320` specifying that the horizontal edge of the document's drawing grid shall begin three inches (4320 twentieths of a point) from the left edge of the page, since the dontUseMarginsForDrawingGridOrigin element's val attribute is equal to `true`. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (Measurement in Twentieths of a Point)	Specifies a positive measurement value, specified in twentieths of a point. This value is interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML element with a val attribute containing a positive measurement in twentieths of a point: `<w:… w:val="720" />`

Attributes	Description
	The val attribute has a value of 720, specifying that this measurement value is 720 twentieths of a point (0.5"). This value is interpreted by the parent element as needed. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

The following XML Schema fragment defines the contents of this element:

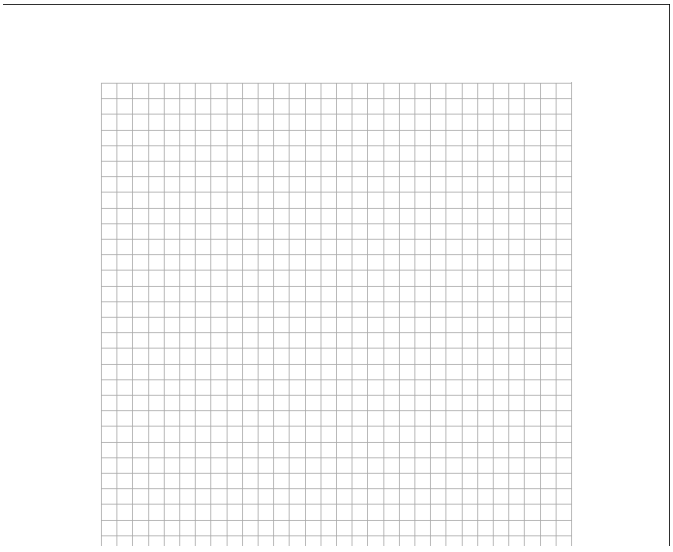
```
<complexType name="CT_TwipsMeasure">
   <attribute name="val" type="ST_TwipsMeasure" use="required"/>
</complexType>
```

2.15.1.44 drawingGridHorizontalSpacing (Drawing Grid Horizontal Grid Unit Size)

This element specifies the width of horizontal grid units in this document. The *drawing grid* is a grid which may be used by applications to help position floating objects in the document.

If this element is omitted, then each horizontal grid unit shall be 180 twentieths of a point (0.125") in width.

[*Example*: Consider the image below illustrating a WordprocessingML document in which all horizontal grid units are each 144 twentieths of a point wide (and all are showing):



If the gridlines in this document shall only be displayed for every half an inch, that requirement would be specified using the following WordprocessingML in the document settings:

```
<w:drawingGridHorizontalSpacing w:val="720" />
```

The resulting grid would look like the following:

The drawingGridHorizontalSpacing element has its val attribute equal to 720, therefore every horizontal gridline has a width of one half of an inch (720 twentieths of a point). *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (Measurement in Twentieths of a Point)	Specifies a positive measurement value, specified in twentieths of a point. This value is interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML element with a val attribute containing a positive measurement in twentieths of a point: `<w:… w:val="720" />` The val attribute has a value of 720, specifying that this measurement value is 720 twentieths of a point (0.5"). This value is interpreted by the parent element as needed. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TwipsMeasure">
   <attribute name="val" type="ST_TwipsMeasure" use="required"/>
</complexType>
```

2.15.1.45 drawingGridVerticalOrigin (Drawing Grid Vertical Origin Point)

This element specifies the distance from of the top edge of the page which shall be used as the origin for the vertical gridlines used by the drawing grid. The *drawing grid* is a virtual grid which may be used by applications to specify where drawing objects shall be positioned on a page when inserted (i.e. to ensure objects are aligned, etc.). Since the grid always covers the entire page when the doNotUseMarginsForDrawingGridOrigin element (§2.15.1.41) is specified, this element shall only affect the starting edge of the first vertical gridline displayed (i.e. it only adjusts the grid by the modulus of the value against the width of one grid unit).

If this element is omitted, then the gridlines shall start at the topmost edge of the page. If the doNotUseMarginsForDrawingGridOrigin element is not specified, then this element is ignored.

[*Example*: Consider a WordprocessingML document whose drawing grid shall begin one inch (1440 twentieths of a point) before the top edge of the page. This requirement would be specified using the following WordprocessingML markup in the document settings:

```
<w:settings>
  …
  <w:dontuseMarginsForDrawingGridOrigin w:val="true" />
  <w:drawingGridVerticallOrigin w:val="1440" />
  …
</w:settings>
```

The drawingGridVerticalOrigin element's val attribute is equal to `1440` specifying that the vertical edge of the document's drawing grid shall begin one inch (1440 twentieths of a point) from the top edge of the page, since the dontUseMarginsForDrawingGridOrigin element's val attribute is equal to `true`. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (Measurement in Twentieths of a Point)	Specifies a positive measurement value, specified in twentieths of a point. This value is interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML element with a val attribute containing a positive measurement in twentieths of a point: `<w:… w:val="720" />` The val attribute has a value of `720`, specifying that this measurement value is 720 twentieths of a point (0.5"). This value is interpreted by the parent element as needed. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type

Attributes	Description
	(§2.18.105).

The following XML Schema fragment defines the contents of this element:

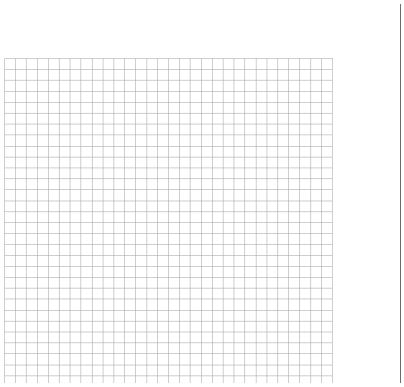
```
<complexType name="CT_TwipsMeasure">
   <attribute name="val" type="ST_TwipsMeasure" use="required"/>
</complexType>
```

2.15.1.46 drawingGridVerticalSpacing (Drawing Grid Vertical Grid Unit Size)

This element specifies the width of vertical grid units in this document. The *drawing grid* is a grid which may be used by applications to help position floating objects in the document.

If this element is omitted, then each vertical grid unit shall be 180 twentieths of a point (0.125") in width.

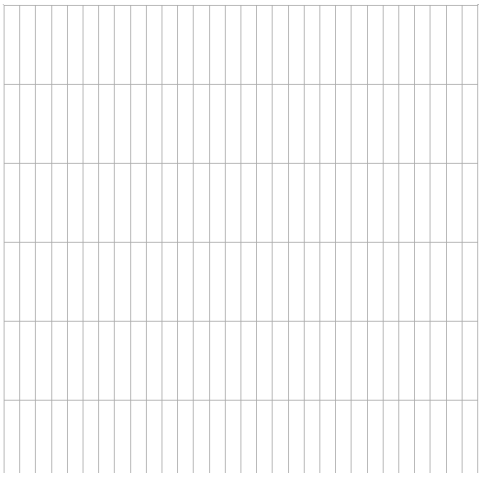
[*Example*: Consider the image below illustrating a WordprocessingML document in which all vertical grid units are each 144 twentieths of a point high (and all are showing):



If the vertical gridlines in this document shall only be displayed for every half an inch, that requirement would be specified using the following WordprocessingML in the document settings:

```
<w:drawingGridVerticalSpacing w:val="720" />
```

The resulting grid would look like the following:



The drawingGridVerticalSpacing element has its val attribute equal to 720, therefore every vertical gridline has a height of one half of an inch (720 twentieths of a point). *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (Measurement in Twentieths of a Point)	Specifies a positive measurement value, specified in twentieths of a point. This value is interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML element with a val attribute containing a positive measurement in twentieths of a point: `<w:… w:val="720" />` The val attribute has a value of 720, specifying that this measurement value is 720 twentieths of a point (0.5"). This value is interpreted by the parent element as needed. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TwipsMeasure">
   <attribute name="val" type="ST_TwipsMeasure" use="required"/>
</complexType>
```

2.15.1.47 forceUpgrade (Upgrade Document on Open)

This element specifies that the contents of this document may be upgraded and that the resulting document shall not have its functionality limited to only those functions compatible with earlier word processing applications. The only actions required as part of upgrading the document are:

- The removal of this and the uiCompat97to2003 element (§2.15.3.54). This is needed in order to prevent future applications from disabling functionality on the resulting 'upgraded' document. If an application does not know how to upgrade a document, this element and the uiCompat97to2003 element should be ignored and persisted.
- The removal of all compatibility options (§2.15.3.9) on the document which maintain compatibility with previous word processing applications. The compatibility settings which simply affect a given behavior shall not be turned off.

[*Note*: The remaining operations which shall be performed as part of upgrading the document are application-defined and outside the scope of this Office Open XML Standard. *end note*]

[*Example*: Consider a WordprocessingML document that specifies that it shall automatically be upgraded when it is opened by an application. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:forceUpgrade w:val="true"/>
```

The forceUpgrade element's val attribute has a value of `true` specifying that this document shall be upgraded by any application which supports this operation. *end example*]

Parent Elements
settings (§2.15.1.78)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

2.15.1.48 formsDesign (Structured Document Tag Placeholder Text Should be Resaved)

This element specifies that the document was last saved while the placeholder text of all structured document tags in this document were being edited. This means that the placeholder text currently displayed in all structured document tags which are displaying the showingPlcHdr element (§2.5.2.38) shall be committed to the corresponding glossary document entry as specified using the docPart element (§2.12.5) when this document is opened, in order to ensure that the most recent placeholder text is stored in the glossary document entry. If the current placeholder text cannot be saved as a glossary document entry, then it should be modified as needed before saving.

If this element is omitted, then the placeholder text in this document should not automatically be resaved when the document is opened.

[*Example*: Consider a WordprocessingML document that specifies that its placeholder text should be resaved to the glossary document when the file is opened. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:formsDesign w:val="true"/>
```

The formsDesign element's val attribute has a value of `true` specifying that this document should be resaved to its glossary document by any application which supports this operation. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.49 gutterAtTop (Position Gutter At Top of Page)

This element specifies that a given WordprocessingML document's gutter shall be positioned at the top of the document's pages when the document is displayed. A *gutter* is the white space formed by the inner margins of two pages facing one another; such as the white space between the text on pages of a book when the book is opened.

If this element is omitted, then the gutter shall not be positioned at the top of the page. If the mirrorMargins (§2.15.1.57), bookFoldPrinting (§2.15.1.11), bookFoldRevPrinting (§2.15.1.13), or printTwoOnOne (§2.15.1.64) elements are used within a given document, the gutterAtTop element shall not be used. Rather,

the gutter shall be positioned automatically as necessary to enable the printing and page layout capabilities of these settings.

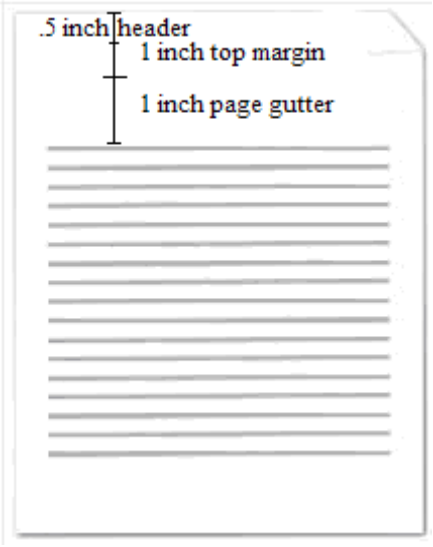
[*Example*: Consider a one page WordprocessingML document with a 1,440 twentieths of a point (one inch) top margin and gutter, and a 720 twentieths of a point (one half of an inch) header. Consider also, that the gutter shall exist at the top of the document's pages. This requirement is specified using the following WordprocessingML in the section properties:

```
<w:pgMar w:top="1440" ... w:header="720" ... w:gutter="1440" />
```

And the following WordprocessingML in the document settings:

```
<w:gutterAtTop w:val="true" />
```

The resulting document's pages would have the gutter positioned as follows:



The gutterAtTop element's val attribute is equal to on, specifying that the gutter shall appear at the top of each page. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted.

Attributes	Description
	A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.50 hdrShapeDefaults (Default Properties for VML Objects in Header and Footer)

This element specifies the default parameters for object using the ' (§6.1) inserted in the header and footer of a WordprocessingML document. The definition and semantics of these parameters is described in the VML - Office Drawing subclause (§6.2) of this Office Open XML Standard.

If this element is omitted, then no default properties are applied to VML objects in the header and footer of this document.

[*Example*: Consider a WordprocessingML document whose document settings contain the following markup:

```
<w:hdrShapeDefaults>
  <o:shapedefaults v:ext="edit" spidmax="2050" fillcolor="none [3207]"
strokecolor="none [3041]">
    <v:fill color="none [3207]" />
    <v:stroke color="none [3041]" weight="3pt" />
    <v:shadow on="t" type="perspective" color="none [1607]" opacity=".5"
offset="1pt" offset2="1pt" />
  </o:shapedefaults>
  <o:shapelayout v:ext="edit">
    <o:idmap v:ext="edit" data="2" />
  </o:shapelayout>
</w:hdrShapeDefaults>
```

The hdrShapeDefaults element specifies a set of shape defaults which shall be applied to the set of all shapes present in the header and footer of this document. *end example*]

Parent Elements
settings (§2.15.1.78)

Child Elements	Subclause
Any element from the urn:schemas-microsoft-com:office:office namespace	§6.2

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShapeDefaults">
   <choice maxOccurs="unbounded">
      <any processContents="lax" namespace="urn:schemas-microsoft-com:office:office" minOccurs="0"
        maxOccurs="unbounded"/>
   </choice>
</complexType>
```

2.15.1.51 hideGrammaticalErrors (Do Not Display Visual Indication of Grammatical Errors)

This element specifies whether a visual cue should be displayed around run content contained in a WordprocessingML document which has been flagged as a possible grammatical error using the proofErr element (§2.13.8.1) or via the application's own grammar engine.

If this element is not present in a WordprocessingML document, visual cues shall be displayed on content contained in a WordprocessingML document which is considered to contain grammatical errors.

[*Example*: Consider a WordprocessingML document which should show no visual indication of grammatical errors. This requirement would be specified using the following WordprocessingML:

```
<w:hideGrammaticalErrors w:val="true"/>
```

The hideGrammaticalErrors element's val attribute has a value of `true` specifying the display of any visual indication of grammatical errors shall be suppressed for this document. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.52 hideSpellingErrors (Do Not Display Visual Indication of Spelling Errors)

This element specifies whether a visual cue should be displayed around run content contained in a WordprocessingML document which has been flagged as a possible spelling error using the proofErr element (§2.13.8.1) or via the application's own spelling engine.

If this element is not present in a WordprocessingML document, visual cues shall be displayed on content contained in a WordprocessingML document which is considered to contain spelling errors.

[*Example*: Consider a WordprocessingML document which should show no visual indication of spelling errors. This requirement would be specified using the following WordprocessingML:

```
<w:hideSpellingErrors w:val="true"/>
```

The hideSpellingErrors element's val attribute has a value of `true` specifying the display of any visual indication of spelling errors shall be suppressed for this document. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

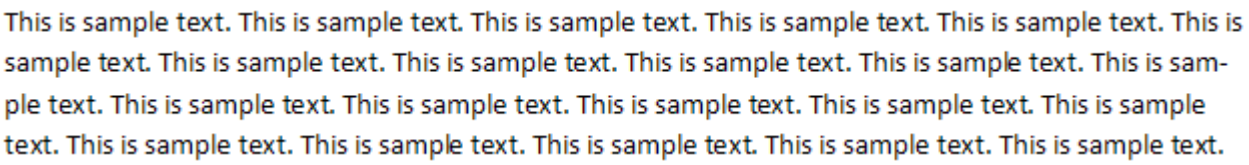
```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.53 hyphenationZone (Hyphenation Zone)

This element specifies the hyphenation zone which shall be used when automatically or manually hyphenating the contents of this document. The *hyphenation zone* is the amount of whitespace which may be left at the end of a line (or added to justified lines) before hyphenation should be attempted on the next word in the document (in order to reduce the amount of whitespace on the line). A smaller hyphenation zone should reduce the raggedness of the right edge of a given document's body text, as more words will be hyphenated. Conversely, a larger hyphenation zone should increase the raggedness of the right edge of a given document's text, as fewer words will be hyphenated.

If this element is omitted, then a default hyphenation zone of 360 twentieths of a point (0.25") shall be applied when performing hyphenation on this document.

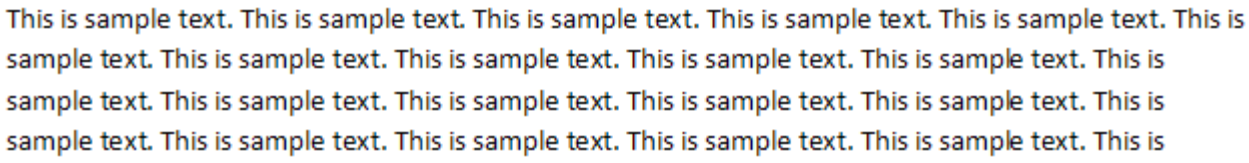
[*Example*: Consider the images below illustrating a paragraph of text in a WordprocessingML document which was automatically hyphenated with the default hyphenation zone:



If document shall instead be hyphenated using a hyphenation zone of one-half of an inch, that requirement would be specified using the following WordprocessingML in the document settings:

```
<w:hyphenationZone w:val="720" />
```

The resulting output would only hyphenate words when the remaining line whitespace was less than half an inch, and would look like the following:



The hyphenationZone element has its val attribute equal to 720, therefore lines in the document are hyphenated when the remaining line spacing is less than 72o twentieths of a point (0.5"), resulting in fewer hyphens. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (Measurement in Twentieths of a Point)	Specifies a positive measurement value, specified in twentieths of a point. This value is interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML element with a val attribute containing a positive measurement in twentieths of a point: `<w:… w:val="720" />` The val attribute has a value of 720, specifying that this measurement value is 720 twentieths of a point (0.5"). This value is interpreted by the parent element as needed. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TwipsMeasure">
   <attribute name="val" type="ST_TwipsMeasure" use="required"/>
</complexType>
```

2.15.1.54 ignoreMixedContent (Ignore Mixed Content When Validating Custom XML Markup)

This element specifies that applications should ignore all text content which is not contained within a leaf custom XML markup element when validating the contents of the custom XML markup in this document against one or more attached custom XML schema(s). A *leaf element* is a custom XML element which has no child custom XML elements (it is a leaf in the custom XML tree).

If this element is omitted, then text content in leaf elements shall not be ignored when validating the custom XML markup against one or more custom XML schema(s).

[*Example*: Consider a WordprocessingML document which should not have its custom XML content validated even by applications which support this operation. This requirement is specified using the following WordprocessingML in the document settings:

```
<w:doNotValidateAgainstSchema w:val="true" />
```

The doNotValidateAgainstSchema element's val attribute has a value of `true` specifying that the custom XML markup in this document shall not be validated. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element.

Attributes	Description
	A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.55 linkStyles (Automatically Update Styles From Document Template)

This element specifies that styles in the given document shall be updated to match the styles in the attached template specified using the attachedTemplate element (§2.15.1.6) when the document is opened by a hosting application. This setting enables the styles contained in documents with attached templates to stay synchronized with the styles used in the attached template.

If this element is omitted, then styles shall not be updated based on the document template regardless of its availability. If the attached template cannot be located or is not a valid file, then this setting should be silently ignored.

[*Example*: Consider a WordprocessingML document which should always update its styles with those defined in the document's attached template. This requirement would be specified using the following WordprocessingML in the document settings:

```
<w:settings>
  <w:linkStyles w:val="true" />
  <w:attachedTemplate r:id="rId10" />
  …
</w:settings>
```

The linkStyles element has a val attribute value of true, specifying that applications should attempt to locate the document template referenced by the relationship specified in the attachedTemplate element and update the document's styles with the styles from that template. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.56 listSeparator (List Separator for Field Code Evaluation)

This element specifies the character that shall be interpreted as a list item separator when evaluating the contents of all fields in the current document.

[*Rationale*: When evaluating field instructions based on the contents of the current document, it is necessary to know the character which shall be treated as the list separator in order to prevent changes to the calculation of the same field instructions based on the current user's locale. This element stores the list separator which shall be used to evaluate fields in the contents of this document, irrespective of the locale of the application loading the file. *end rationale*]

If this element is omitted, the application shall use the default list separator of its current locale setting to evaluate field instructions. If this element's attribute value is more than a single character, then the document is non-conformant.

[*Example*: Consider a WordprocessingML document which should use the semicolon character as the list separator for all field instructions. This requirement is specified using the following WordprocessingML in the document settings:

```
<w:listSeparator w:val=";" />
```

The listSeparator element's val attribute has a value of `;` specifying that the semicolon character shall be interpreted as a list item separator.

For instance, the string `10;20,5` would be interpreted as having two values - `10` and `20,5`. If the listSeparator was a comma, the same string would be interpreted as `10;20` and `5`. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.15.1.57 mirrorMargins (Mirror Page Margins)

This element specifies that the left and right margins defined in the section properties shall be swapped on facing pages.

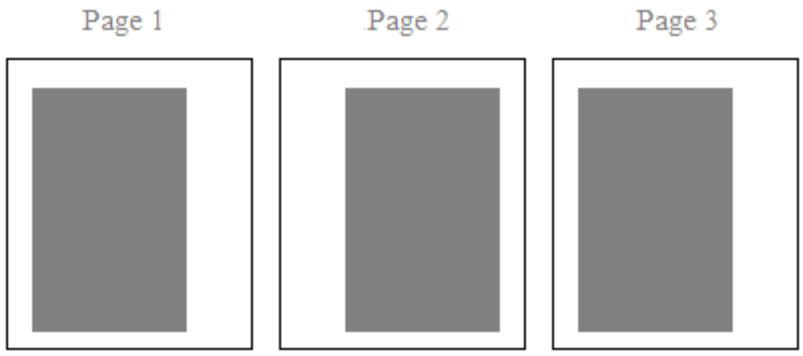
[*Guidance*: This setting is generally used when printing on both sides of pages and binding them like a book. *end guidance*]

[*Example*: Consider a graphical representation (below) of a three page WordprocessingML document with a left margin of 1" and a right margin of 2".

If the mirrorMargins element is present in the document settings with its val attribute equal to `true`, as follows:

```
<w:mirrorMargins w:val="true" />
```

The resulting pages will have mirrored margins as follows (un this representation, the gray rectangles representing the text extents on each page):



end example]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.58 noLineBreaksAfter (Custom Set of Characters Which Cannot End a Line)

This element specifies the set of characters which shall be restricted from ending a line for runs of text which shall be subject to custom line breaking logic using the kinsoku element (§2.3.1.16) when the contents of the document are displayed. This constraint shall only apply to text which has been flagged in the language of this rule via the lang element (§2.3.2.18) or automatic detection methods outside the scope of this Office Open XML Standard.

If this element is omitted, then no custom set of characters shall be used to restrict the characters which may end a line when using the kinsoku element.

[*Example*: Consider a paragraph of WordprocessingML text displayed as follows, with the dollar symbol $ was flagged as Japanese content using the following WordprocessingML in the run properties:

```
<w:r>
  <w:rPr>
    <w:lang w:eastAsia="ja-JP" />
  </w:rPr>
  <w:t>$</w:t>
</w:r>
```

This is a sample line of text. This is a sample line of text. This is a sample line of text. This is a $ sample line of text.

This text is displayed and the resulting first line ends with the dollar sign symbol. If this character shall not be used to end a line, that requirement would be specified as follows in the document settings:

```
<w:noLineBreaksAfter w:lang="ja-JP" w:val="$" />
```

The noLineBreaksAfter element's val attribute has a value of `ja-JP`, specifying that all dollar signs in this document which are marked as Japanese text shall not be allowed to end a line. This means that the dollar sign character must therefore be moved to the next line as it can no longer be the last character on a line:

This is a sample line of text. This is a sample line of text. This is a sample line of text. This is a $ sample line of text.

end example]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
lang (Language For Which Custom Line Breaking Rule Applies)	Specifies the language of text for which the parent custom line breaking rule shall be applied. Applications supporting this functionality shall support custom line breaking for the following four languages: • Chinese (Traditional) • Chinese (Simplified) • Japanese • Korean Applications may also support custom line breaking rules for other languages, but this is not required. [*Example*: Consider a WordprocessingML document which shall have a custom line breaking rule for Japanese. That requirement would be specified as follows in the document settings: `<w:… w:lang="ja-JP" w:val="$" />` The lang attribute has a value of `ja-JP`, specifying that the rules shall be applied to Japanese text. *end example*] The possible values for this attribute are defined by the ST_Lang simple type (§2.18.51).
val (Characters For Custom Line Breaking Rule)	Specifies the set of characters which shall be included in the custom line breaking rule. [*Example*: Consider a WordprocessingML document which shall have a custom line breaking rule for Japanese. That requirement would be specified as follows in the document settings: `<w:… w:lang="ja-JP" w:val="$" />` The val attribute has a value of `$`, specifying that the dollar sign character is the only restricted character for Japanese text. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Kinsoku">
   <attribute name="lang" type="ST_Lang" use="required"/>
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.15.1.59 noLineBreaksBefore (Custom Set Of Characters Which Cannot Begin A Line)

This element specifies the set of characters which shall be restricted from beginning a new line for runs of text which shall be subject to custom line breaking logic using the kinsoku element (§2.3.1.16) when the contents of the document are displayed. This constraint shall only apply to text which has been flagged in the language of

this rule via the lang element (§2.3.2.18) or automatic detection methods outside the scope of this Office Open XML Standard.

If this element is omitted, then no custom set of characters shall be used to restrict the characters which may end a line when using the kinsoku element.

[*Example*: Consider a paragraph of WordprocessingML text displayed as follows, with the dollar symbol $ was flagged as Korean content using the following WordprocessingML in the run properties:

```
<w:r>
  <w:rPr>
    <w:lang w:eastAsia="ko-KR" />
  </w:rPr>
  <w:t>$</w:t>
</w:r>
```

This is a sample line of text. This is a sample line of text. This is a sample line of text. This is a
$ sample line of text.

This text is displayed and the resulting second line begins with the dollar sign symbol. If this character shall not be used to begin a line, that requirement would be specified as follows in the document settings:

```
<w:noLineBreaksBefore w:lang="ko-KR" w:val="$" />
```

The noLineBreaksBefore element's val attribute has a value of ko-KR, specifying that all dollar signs in this document which are marked as Korean text shall not be allowed to begin a line. This means that the previous word character must therefore be moved to the next line as the dollar sign can no longer be the first character on a line:

This is a sample line of text. This is a sample line of text. This is a sample line of text. This is
a $ sample line of text.

end example]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
lang (Language For Which Custom Line Breaking Rule Applies)	Specifies the language of text for which the parent custom line breaking rule shall be applied. Applications supporting this functionality shall support custom line breaking for the following four languages: • Chinese (Traditional) • Chinese (Simplified) • Japanese

Attributes	Description
	• Korean Applications may also support custom line breaking rules for other languages, but this is not required. [*Example*: Consider a WordprocessingML document which shall have a custom line breaking rule for Japanese. That requirement would be specified as follows in the document settings: `<w:… w:lang="ja-JP" w:val="$" />` The lang attribute has a value of `ja-JP`, specifying that the rules shall be applied to Japanese text. *end example*] The possible values for this attribute are defined by the ST_Lang simple type (§2.18.51).
val (Characters For Custom Line Breaking Rule)	Specifies the set of characters which shall be included in the custom line breaking rule. [*Example*: Consider a WordprocessingML document which shall have a custom line breaking rule for Japanese. That requirement would be specified as follows in the document settings: `<w:… w:lang="ja-JP" w:val="$" />` The val attribute has a value of `$`, specifying that the dollar sign character is the only restricted character for Japanese text. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Kinsoku">
   <attribute name="lang" type="ST_Lang" use="required"/>
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.15.1.60 noPunctuationKerning (Never Kern Punctuation Characters)

This element specifies that punctuation characters shall not be kerned in the current document when kerning is enabled on a run using the kern element (§2.3.2.17). *Kerning* refers to a process by which a hosting application shall reduce the spacing of adjacent characters and/or punctuation to improve the visual appearance of text. Well kerned text has a similar amount of blank space between each pair of characters and/or each set of a character and punctuation symbol. When kerning is enabled, Latin text shall always be kerned, and this option shall control whether punctuation characters are also kerned.

If this element is omitted, then punctuation characters shall be kerned when kerning is enabled on a given run.

[*Example*: Consider a WordprocessingML document that shall not kern punctuation even when kerning is enabled on a given run. This requirement is specified using the following WordprocessingML in the document settings:

```
<w:noPunctuationKerning w:val="true" />
```

The noPunctuationKerning element's val attribute has a value of `true`, specifying that punctuation characters shall not be kerned in this document. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

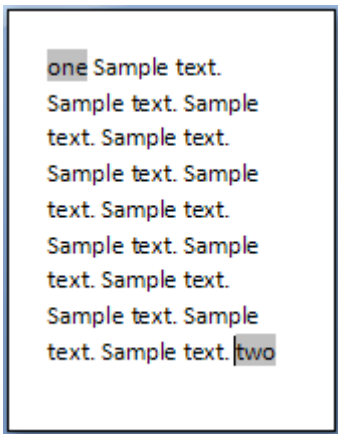
2.15.1.61 printFormsData (Only Print Form Field Content)

This element specifies that printing the contents of this document shall only print the contents of WordprocessingML form fields defined using the `FORMTEXT`, `FORMCHECKBOX`, and `FORMDROPDOWN` field codes in their current locations on the page - all other document contents shall be suppressed.

[*Rationale*: This setting is typically used to allow duplication of paper forms in electronic WordprocessingML document form, allowing the resulting online document to be printed into the correct locations on the existing paper form. *end rationale*]

If this element is omitted, then the contents of the entire document (not just form fields) should be printed according to the normal print settings.

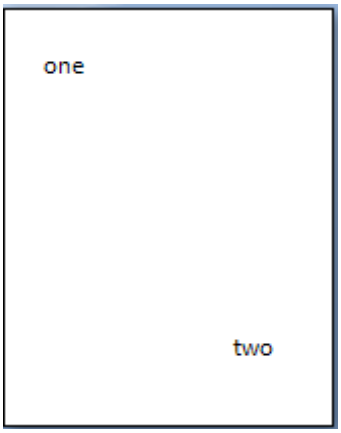
[*Example*: Consider a WordprocessingML document which has form fields in the top right and bottom left corners of the first page, as follows (with the text box form fields shaded in grey):



If the only content which shall be printed on the page are the form fields' contents, this requirement is specified using the following WordprocessingML in the document settings:

```
<w:printFormsData w:val="true" />
```

The printFormsData element's val attribute as a value of `true`, specifying that only form field data shall be printed, resulting in output as follows when printed:



end example]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the

Attributes	Description
	default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.62 printFractionalCharacterWidth (Print Fractional Character Widths)

This element specifies the contents of this document shall be printed with fractional character widths. *Fractional character widths* exist when the spacing between characters is not constant (i.e. a proportional font face is used).

[*Note*: Fractional character widths are generally used in conjunction with large font sizes to prevent characters from running together or having too much space between one another. *end note*]

[*Example*: Consider a WordprocessingML document which should be printed using fractional character widths as needed. This requirement is specified using the following WordprocessingML markup in the document settings:

```
<w:printFractionalCharacterWidth w:val="true"/>
```

The printFractionalCharacterWidth element's val attribute is equal to `true`, specifying that fractional character widths may be used as necessary. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted.

Attributes	Description
	A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.63 printPostScriptOverText (Print PostScript Codes With Document Text)

This element specifies that the PostScript codes specified in WordprocessingML documents containing `PRINT` fields shall be included in foreground (on the same Z-order as text) with the data printed in the contents of a given WordprocessingML document.

[*Note*: This setting is maintained to ensure compatibility of legacy word processing documents. The `PRINT` field should not be used in lieu of newer technologies in this Office Open XML Standard. *end note*]

If this element is omitted, then the contents of `PRINT` fields shall be printed behind text (i.e. in the background).

[*Example*: Consider a WordprocessingML document containing `PRINT` fields whose PostScript code shall be printed in the foreground of the WordprocessingML document. This requirement is specified using the following WordprocessingML in the document settings:

```
<w:printPostScriptOverText w:val="true"/>
```

The printPostScriptOverText element's val attribute is equal to `true` specifying that the PostScript codes shall be treated as results for the main text level of the document (i.e. not behind that text). *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but

Attributes	Description
	this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.64 printTwoOnOne (Print Two Pages Per Sheet)

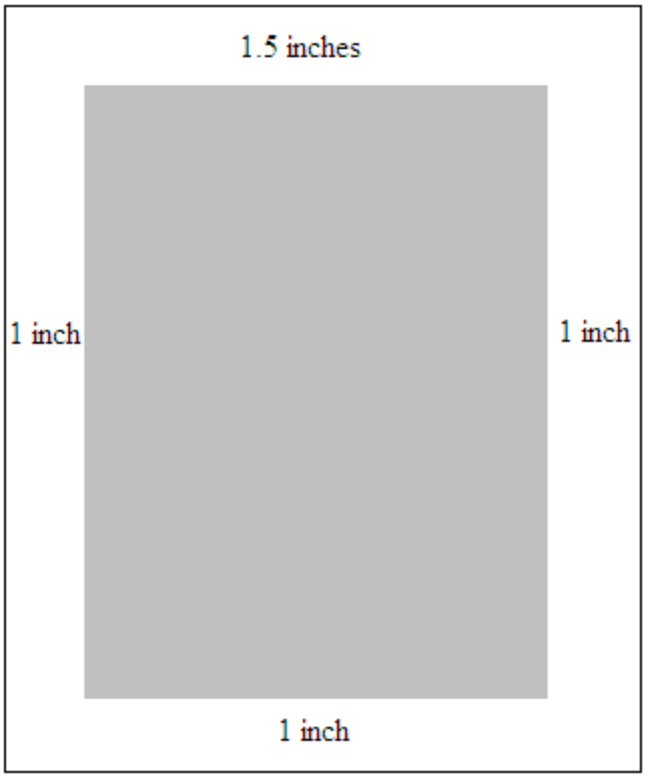
This element specifies whether two pages should be printed on one sheet of paper when this document is printed. Specifically, this element specifies that each page displayed for the contents in a given WordprocessingML document should be the page size specified in the section settings divided in half with two top margins originating from the bisector of the page, and bottom margins instantiated at the top and bottom of each page.

If this element is omitted, then pages should be displayed and printed as one per sheet.

[*Example*: Consider a one section document with a 2,160 twentieths of a point (one and a half inch) top margin, and 1,440 twentieths of a point (one inch) bottom, right, and left margins surrounding the document editing canvas (represented by the gray shaded area in diagrams below). This page setup is represented in WordprocessingML using the following fragment:

```
<w:pgMar w:top="2160" w:right="1440" w:bottom="1440" w:left="1440" />
```

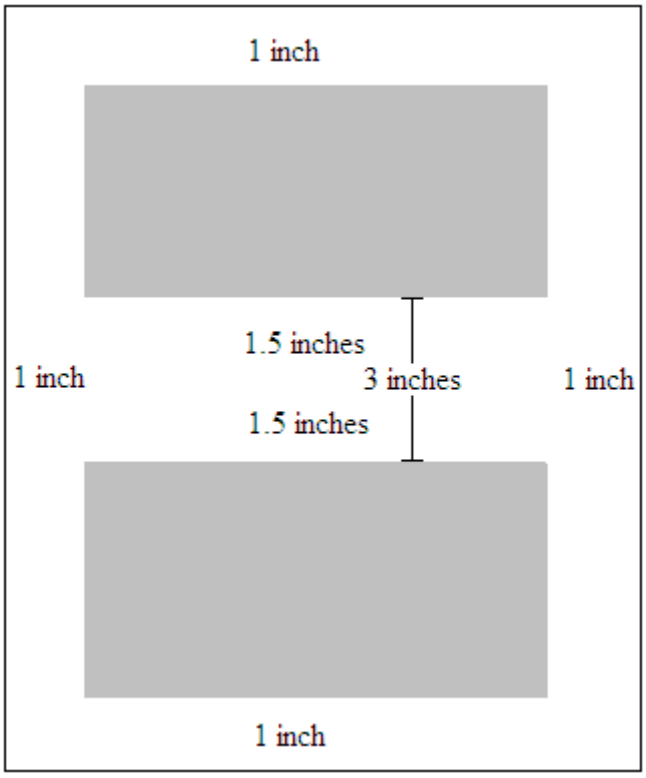
The resulting printed pages would appear as follows:



If a document should be displayed and printed as though two pages were printed on a single sheeting, this requirement would be specified using the following WordprocessingML:

```
<w:printTwoOnOne w:val="true" />
```

The printTwoOnOne element's val attribute is equal to `true` specifying that pages should be printed two to a sheet, resulting in the following printout given these page margins:



end example]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.65 proofState (Spelling and Grammatical Checking State)

This element specifies if the grammar and spell checking engines of the last application to process this document completed checking the grammar and spelling of a the document before the document was last saved. Applications which modify the document contents without checking spelling or grammar should reset these states as needed.

[*Note*: If this element specifies that an application's grammar and spell checking engines completed checking the grammar and spelling of the document when the document was last saved, then subsequent applications may elect to not run their grammar and spell checking engines when the given WordprocessingML document is loaded.

This may increase the speed with which the hosting application loads the file, and does not compromise the state of the grammar or spell checking of the document, as all errors have already been found and flagged with the proofErr element (§2.13.8.1) as the document has not been edited, only loaded, since it was last saved. *end note*]

[*Example*: Consider a WordprocessingML document that is saved by a hosting application whose spelling and grammar checking engines have completed checking grammar and spelling in the given WordprocessingML document. This state is specified using the following WordprocessingML in the document settings:

```
<w:proofState w:spelling="clean" w:grammar="clean" />
```

The proofState element's attributes spelling and grammar attribute both have the value `clean` specifying that the hosting application's grammar and spell checking engines completed checking both the grammar and spelling of the given document when it was last saved. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
grammar (Grammatical Checking State)	Specifies if an application's grammar checking engine completed checking the grammatical content of the document when it was last saved. If this attribute is omitted, then its value is assumed to be `dirty` (not complete). [*Example*: Consider a WordprocessingML document saved by a hosting application whose spelling and grammar checking engines have completed checking grammar and spelling in the given WordprocessingML document. This state is specified using the following WordprocessingML in the document settings:

Attributes	Description
	<w:proofState w:spelling="clean" w:grammar="clean" /> The grammar attribute has the value clean specifying that the hosting application's grammar checking engine completed checking the grammar of the given document when it was last saved. *end example*] The possible values for this attribute are defined by the ST_Proof simple type (§2.18.76).
spelling (Spell Checking State)	Specifies if an application's spell checking engine completed checking the spelling of the document when it was last saved. If this attribute is omitted, then its value is assumed to be dirty (not complete). [*Example*: Consider a WordprocessingML document saved by a hosting application whose spelling and grammar checking engines have completed checking grammar and spelling in the given WordprocessingML document. This state is specified using the following WordprocessingML in the document settings: <w:proofState w:spelling="clean" w:grammar="clean" /> The spelling attribute has the value clean specifying that the hosting application's spell checking engine completed checking the spelling of the given document when it was last saved. *end example*] The possible values for this attribute are defined by the ST_Proof simple type (§2.18.76).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Proof">
   <attribute name="spelling" type="ST_Proof" use="optional"/>
   <attribute name="grammar" type="ST_Proof" use="optional"/>
</complexType>
```

2.15.1.66 readModeInkLockDown (Freeze Document Layout)

This element specifies the exact set of page and text sizing parameters which shall be used to display the contents of a WordprocessingML document. [*Rationale*: This setting is typically used for documents that have been annotated using ink. This setting freezes the document's presentation such that the ink annotations shall exist at the same position of the WordprocessingML document irrespective of the monitor on which the WordprocessingML document is rendered. *end rationale*]

This element shall only affect the display of WordprocessingML documents as follows:

- When the actualPage attribute is specified with a value of true, the given WordprocessingML document's pages shall be rendered as they would normally be displayed. The resulting pages may have their magnification setting changed as desired. All other attributes shall be ignored.

- When the actualPage attribute is specified with a value of `false`, the given WordprocessingML document's pages shall be rendered as *virtual pages* when loaded by a conforming hosting application irrespective of the given WordprocessingML document's view (§2.15.1.93). *Virtual pages* are pages with no correlation with the printed layout of a given WordprocessingML document that have been scaled by a conforming hosting application to improve the readability of a given WordprocessingML document when it is displayed. Specifically, the w and h attributes specify the width and height of the virtual pages, and the fontSz attribute specifies the scaling to be applied to text within the given WordprocessingML document.

[*Example*: Consider a WordprocessingML document that shall be displayed using virtual pages when its contents are displayed. This state is specified using the following WordprocessingML in the document settings:

```
<w:readModeInkLockDown w:w="692" w:h="986" w:fontSz="95" w:actualPg="0"/>
```

The readModeInkLockDown element has w and h attribute values which specify the width and height of the virtual pages to be used to render the given WordprocessingML document. Finally, the fontSz attribute specifies the scaling to be applied to text within the given WordprocessingML document. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
actualPg (Use Actual Pages, Not Virtual Pages)	Specifies if applications shall render this WordprocessingML document with actual pages, not virtual pages. A*ctual pages* are pages rendered as they will be printed. A value of `true` specifies that the given WordprocessingML document's pages will be rendered as they are printed, and the w, h, and fontSz attributes shall be ignored. A value of `false` specifies that the given WordprocessingML document's pages shall be rendered as virtual pages using the other attributes on this element. [*Example*: Consider a WordprocessingML document that shall be displayed using virtual pages. This state is specified using the following WordprocessingML in the document settings: `<w:readModeInkLockDown w:w="692" w:h="986" w:fontSz="95" w:actualPg="0" />` The actualPage attribute is equal to `0` specifying that the given WordprocessingML document shall be rendered by conforming hosting applications using virtual pages. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
fontSz (Font Size Scaling)	Specifies the percentage that text in a given WordprocessingML document shall be scaled by before it is displayed on a virtual page. The attribute's value stores the percentage

Attributes	Description
	specified as an integer who units correspond to the percentage that text runs shall be scaled to [*Example*: 200 means a scale to 200% *end example*]. This attribute shall only be used if the actualPage attribute equals `off`, `0`, or `false`. [*Example*: Consider a WordprocessingML document that shall be displayed using virtual pages. This state is specified using the following WordprocessingML in the document settings: `<w:readModeInkLockDown w:w="692" w:h="986" w:fontSz="95" w:actualPg="0" />` The fontSz attribute is equal to 95 specifying that the text in the WordprocessingML document shall be displayed at 95% of its normal size when it is displayed on a virtual page. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
h (Virtual Page Height)	Specifies the height of the virtual pages which shall be used in this document. This value is specified in pixels. This attribute shall only be used if the actualPage attribute equals `off`, `0`, or `false`. [*Example*: Consider a WordprocessingML document that shall be displayed using virtual pages. This state is specified using the following WordprocessingML in the document settings: `<w:readModeInkLockDown w:w="692" w:h="986" w:fontSz="95" w:actualPg="0" />` The h attribute is equal to 986 specifying that virtual pages in this document shall be 986 pixels high. *end example*] The possible values for this attribute are defined by the ST_PixelsMeasure simple type (§2.18.74).
w (Virtual Page Width)	Specifies the width of the virtual pages which shall be used in this document. This value is specified in pixels. This attribute shall only be used if the actualPage attribute equals `off`, `0`, or `false`. [*Example*: Consider a WordprocessingML document that shall be displayed using virtual pages. This state is specified using the following WordprocessingML in the document settings: `<w:readModeInkLockDown w:w="692" w:h="986" w:fontSz="95" w:actualPg="0" />`

Attributes	Description
	The w attribute is equal to 692 specifying that virtual pages in this document shall be 692 pixels wide. *end example*] The possible values for this attribute are defined by the ST_PixelsMeasure simple type (§2.18.74).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ReadingModeInkLockDown">
   <attribute name="actualPg" type="ST_OnOff" use="required"/>
   <attribute name="w" type="ST_PixelsMeasure" use="required"/>
   <attribute name="h" type="ST_PixelsMeasure" use="required"/>
   <attribute name="fontSz" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.15.1.67 removeDateAndTime (Remove Date and Time from Annotations)

This element specifies that the date and time information shall be removed from all annotations which are present in the current document when it is saved. Annotations store this information in the date attribute on the annotation's XML element.

If this element is omitted, then date information shall not be removed when the document is saved. If the removePersonalInformation element is not turned on, then this setting shall be ignored.

[*Example*: Consider a WordprocessingML document that shall not save date and time information on annotations in the document content. This state is specified using the following WordprocessingML in the document settings:

```
<w:settings>
  …
  <w:removePersonalInformation w:val="true" />
  <w:removeDateAndTime w:val="true" />
  …
</w:settings>
```

The removeDateAndTime element's val attribute has a value of `true` specifying that all annotations in the document shall have and date and time information removed before they are saved by omitting their date attributes. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element.

Attributes	Description
	A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.68 removePersonalInformation (Remove Personal Information from Document Properties)

This element specifies that hosting applications shall remove all personal information of document authors upon saving a given WordprocessingML document. The definition and extent of personal information is not defined by this Office Open XML Standard.

If this element is omitted, then personal information shall not be removed when the document is saved.

[*Example*: Consider a WordprocessingML document that shall not save personal information in the document. This state is specified using the following WordprocessingML in the document settings:

```
<w:removePersonalInformation w:val="true" />
```

The removePersonalInformation element's val attribute has a value of true specifying that applications shall remove any personal information when saving this file. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the

Attributes	Description
	default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.69 revisionView (Visibility of Annotation Types)

This element specifies which forms of annotations shall be visible for a WordprocessingML document when it is displayed. This setting shall not affect whether annotations are added or persisted, it shall only affect the display of the annotations which exist in the document's contents (persisted or in memory).

If this element is omitted, then all forms of annotations shall be visible.

[*Example*: Consider the WordprocessingML below specifying that only formatting and ink annotations within a given WordprocessingML document shall be displayed when the document is opened:

```
<w:revisionView w:markup="false" w:comments="false" w:insDel="false" />
```

The revisionView element specifies that the visibility of the markup region, comments and content additions/deletions shall be suppressed by setting a value of `false`. Since the formatting and inkAnnotation attributes are omitted, they inherit the default of `true` and shall be displayed. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
comments (Display Comments)	Specifies if comments should be included when the contents of this document are displayed. If this attribute is omitted, then comments shall be displayed when annotations are visible based on application-level settings.

Attributes	Description
	[*Example*: Consider the WordprocessingML below specifying that comments shall be displayed: `<w:revisionView w:comments="true" />` The comments attribute has a value of `true`, specifying that comments shall be rendered when the document's annotations are displayed. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
formatting (Display Formatting Revisions)	Specifies if revisions to properties (i.e. formatting revisions) should be included when the contents of this document are displayed. If this attribute is omitted, then formatting revisions shall be displayed when annotations are visible based on application-level settings. [*Example*: Consider the WordprocessingML below specifying that formatting revisions shall be displayed: `<w:revisionView w:formatting="true" />` The formatting attribute has a value of `true`, specifying that formatting revisions shall be rendered when the document's annotations are displayed. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
inkAnnotations (Display Ink Annotations)	Specifies if ink annotations, specified in VML syntax (§6.1), should be included when the contents of this document are displayed. If this attribute is omitted, then ink annotations shall be displayed when annotations are visible based on application-level settings. [*Example*: Consider the WordprocessingML below specifying that ink annotations shall be displayed: `<w:revisionView w:inkAnnotations="true" />` The inkAnnotations attribute has a value of `true`, specifying that ink annotations shall be rendered when the document's annotations are displayed. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
insDel (Display Content Revisions)	Specifies if revisions to content (i.e. insertions, deletions, and moves) should be included when the contents of this document are displayed. If this attribute is omitted, then insertions, deletions, and moves shall be displayed when annotations are visible based on application-level settings.

Attributes	Description
	[*Example*: Consider the WordprocessingML below specifying that insertions, deletions, and moves shall be displayed: `<w:revisionView w:insDel="true" />` The insDel attribute has a value of `true`, specifying that insertions, deletions, and moves shall be rendered when the document's annotations are displayed. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
markup (Display Visual Indicator Of Markup Area)	Specifies if the application shall visually indicate any additional non-printing area used to display annotations when the annotations in this document are displayed. If this attribute is omitted, then any additional non-printing area shall be indicated when they are visible based on application-level settings. [*Example*: Consider the WordprocessingML below specifying that no visual indicator shall be displayed for non-printing regions holding annotations: `<w:revisionView w:markup="false" />` The markup attribute has a value of `false`, specifying that nothing shall be rendered indicating when a non-printing region is added when the document's annotations are displayed. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TrackChangesView">
   <attribute name="markup" type="ST_OnOff" use="optional"/>
   <attribute name="comments" type="ST_OnOff" use="optional"/>
   <attribute name="insDel" type="ST_OnOff" use="optional"/>
   <attribute name="formatting" type="ST_OnOff" use="optional"/>
   <attribute name="inkAnnotations" type="ST_OnOff" use="optional"/>
</complexType>
```

2.15.1.70 rsid (Single Session Revision Save ID)

This element specifies the revision save ID that was associated with a single editing session for a document. An editing session is a span of time that begins and ends with any event that produces an editable file, such as a save or an e-mail send, and contains no such event. When revision save IDs are added to a document, they shall follow these rules:

- Every editing session shall be assigned a revision save ID that is larger than all earlier ones in the same file

- Revision save IDs should be randomly generated based on the current time (to minimize the chance that two disparate editing sessions starting with the same immediate predecessor are assigned the same revision save ID)
- Changes to document content in an editing session shall be stamped with the current revision save ID using the appropriate rsid* attributes
- An identical rsid value between two documents with the same rsidRoot (§2.15.1.71) shall indicate the same editing sessions

[*Note*: A revision save ID should be treated as unique within the context of all documents with the same rsidRoot value. Although in practice it is possible for two independent sessions to result in the same value, this outcome is extremely rare as the values are based on the current time. However, the meaning of two revision save IDs is not defined for documents with a different rsidRoot. Applications may use this information as desired. *end note*]

[*Example*: Consider the following fragments from two WordprocessingML documents' document settings:

Document 1	Document 2
<w:rsids> <w:rsidRoot w:val="00464813"/> <w:rsid w:val="00455AAB" /> <w:rsid w:val="00464813" /> <w:rsid w:val="00996E03" /> </w:rsids>	<w:rsids> <w:rsidRoot w:val="00464813"/> <w:rsid w:val="00455AAB" /> <w:rsid w:val="00464813" /> <w:rsid w:val="00473403" /> <w:rsid w:val="0048414E" /> </w:rsids>

The rsid elements are identical for the first three editing sessions for both documents, indicating that these documents, although they are now separate, originated from the same document. The documents were then separated and the first was saved once afterwards; and the second, twice. *end example*]

Parent Elements
rsids (§2.15.1.72)

Attributes	Description
val (Long Hexadecimal Number Value)	Specifies a number value specified as a four digit hexadecimal number), whose contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following value for an attribute of type ST_LongHexNumber: 00BE2C6C. This value is valid, as it contains four hexadecimal digits, each an encoding of an octet of the actual decimal number value. It may therefore be interpreted as desired in the context of the parent XML element, *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LongHexNumber">
   <attribute name="val" type="ST_LongHexNumber" use="required"/>
</complexType>
```

2.15.1.71 rsidRoot (Original Document Revision Save ID)

This element specifies the revision save ID which was associated with the first editing session for this document. [*Note*: This information shall be identical between any number of copies of the same document, as they all originate from the same original editing session. Applications may use this information as desired. *end note*]

If this element is omitted, then the original document revision save ID is unknown.

[*Example*: Consider the following fragments from two WordprocessingML documents' document settings:

Document 1	Document 2
<w:rsids> <w:rsidRoot w:val="00464813"/> <w:rsid w:val="00455AAB" /> <w:rsid w:val="00464813" /> <w:rsid w:val="00996E03" /> </w:rsids>	<w:rsids> <w:rsidRoot w:val="00464813"/> <w:rsid w:val="00455AAB" /> <w:rsid w:val="00464813" /> <w:rsid w:val="00473403" /> <w:rsid w:val="0048414E" /> </w:rsids>

The rsidRoot element's val attribute has a value of `00464813` for both documents, indicating that these documents, although they are now separate, originated from the same document. This information may be used as desired. *end example*]

Parent Elements
rsids (§2.15.1.72)

Attributes	Description
val (Long Hexadecimal Number Value)	Specifies a number value specified as a four digit hexadecimal number), whose contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following value for an attribute of type ST_LongHexNumber: `00BE2C6C`. This value is valid, as it contains four hexadecimal digits, each an encoding of an octet of

Attributes	Description
	the actual decimal number value. It may therefore be interpreted as desired in the context of the parent XML element, *end example*] The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LongHexNumber">
   <attribute name="val" type="ST_LongHexNumber" use="required"/>
</complexType>
```

2.15.1.72 rsids (Listing of All Revision Save ID Values)

This element specifies the set of revision save ID values for the current document. *Revision save ID values* refer to four digit hexadecimal values which uniquely identify an editing session in the life of the current document. An *editing session* is the period of time between two subsequent save operations by an application.

[*Guidance*: The set of revision save IDs stored with a document only supplies information about the editing session in which document components were last saved, which may be used by applications in any manner desired. *end guidance*]

If this element is omitted, then no information is available about the set of revision save ID values for this document.

[*Example*: Consider a WordprocessingML document with the following information present in its document settings:

```
<w:rsids>
  <w:rsidRoot w:val="00464813" />
  <w:rsid w:val="00455AAB" />
  <w:rsid w:val="00464813" />
  <w:rsid w:val="00473403" />
</w:rsids>
```

The rsids element contains four child elements, specifying that the document was edited over four distinct editing sessions (i.e. it was saved three times). *end example*]

Parent Elements
settings (§2.15.1.78)

Child Elements	Subclause
rsid (Single Session Revision Save ID)	§2.15.1.70
rsidRoot (Original Document Revision Save ID)	§2.15.1.71

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DocRsids">
    <sequence>
        <element name="rsidRoot" type="CT_LongHexNumber" minOccurs="0" maxOccurs="1"/>
        <element name="rsid" type="CT_LongHexNumber" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
</complexType>
```

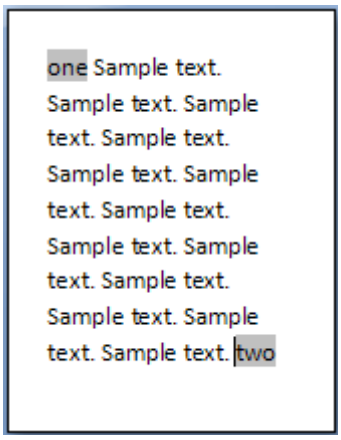
2.15.1.73 saveFormsData (Only Save Form Field Content)

This element specifies that saving the contents of this document shall only save the contents of WordprocessingML form fields defined using the `FORMTEXT`, `FORMCHECKBOX`, and `FORMDROPDOWN` field codes in a comma-delimited text format which does not conform to this Office Open XML Standard (i.e. it is a one-way export from a WordprocessingML document).

[*Rationale*: This setting is typically used to allow duplication of paper forms in electronic WordprocessingML document form, allowing the resulting content to be extracted as a comma-delimited text file. *end rationale*]

If this element is omitted, then the contents of the entire document (not just form fields) should be saved according to the definition of WordprocessingML in this Office Open XML Standard.

[*Example*: Consider a WordprocessingML document which has form fields in the top right and bottom left corners of the first page, as follows (with the text box form fields shaded in grey):



If the only content which shall be saved are the form fields' contents, this requirement is specified using the following WordprocessingML in the document settings:

```
<w:saveFormsData w:val="true" />
```

The saveFormsData element's val attribute as a value of `true`, specifying that only form field data shall be saved, resulting in output as follows in a text file:

```
one,two
```

end example]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.74 saveInvalidXml (Allow Saving Document As XML File When Custom XML Markup Is Invalid)

This element specifies that this document should be capable of being saved into a format consisting of a single XML file (not defined by this Office Open XML Standard) when its contents are invalid based on the custom XML markup contained in the document. This setting has no effect on documents that do not contain custom XML markup, or that do contain custom XML markup but do not have a schema attached. [*Guidance*: Because this setting specifies behavior when saving to an alternative file format not defined by this Office Open XML Standard, this behavior is optional. *end guidance*]

If this element is omitted, then applications should not allow this document to be saved into a single XML file when its contents are invalid based on the custom XML markup contained in the document. If the doNotValidateAgainstSchema element (§2.15.1.42) is set, then the XML is never "invalid" and this property is ignored.

[*Example*: Consider a WordprocessingML document which should be saved into a single XML file even when its custom XML content is marked invalid by applications which support this operation. This requirement is specified using the following WordprocessingML in the document settings:

```
<w:saveInvalidXml w:val="true" />
```

The saveInvalidXml element's val attribute has a value of `true` specifying that the content in this document can be saved regardless of its validation status. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.75 savePreviewPicture (Generate Thumbnail For Document On Save)

This element specifies if a document's Thumbnail part should be generated for the contents of the first page of this document when saved by application which support document thumbnail generation.

If this element is omitted, then applications may choose to save a thumbnail, however, that behavior is not required. If this element is specified, a thumbnail must be produced if that functionality is supported.

[*Example*: Consider a WordprocessingML document that specifies that a document thumbnail shall always be created when it is saved. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:savePreviewPicture w:val="true"/>
```

The savePreviewPicture element's val attribute has a value of `true` specifying that a document thumbnail should be generated each time this document is saved. *end example*]

Parent Elements

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.76 saveThroughXslt (Custom XSL Transform To Use When Saving As XML File)

This element specifies the location of a custom XSL transform which shall be used when this document is saved as a single XML file (in a format not defined by this Office Open XML Standard). [*Guidance*: Because this setting specifies behavior when saving to an alternative file format not defined by this Office Open XML Standard, this behavior is optional. *end guidance*]

If this element is omitted, then no custom XSL transform shall be used when saving this file as a single XML file. If the useXSLTWhenSaving element (§2.15.1.92) is omitted or set to `false`, then this transform shall not be applied when the document is saved as a single XML file.

[*Example*: Consider a XML document that shall have the XSL transform applied when the document is saved as a single XML file. This requirement would be specified using the following WordprocessingML in the document settings:

```
<w:useXSLTWhenSaving w:val="on"/>
<w:saveThroughXslt r:id="rId5" />
```

The useXSLTWhenSaving element's val is set to on indicating that applications shall apply the XSLT specified by the relationship targeted by the id attribute of the saveThroughXslt element, located at `rId5`, when saving as a single XML file*. end example]*

Parent Elements
settings (§2.15.1.78)

Attributes	Description
id (XSL Transformation Location) Namespace: .../officeDocument/2006/relationships	Specifies an explicit relationship to the location of the XSL Transformation which shall be applied. The relationship targeted by this element shall be of type http://schemas.openxmlformats.org/officeDocument/2006/relationships/transform , or this document shall be declared invalid. [*Example*: Consider a XML document that shall have the XSL transform located at `c:\Example Transform.xslt` applied when the document is saved as a single XML file. This requirement would be specified using the following WordprocessingML in the document settings: `<w:saveThroughXslt r:id="rId5" />` The saveThroughXslt element specifies that the relationship located at `rId5` shall be used when saving as a single XML file in this case, that relationship shall target `c:\Example Transform.xslt`. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
solutionID (Local Identifier for XSL Transform)	Specifies a string identifier which may be used to locate the XSL transform to be applied. The semantics of this attribute are not defined by this Office Open XML Standard - applications may use this information in any application-defined manner to resolve the location of the XSL transform to apply. If this attribute is omitted, then no local identifier is specified for the XSL transform. If both this and the xslt attributes are present, then this data shall be used first, and the latter shall only be used if this information cannot be used successfully. [*Example*: Consider a XML document that shall have the XSL transform identified by `mySolution` applied to when the document is saved as a single XML file. This requirement would be specified using the following WordprocessingML in the document settings: `<w:saveThroughXslt w:solutionID="mySolution" />` The solutionID attribute has a value of `mySolution` indicating that applications shall

Attributes	Description
	apply the XSLT identified by this value (if known) when saving as a single XML file. *end example]* The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SaveThroughXslt">
   <attribute ref="r:id" use="optional"/>
   <attribute name="solutionID" type="ST_String" use="optional"/>
</complexType>
```

2.15.1.77 saveXmlDataOnly (Only Save Custom XML Markup)

This element specifies that the contents of this document shall be saved as an XML file containing only the custom XML markup in this document in its regular form. The resulting document will not conform to this Office Open XML Standard (i.e. this is an export-only save option for a WordprocessingML document).

[*Rationale*: This setting is typically used to extract custom XML markup from a WordprocessingML document for further processing by XML-enabled applications. *end rationale*]

If this element is omitted, then the contents of the entire document (not just custom XML markup) should be saved according to the definition of WordprocessingML in this Office Open XML Standard.

[*Example*: Consider a WordprocessingML document which should be saved as an XML file containing only its custom XML markup. This requirement is specified using the following WordprocessingML fragment in the document settings:

```
<w:saveXmlDataOnly w:val=”true”/>
```

The saveXmlDataOnly element's val attribute has a value of `true` specifying that only custom XML shall be saved into a regular XML file when saving this document. For example, the document body formerly looked like this:

```
<w:body>
  <w:p>
    <w:customXml w:element="root" w:namespaceuri="urn:example">
      <w:r>
        <w:t>Hello world<w:t>
      </w:r>
    </w:customXml>
  </w:p>
</w:body>
```

The presence of this element specifies that the resulting document only contains the custom Xml markup, resulting in the following:

```
<ns0:root xmlns:ns0="urn:example">Hello world</ns0:root>
```

end example]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.78 settings (Document Settings)

This element specifies the settings that are applied to a WordprocessingML document. This element is the root element of the Document Settings part in a WordprocessingML document.

[*Example*: Consider the following WordprocessingML fragment for the settings part of a document:

```
<w:settings>
  <w:defaultTabStop w:val="720" />
  <w:characterSpacingControl w:val="dontCompress" />
</w:settings>
```

The settings element contains all of the settings for this document. In this case, the two settings applied are automatic tab stop increments of 0.5" using the defaultTabStop element, and no character level whitespace compression using the characterSpacingControl element. *end example*]

Parent Elements

Parent Elements
Root element of WordprocessingML Document Settings part

Child Elements	Subclause
activeWritingStyle (Grammar Checking Settings)	§2.15.1.1
alignBordersAndEdges (Align Paragraph and Table Borders with Page Border)	§2.15.1.2
alwaysMergeEmptyNamespace (Do Not Mark Custom XML Elements With No Namespace As Invalid)	§2.15.1.3
alwaysShowPlaceholderText (Use Custom XML Element Names as Default Placeholder Text)	§2.15.1.4
attachedSchema (Attached Custom XML Schema)	§2.15.1.5
attachedTemplate (Attached Document Template)	§2.15.1.6
autoFormatOverride (Allow Automatic Formatting to Override Formatting Protection Settings)	§2.15.1.9
autoHyphenation (Automatically Hyphenate Document Contents When Displayed)	§2.15.1.10
bookFoldPrinting (Book Fold Printing)	§2.15.1.11
bookFoldPrintingSheets (Number of Pages Per Booklet)	§2.15.1.12
bookFoldRevPrinting (Reverse Book Fold Printing)	§2.15.1.13
bordersDoNotSurroundFooter (Page Border Excludes Footer)	§2.15.1.14
bordersDoNotSurroundHeader (Page Border Excludes Header)	§2.15.1.15
captions (Caption Settings)	§2.15.1.17
characterSpacingControl (Character-Level Whitespace Compression)	§2.15.1.18
clickAndTypeStyle (Paragraph Style Applied to Automatically Generated Paragraphs)	§2.15.1.19
clrSchemeMapping (Theme Color Mappings)	§2.15.1.20
compat (Compatibility Settings)	§2.15.3.9
consecutiveHyphenLimit (Maximum Number of Consecutively Hyphenated Lines)	§2.15.1.21
decimalSymbol (Radix Point for Field Code Evaluation)	§2.15.1.22
defaultTableStyle (Default Table Style for Newly Inserted Tables)	§2.15.1.23
defaultTabStop (Distance Between Automatic Tab Stops)	§2.15.1.24
displayBackgroundShape (Display Background Objects When Displaying Document)	§2.15.1.25
displayHorizontalDrawingGridEvery (Distance between Horizontal Gridlines)	§2.15.1.26
displayVerticalDrawingGridEvery (Distance between Vertical Gridlines)	§2.15.1.27
documentProtection (Document Editing Restrictions)	§2.15.1.28
documentType (Document Classification)	§2.15.1.29
docVars (Document Variables)	§2.15.1.31
doNotAutoCompressPictures (Do Not Automatically Compress Images)	§2.15.1.32
doNotDemarcateInvalidXml (Do Not Show Visual Indicator For Invalid Custom XML Markup)	§2.15.1.33

Child Elements	Subclause
doNotDisplayPageBoundaries (Do Not Display Visual Boundary For Header/Footer or Between Pages)	§2.15.1.34
doNotEmbedSmartTags (Remove Smart Tags When Saving)	§2.15.1.35
doNotHyphenateCaps (Do Not Hyphenate Words in ALL CAPITAL LETTERS)	§2.15.1.36
doNotIncludeSubdocsInStats (Do Not Include Content in Text Boxes, Footnotes, and Endnotes in Document Statistics)	§2.15.1.37
doNotShadeFormData (Do Not Show Visual Indicator For Form Fields)	§2.15.1.38
doNotTrackFormatting (Do Not Track Formatting Revisions When Tracking Revisions)	§2.15.1.39
doNotTrackMoves (Do Not Use Move Syntax When Tracking Revisions)	§2.15.1.40
doNotUseMarginsForDrawingGridOrigin (Do Not Use Margins for Drawing Grid Origin)	§2.15.1.41
doNotValidateAgainstSchema (Do Not Validate Custom XML Markup Against Schemas)	§2.15.1.42
drawingGridHorizontalOrigin (Drawing Grid Horizontal Origin Point)	§2.15.1.43
drawingGridHorizontalSpacing (Drawing Grid Horizontal Grid Unit Size)	§2.15.1.44
drawingGridVerticalOrigin (Drawing Grid Vertical Origin Point)	§2.15.1.45
drawingGridVerticalSpacing (Drawing Grid Vertical Grid Unit Size)	§2.15.1.46
embedSystemFonts (Embed Common System Fonts)	§2.8.2.7
embedTrueTypeFonts (Embed TrueType Fonts)	§2.8.2.8
endnotePr (Document-Wide Endnote Properties)	§2.11.4
evenAndOddHeaders (Different Even/Odd Page Headers and Footers)	§2.10.1
footnotePr (Document-Wide Footnote Properties)	§2.11.11
forceUpgrade (Upgrade Document on Open)	§2.15.1.47
formsDesign (Structured Document Tag Placeholder Text Should be Resaved)	§2.15.1.48
gutterAtTop (Position Gutter At Top of Page)	§2.15.1.49
hdrShapeDefaults (Default Properties for VML Objects in Header and Footer)	§2.15.1.50
hideGrammaticalErrors (Do Not Display Visual Indication of Grammatical Errors)	§2.15.1.51
hideSpellingErrors (Do Not Display Visual Indication of Spelling Errors)	§2.15.1.52
hyphenationZone (Hyphenation Zone)	§2.15.1.53
ignoreMixedContent (Ignore Mixed Content When Validating Custom XML Markup)	§2.15.1.54
linkStyles (Automatically Update Styles From Document Template)	§2.15.1.55
listSeparator (List Separator for Field Code Evaluation)	§2.15.1.56
mailMerge (Mail Merge Settings)	§2.14.20
mathPr (Math Properties)	§7.1.2.62
mirrorMargins (Mirror Page Margins)	§2.15.1.57
noLineBreaksAfter (Custom Set of Characters Which Cannot End a Line)	§2.15.1.58
noLineBreaksBefore (Custom Set Of Characters Which Cannot Begin A Line)	§2.15.1.59

Child Elements	Subclause
noPunctuationKerning (Never Kern Punctuation Characters)	§2.15.1.60
printFormsData (Only Print Form Field Content)	§2.15.1.61
printFractionalCharacterWidth (Print Fractional Character Widths)	§2.15.1.62
printPostScriptOverText (Print PostScript Codes With Document Text)	§2.15.1.63
printTwoOnOne (Print Two Pages Per Sheet)	§2.15.1.64
proofState (Spelling and Grammatical Checking State)	§2.15.1.65
readModeInkLockDown (Freeze Document Layout)	§2.15.1.66
removeDateAndTime (Remove Date and Time from Annotations)	§2.15.1.67
removePersonalInformation (Remove Personal Information from Document Properties)	§2.15.1.68
revisionView (Visibility of Annotation Types)	§2.15.1.69
rsids (Listing of All Revision Save ID Values)	§2.15.1.72
saveFormsData (Only Save Form Field Content)	§2.15.1.73
saveInvalidXml (Allow Saving Document As XML File When Custom XML Markup Is Invalid)	§2.15.1.74
savePreviewPicture (Generate Thumbnail For Document On Save)	§2.15.1.75
saveSubsetFonts (Subset Fonts When Embedding)	§2.8.2.15
saveThroughXslt (Custom XSL Transform To Use When Saving As XML File)	§2.15.1.76
saveXmlDataOnly (Only Save Custom XML Markup)	§2.15.1.77
schemaLibrary (Embedded Custom XML Schema Supplementary Data)	§8.2.2
shapeDefaults (Default Properties for VML Objects in Main Document)	§2.15.1.79
showEnvelope (Show E-Mail Message Header)	§2.15.1.80
showXMLTags (Show Visual Indicators for Custom XML Markup Start/End Locations)	§2.15.1.81
smartTagType (Supplementary Smart Tag Information)	§2.15.1.82
strictFirstAndLastChars (Use Strict Kinsoku Rules for Japanese Text)	§2.15.1.83
styleLockQFSet (Prevent Replacement of Styles Part)	§2.15.1.84
styleLockTheme (Prevent Modification of Themes Part)	§2.15.1.85
stylePaneFormatFilter (Suggested Filtering for List of Document Styles)	§2.15.1.86
stylePaneSortMethod (Suggested Sorting for List of Document Styles)	§2.15.1.87
summaryLength (Percentage of Document to Use When Generating Summary)	§2.15.1.88
themeFontLang (Theme Font Languages)	§2.15.1.89
trackRevisions (Track Revisions to Document)	§2.15.1.90
uiCompat97To2003 (Disable Features Incompatible With Earlier Word Processing Formats)	§2.15.3.54
updateFields (Automatically Recalculate Fields on Open)	§2.15.1.91
useXSLTWhenSaving (Save Document as XML File through Custom XSL Transform)	§2.15.1.92
view (Document View Setting)	§2.15.1.93

Child Elements	Subclause
writeProtection (Write Protection)	§2.15.1.94
zoom (Magnification Setting)	§2.15.1.95

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Settings">
    <sequence>
        <element name="writeProtection" type="CT_WriteProtection" minOccurs="0"/>
        <element name="view" type="CT_View" minOccurs="0"/>
        <element name="zoom" type="CT_Zoom" minOccurs="0"/>
        <element name="removePersonalInformation" type="CT_OnOff" minOccurs="0"/>
        <element name="removeDateAndTime" type="CT_OnOff" minOccurs="0"/>
        <element name="doNotDisplayPageBoundaries" type="CT_OnOff" minOccurs="0"/>
        <element name="displayBackgroundShape" type="CT_OnOff" minOccurs="0"/>
        <element name="printPostScriptOverText" type="CT_OnOff" minOccurs="0"/>
        <element name="printFractionalCharacterWidth" type="CT_OnOff" minOccurs="0"/>
        <element name="printFormsData" type="CT_OnOff" minOccurs="0"/>
        <element name="embedTrueTypeFonts" type="CT_OnOff" minOccurs="0"/>
        <element name="embedSystemFonts" type="CT_OnOff" minOccurs="0"/>
        <element name="saveSubsetFonts" type="CT_OnOff" minOccurs="0"/>
        <element name="saveFormsData" type="CT_OnOff" minOccurs="0"/>
        <element name="mirrorMargins" type="CT_OnOff" minOccurs="0"/>
        <element name="alignBordersAndEdges" type="CT_OnOff" minOccurs="0"/>
        <element name="bordersDoNotSurroundHeader" type="CT_OnOff" minOccurs="0"/>
        <element name="bordersDoNotSurroundFooter" type="CT_OnOff" minOccurs="0"/>
        <element name="gutterAtTop" type="CT_OnOff" minOccurs="0"/>
        <element name="hideSpellingErrors" type="CT_OnOff" minOccurs="0"/>
        <element name="hideGrammaticalErrors" type="CT_OnOff" minOccurs="0"/>
        <element name="activeWritingStyle" type="CT_WritingStyle" minOccurs="0"
          maxOccurs="unbounded"/>
        <element name="proofState" type="CT_Proof" minOccurs="0"/>
        <element name="formsDesign" type="CT_OnOff" minOccurs="0"/>
        <element name="attachedTemplate" type="CT_Rel" minOccurs="0"/>
        <element name="linkStyles" type="CT_OnOff" minOccurs="0"/>
        <element name="stylePaneFormatFilter" type="CT_ShortHexNumber" minOccurs="0"/>
        <element name="stylePaneSortMethod" type="CT_ShortHexNumber" minOccurs="0"/>
        <element name="documentType" type="CT_DocType" minOccurs="0"/>
        <element name="mailMerge" type="CT_MailMerge" minOccurs="0"/>
        <element name="revisionView" type="CT_TrackChangesView" minOccurs="0"/>
        <element name="trackRevisions" type="CT_OnOff" minOccurs="0"/>
        <element name="doNotTrackMoves" type="CT_OnOff" minOccurs="0"/>
        <element name="doNotTrackFormatting" type="CT_OnOff" minOccurs="0"/>
        <element name="documentProtection" type="CT_DocProtect" minOccurs="0"/>
        <element name="autoFormatOverride" type="CT_OnOff" minOccurs="0"/>
        <element name="styleLockTheme" type="CT_OnOff" minOccurs="0"/>
        <element name="styleLockQFSet" type="CT_OnOff" minOccurs="0"/>
        <element name="defaultTabStop" type="CT_TwipsMeasure" minOccurs="0"/>
        <element name="autoHyphenation" type="CT_OnOff" minOccurs="0"/>
        <element name="consecutiveHyphenLimit" type="CT_DecimalNumber" minOccurs="0"/>
        <element name="hyphenationZone" type="CT_TwipsMeasure" minOccurs="0"/>
        <element name="doNotHyphenateCaps" type="CT_OnOff" minOccurs="0"/>
        <element name="showEnvelope" type="CT_OnOff" minOccurs="0"/>
        <element name="summaryLength" type="CT_DecimalNumber" minOccurs="0"/>
        <element name="clickAndTypeStyle" type="CT_String" minOccurs="0"/>
        <element name="defaultTableStyle" type="CT_String" minOccurs="0"/>
        <element name="evenAndOddHeaders" type="CT_OnOff" minOccurs="0"/>
        <element name="bookFoldRevPrinting" type="CT_OnOff" minOccurs="0"/>
```

```
        <element name="bookFoldPrinting" type="CT_OnOff" minOccurs="0"/>
        <element name="bookFoldPrintingSheets" type="CT_DecimalNumber" minOccurs="0"/>
        <element name="drawingGridHorizontalSpacing" type="CT_TwipsMeasure" minOccurs="0"/>
        <element name="drawingGridVerticalSpacing" type="CT_TwipsMeasure" minOccurs="0"/>
        <element name="displayHorizontalDrawingGridEvery" type="CT_DecimalNumber" minOccurs="0"/>
        <element name="displayVerticalDrawingGridEvery" type="CT_DecimalNumber" minOccurs="0"/>
        <element name="doNotUseMarginsForDrawingGridOrigin" type="CT_OnOff" minOccurs="0"/>
        <element name="drawingGridHorizontalOrigin" type="CT_TwipsMeasure" minOccurs="0"/>
        <element name="drawingGridVerticalOrigin" type="CT_TwipsMeasure" minOccurs="0"/>
        <element name="doNotShadeFormData" type="CT_OnOff" minOccurs="0"/>
        <element name="noPunctuationKerning" type="CT_OnOff" minOccurs="0"/>
        <element name="characterSpacingControl" type="CT_CharacterSpacing" minOccurs="0"/>
        <element name="printTwoOnOne" type="CT_OnOff" minOccurs="0"/>
        <element name="strictFirstAndLastChars" type="CT_OnOff" minOccurs="0"/>
        <element name="noLineBreaksAfter" type="CT_Kinsoku" minOccurs="0"/>
        <element name="noLineBreaksBefore" type="CT_Kinsoku" minOccurs="0"/>
        <element name="savePreviewPicture" type="CT_OnOff" minOccurs="0"/>
        <element name="doNotValidateAgainstSchema" type="CT_OnOff" minOccurs="0"/>
        <element name="saveInvalidXml" type="CT_OnOff" minOccurs="0"/>
        <element name="ignoreMixedContent" type="CT_OnOff" minOccurs="0"/>
        <element name="alwaysShowPlaceholderText" type="CT_OnOff" minOccurs="0"/>
        <element name="doNotDemarcateInvalidXml" type="CT_OnOff" minOccurs="0"/>
        <element name="saveXmlDataOnly" type="CT_OnOff" minOccurs="0"/>
        <element name="useXSLTWhenSaving" type="CT_OnOff" minOccurs="0"/>
        <element name="saveThroughXslt" type="CT_SaveThroughXslt" minOccurs="0"/>
        <element name="showXMLTags" type="CT_OnOff" minOccurs="0"/>
        <element name="alwaysMergeEmptyNamespace" type="CT_OnOff" minOccurs="0"/>
        <element name="updateFields" type="CT_OnOff" minOccurs="0"/>
        <element name="hdrShapeDefaults" type="CT_ShapeDefaults" minOccurs="0"/>
        <element name="footnotePr" type="CT_FtnDocProps" minOccurs="0"/>
        <element name="endnotePr" type="CT_EdnDocProps" minOccurs="0"/>
        <element name="compat" type="CT_Compat" minOccurs="0"/>
        <element name="docVars" type="CT_DocVars" minOccurs="0"/>
        <element name="rsids" type="CT_DocRsids" minOccurs="0"/>
        <element ref="m:mathPr" minOccurs="0" maxOccurs="1"/>
        <element name="uiCompat97To2003" type="CT_OnOff" minOccurs="0"/>
        <element name="attachedSchema" type="CT_String" minOccurs="0" maxOccurs="unbounded"/>
        <element name="themeFontLang" type="CT_Language" minOccurs="0" maxOccurs="1"/>
        <element name="clrSchemeMapping" type="CT_ColorSchemeMapping" minOccurs="0"/>
        <element name="doNotIncludeSubdocsInStats" type="CT_OnOff" minOccurs="0"/>
        <element name="doNotAutoCompressPictures" type="CT_OnOff" minOccurs="0"/>
        <element name="forceUpgrade" type="CT_Empty" minOccurs="0" maxOccurs="1"/>
        <element name="captions" type="CT_Captions" minOccurs="0" maxOccurs="1"/>
        <element name="readModeInkLockDown" type="CT_ReadingModeInkLockDown" minOccurs="0"/>
        <element name="smartTagType" type="CT_SmartTagType" minOccurs="0" maxOccurs="unbounded"/>
        <element ref="sl:schemaLibrary" minOccurs="0" maxOccurs="1"/>
        <element name="shapeDefaults" type="CT_ShapeDefaults" minOccurs="0"/>
        <element name="doNotEmbedSmartTags" type="CT_OnOff" minOccurs="0"/>
        <element name="decimalSymbol" type="CT_String" minOccurs="0" maxOccurs="1"/>
        <element name="listSeparator" type="CT_String" minOccurs="0" maxOccurs="1"/>
    </sequence>
</complexType>
```

2.15.1.79 shapeDefaults (Default Properties for VML Objects in Main Document)

This element specifies the default parameters for object using the VML syntax (§6.1) inserted in the body (the main document story, comments, footnotes, and endnotes) of the WordprocessingML document. The definition and semantics of these parameters is described in the VML - Office Drawing subclause (§6.2) of this Office Open XML Standard.

If this element is omitted, then no default properties are applied to VML objects in the body of this document.

[*Example*: Consider a WordprocessingML document whose document settings contain the following markup:

```
<w:shapeDefaults>
  <o:shapedefaults v:ext="edit" spidmax="1026" />
  <o:shapelayout v:ext="edit">
    <o:idmap v:ext="edit" data="1" />
  </o:shapelayout>
</w:shapeDefaults>
```

The shapeDefaults element specifies a set of shape defaults which shall be applied to the set of all shapes present in the body document. *end example*]

Parent Elements
settings (§2.15.1.78)

Child Elements	Subclause
Any element from the urn:schemas-microsoft-com:office:office namespace	§6.2

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShapeDefaults">
   <choice maxOccurs="unbounded">
      <any processContents="lax" namespace="urn:schemas-microsoft-com:office:office" minOccurs="0"
        maxOccurs="unbounded"/>
   </choice>
</complexType>
```

2.15.1.80 showEnvelope (Show E-Mail Message Header)

This element specifies that an e-mail message header shall be displayed when this document is opened, if an e-mail header is supported by the application opening the file.

If this element is omitted, then applications shall not display the e-mail message header automatically when this file is opened, even if one is available in the application opening the file.

[*Example*: Consider a WordprocessingML document which should show an e-mail message header when opened. This requirement is specified using the following WordprocessingML in the document settings:

```
<w:showEnvelope w:val="true" />
```

The showEnvelope element's val attribute has a value of `true` specifying that an e-mail message header shall be displayed when the document is viewed, whenever such functionality is available. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.81 showXMLTags (Show Visual Indicators for Custom XML Markup Start/End Locations)

This element specifies that some visual indicator shall be provided for the start and end locations of custom XML markup present in this document, if any.

If this element is omitted, then applications should not provide any visual indicator of the locations of custom XML markup start/end tags.

[*Example*: Consider a WordprocessingML document which should show a visual indicator to the location of custom XML markup elements. This requirement is specified using the following WordprocessingML in the document settings:

```
<w:showXMLTags w:val="true" />
```

The showXMLTags element's val attribute has a value of `true` specifying that custom XML markup should have a visual indicator in the document when displayed. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.82 smartTagType (Supplementary Smart Tag Information)

This element specifies optional supplementary information about one or more smart tags (§2.5.1.9) used in the current WordprocessingML document. This supplementary data is linked to the smart tag to which it applies via its name and namespaceuri attributes.

[*Example*: Consider a smart tag which has supplementary information defined as using the following WordprocessingML:

```
<w:smartTagType w:name="companyName" w:namespaceuri="urn:smartTagExample"
w:url="http://www.contoso.com/smartTag" >
```

The name and namespaceuri attributes specify that the smart tag to which this data shall be `companyName` in the `urn:smartTagExample` namespace. The supplementary data is an associated URL of `http://www.contoso.com/smartTag`. *end example*]

Parent Elements

Parent Elements
settings (§2.15.1.78)

Attributes	Description
name (Smart Tag Name)	Specifies the name of the smart tag within the document for which supplementary data is provided. [*Example*: Consider a smart tag which has a name of `companyName`. This name would be referenced using the following WordprocessingML: `<w:smartTagType w:name="companyName" … >` The name attribute specifies that the name for this smart tag shall be `companyName`. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).
namespaceuri (Smart Tag Namespace)	Specifies the namespace URI of the smart tag for which supplementary data is provided. If this attribute is omitted, the URI shall be assumed to be null (no associated URI). [*Example*: Consider a smart tag which shall have a namespace URI of `urn:smartTagExample`. This namespace would be referenced using the following WordprocessingML: `<w:smartTagType w:namespaceuri="urn:smartTagExample" />` The namespaceuri attribute specifies that the namespace for the smart tag to which this data applies shall be `urn:smartTagExample`. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).
url (Smart Tag Supplementary URL)	Specifies a URL provided for a particular smart tag type in this document. [*Note*: This URL is typically used to provide access to a URL for additional updates to this smart tag type as requested by the smart tag provider. *end note*] If this attribute is omitted, then no supplementary URL is provided for this type. [*Example*: Consider a smart tag which shall have a supplementary URL of `http://www.contoso.com/smartTag`. This URL would be specified using the following WordprocessingML: `<w:smartTagType … w:url="http://www.contoso.com/smartTag" />` The url attribute specifies that the supplementary data for the smart tag to which this data applies shall be `http://www.contoso.com/smartTag`. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SmartTagType">
   <attribute name="namespaceuri" type="ST_String"/>
   <attribute name="name" type="ST_String"/>
   <attribute name="url" type="ST_String"/>
</complexType>
```

2.15.1.83 strictFirstAndLastChars (Use Strict Kinsoku Rules for Japanese Text)

This element specifies that the strict set of Kinsoku rules shall be applied to Japanese text in this document when the kinsoku element (§2.3.1.16) is applied to that text. The resulting line breaking rules are provided on the kinsoku element.

If this element is omitted, then standard rules shall apply to Japanese text when the kinsoku element is applied to that text.

[*Example*: Consider a WordprocessingML document that specifies that strict Kinsoku rules shall be applied to Japanese text. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:strictFirstAndLastChars w:val="true"/>
```

The strictFirstAndLastChars element's val attribute has a value of `true` specifying that a document shall apply the strict set of invalid characters for the start and end of a line. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.84 styleLockQFSet (Prevent Replacement of Styles Part)

This element specifies whether applications shall prevent the replacement of the complete set of styles stored in the Styles part when editing this document. This setting should not preclude the editing or removal of individual styles, instead, it should only prevent the removal and replacement of the entire styles part in a single operation (either through a user interface or a programmatic operation).

If this element is omitted, then applications may allow the replacement of the entire styles part in this document.

[*Example*: Consider a WordprocessingML document that specifies that applications shall prevent the replacement of the entire styles part. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:styleLockQFSet w:val="true"/>
```

The styleLockQFSet element's val attribute has a value of `true` specifying that individual style changes should be allowed, but the styles data shall not be replaced as a whole via a single operation. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.85 styleLockTheme (Prevent Modification of Themes Part)

This element specifies whether applications shall prevent the modification of the document's theme information stored in the Theme part when editing this document. This setting should not preclude the use of the theme information, instead, it should only prevent the modification of the theme part in a single operation (either through a user interface or a programmatic operation).

If this element is omitted, then applications may allow the replacement or modification of the theme part in this document.

[*Example*: Consider a WordprocessingML document that specifies that applications shall prevent the modification of the theme part. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:styleLockTheme w:val="true"/>
```

The styleLockTheme element's val attribute has a value of `true` specifying that theme data shall not be modified when modifying the contents of this document. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.86 stylePaneFormatFilter (Suggested Filtering for List of Document Styles)

This element specifies a set of suggested filters which should be applied to the list of document styles in this application if the styles are displayed in a user interface.

The val attribute of this element contains a bitmask of the following filtering options:

Value	Description
0x0001	Specifies that all styles present in the styles part should be displayed in the list of document styles.
0x0002	Specifies that only styles with the customStyle attribute should be displayed in the list of document styles.
0x0004	Specifies that all latent styles should be displayed in the list of document styles.
0x0008	Specifies that only styles used in the document should be displayed in the list of document styles.
0x0010	Undefined. Shall not be used.
0x0020	Specifies that heading styles (styles with a styleId of Heading1 to Heading9) should be displayed in the list of document styles when the previous style is used in the document and/or is present in the styles part.
0x0040	Specifies that numbering styles should be displayed in the list of document styles.
0x0080	Specifies that table styles should be displayed in the list of document styles.
0x0100	Specifies that all unique forms of run-level direct formatting should be displayed in the list of document styles as though they were each a unique style.
0x0200	Specifies that all unique forms of paragraph-level direct formatting should be displayed in the list of document styles as though they were each a unique style.
0x0400	Specifies that all unique forms of direct formatting of numbering data should be displayed in the list of document styles as though they were each a unique style.
0x0800	Specifies that all unique forms of direct formatting of tables should be displayed in the list of document styles as though they were each a unique style.
0x1000	Specifies that a style should be present which removes all formatting and styles from text.
0x2000	Specifies that heading styles with a styleId of Heading1 to Heading3 should always be displayed in the list of document styles.
0x4000	Specifies that styles should only be shown the semiHidden element (§2.7.3.16) is false

Value	Description
	and the hidden element (§2.7.3.4) is `false`.
`0x8000`	Specifies that primary names for styles should not be shown if an alternate name using the name element (§2.7.3.9) exists.
Any other value	Undefined. Shall not be used.

If this element is omitted, then all settings defined by this element are turned off.

[*Example*: Consider a document with the following value in its document settings:

```
<w:stylePaneFormatFilter w:val="2002" />
```

The stylePaneFormatFilter element's settings specify two suggested filter options for the list of document styles:

- Only custom styles should be shown (`0002`)
- Heading styles with a styleId of `Heading1` to `Heading3` should always be displayed in the list (`2000`)

end example]

Parent Elements
settings (§2.15.1.78)

Attributes	Description

val (Two Digit Hexadecimal Value)	Specifies a value specified as a two digit hexadecimal number), whose contents are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:tblPr>`
	`  <w:tblLook w:val="0010" />`
	`</w:tblPr>`
	
	The value of `0010` is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_ShortHexNumber simple type (§2.18.86).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShortHexNumber">
   <attribute name="val" type="ST_ShortHexNumber" use="required"/>
</complexType>
```

2.15.1.87 stylePaneSortMethod (Suggested Sorting for List of Document Styles)

This element specifies a suggested sorting which should be applied to the list of document styles in this application if the styles are displayed in a user interface.

The val attribute of this element specifies one of the following sorting options:

Value	Description
0x0000	Specifies that styles which are visible should be sorted by their names.
0x0001	Specifies that styles which are visible should be sorted by their UI priority using the uiPriority element (§2.7.3.19).
0x0002	Specifies that styles which are visible should be sorted by the default sorting of the host application.
0x0003	Specifies that styles which are visible should be sorted by the font which they apply.
0x0004	Specifies that styles which are visible should be sorted by the style on which they are based using the basedOn element (§2.7.3.3).
0x0005	Specifies that styles which are visible should be sorted by their style types (i.e. character, linked, paragraph).
Any other value	Undefined. Shall not be used.

If this element is omitted, then styles which are visible should be sorted by the default sorting of the host application.

[*Example*: Consider a document with the following value in its document settings:

```
<w:stylePaneSortMethod w:val="0005" />
```

The stylePaneFormatFilter element's val attribute specifies that styles which are visible should be sorted by their style types (i.e. character, linked, paragraph) via a value of 0005. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description

val (Two Digit Hexadecimal Value)	Specifies a value specified as a two digit hexadecimal number), whose contents are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:tblPr>`
	`  <w:tblLook w:val="0010" />`

Attributes	Description
	</w:tblPr> The value of 0010 is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_ShortHexNumber simple type (§2.18.86).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShortHexNumber">
   <attribute name="val" type="ST_ShortHexNumber" use="required"/>
</complexType>
```

2.15.1.88 summaryLength (Percentage of Document to Use When Generating Summary)

This element specifies the size for automatic document summaries performed on the content of a WordprocessingML document. An *automatic document summary* is a subset of text contained in a document deemed by the hosting application to summarize the content of the WordprocessingML document. The val attribute of this element specifies the size of an automatic document summary to be performed on a given WordprocessingML document as a percentage of the total size of the given WordprocessingML document. Performing an automatic document summary is a runtime operation outside the scope of this Office Open XML Standard.

If this element is omitted, then applications may summarize this document to any desired size.

[*Example*: Consider a WordprocessingML document whose automatic document summary shall be ten percent of the size of the given WordprocessingML document. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:summaryLength w:val="10" />
```

The summaryLength element's val attribute is equal to 10 specifying that any automatic document summary shall be ten percent of the size of the document. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber:

Attributes	Description
	<w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.15.1.89 themeFontLang (Theme Font Languages)

This element specifies the language which shall be used to determine the appropriate theme fonts in the document's Theme part which map to the major/minor theme fonts.

These mappings are performed as follows:

- For `majorAscii`/`majorHAnsi`, locate the font element (§5.1.4.1.16) in the majorFont element (§5.1.4.1.24) in the theme part for the language specified by the val attribute
- For `majorBidi`, locate the font element in the majorFont element in the theme part for the language specified by the bidi attribute
- For `majorEastAsia`, locate the font element in the majorFont element in the theme part for the language specified by the eastAsia attribute
- For `minorAscii`/`minorHAnsi`, locate the font element in the minorFont element (§5.1.4.1.25) in the theme part for the language specified by the val attribute
- For `minorBidi`, locate the font element in the minorFont element in the theme part for the language specified by the bidi attribute
- For `minorEastAsia`, locate the font element in the minorFont element in the theme part for the language specified by the eastAsia attribute

If this element is omitted, then the default fonts for each region as specified by the latin, ea, and cs elements (§5.1.5.3.7; §5.1.5.3.3; §5.1.5.3.1) should be used.

[*Example*: Consider a document with the following WordprocessingML in its document settings:

```
<w:themeFontLang w:val="ja-JP" />
```

The themeFontLang element's val attribute has a value of `ja-JP`, specifying that the theme fonts used for Latin text shall be the theme fonts for Japanese. If the following content was present in the theme part:

```
…
<a:majorFont>
  …
  <a:font script="Jpan" typeface="MS Mincho"/>
  …
</a:majorFont>
…
```

Then this setting would specify that uses of the `majorAscii` and `majorHAnsi` theme font enumerations shall be mapped to the `MS Mincho` font. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description

bidi (Complex Script Language)	Specifies the language which shall be used when processing the contents of this run which use complex script characters, as determined by the Unicode character values of the run content.
	
	If this attribute is omitted, then the languages for the contents of this run using complex script characters shall be automatically determined based on their contents using any appropriate method.
	
	[*Example*: Consider a run which contains complex script characters in its contents. If those contents should be interpreted as Hebrew, that requirement would be specified as follows in the resulting WordprocessingML:
	
	`<w:r>`
	`  <w:rPr>`
	`  <w:lang w:bidi="he-IL" />`
	`  </w:rPr>`
	`</w:r>`
	
	The resulting run specifies that any complex script contents shall be spell and grammar checked using a Hebrew dictionary and grammar engine, if one is available. *end example*]
	
	The possible values for this attribute are defined by the ST_Lang simple type (§2.18.51).

eastAsia (East Asian Language)	Specifies the language which shall be used when processing the contents of this run which use East Asian characters, as determined by the Unicode character values of the run content. If this attribute is omitted, then the languages for the contents of this run using East Asian characters shall be automatically determined based on their contents using any appropriate method.

Attributes	Description

	[*Example*: Consider a run which contains East Asian characters in its contents. If those contents should be interpreted as Korean, that requirement would be specified as follows in the resulting WordprocessingML:
	
	`<w:r>`
	`  <w:rPr>`
	`  <w:lang w:bidi="ko-KR" />`
	`  </w:rPr>`
	`</w:r>`
	
	The resulting run specifies that any complex script contents shall be spell and grammar checked using a Korean dictionary and grammar engine, if one is available. *end example*]
	
	The possible values for this attribute are defined by the ST_Lang simple type (§2.18.51).

val (Latin Language)	Specifies the language which shall be used to check spelling and grammar (if requested) when processing the contents of this run which use Latin characters, as determined by the Unicode character values of the run content.
	
	If this attribute is omitted, then the languages for the contents of this run using Latin characters shall be automatically determined based on their contents using any appropriate method.
	
	[*Example*: Consider a run which contains Latin characters in its contents. If those contents should be interpreted as English (Canada), that requirement would be specified as follows in the resulting WordprocessingML:
	
	`<w:r>`
	`  <w:rPr>`
	`  <w:lang w:bidi="en-CA" />`
	`  </w:rPr>`
	`</w:r>`
	
	The resulting run specifies that any complex script contents shall be spell and grammar checked using a English (Canada) dictionary and grammar engine, if one is available. *end example*]
	
	The possible values for this attribute are defined by the ST_Lang simple type (§2.18.51).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Language">
   <attribute name="val" type="ST_Lang" use="optional"/>
   <attribute name="eastAsia" type="ST_Lang" use="optional"/>
   <attribute name="bidi" type="ST_Lang" use="optional"/>
</complexType>
```

2.15.1.90 trackRevisions (Track Revisions to Document)

This element specifies that applications shall track revisions made to the WordprocessingML document. *Revisions* are changes to a WordprocessingML document which are recorded such that they can be viewed independently, accepted or removed, and reverted if needed. When revisions are tracked, the resulting WordprocessingML markup in the Revisions subclause of this document describes the necessary syntax.

If this element is omitted, then revisions shall not be generated by changes to the contents of this document.

[*Example*: Consider a WordprocessingML document containing the text run `Example` that shall not have revisions tracked. Example WordprocessingML from Document 1 is given below:

```
<w:document>
  <w:body>
    <w:p>
      <w:r>
        <w:t>Example</w:t>
      </w:r>
    </w:p>
  </w:body>
</w:document>
```

And the corresponding document settings:

```
<w:settings>
  <w:trackRevisions w:val="false"/>
  …
</w:settings>
```

If the word `text` was added to the end of this document and bolded without revisions tracked, the resulting WordprocessingML would be output as follows:

```
<w:document>
  <w:body>
    <w:p>
      <w:r>
        <w:t>Example</w:t>
      </w:r>
      <w:r>
        <w:rPr>
          <w:b/>
        </w:rPr>
        <w:t>text</w:t>
      </w:r>
    </w:p>
  </w:body>
</w:document>
```

And the corresponding document settings:

```
<w:settings>
  <w:trackRevisions w:val="false"/>
  …
</w:settings>
```

Finally, assume the same insertion and formatting took place when the trackRevisions element's val attribute was set to on, the resulting WordprocessingML would be output as follows:

```
<w:document>
  <w:body>
    <w:p>
      <w:r>
        <w:t>Example</w:t>
      </w:r>
      <w:ins … >
        <w:r>
          <w:rPr>
            <w:b/>
            <w:rPrChange … >
              <w:rPr/>
            <w:rPrChange>
          </w:rPr>
          <w:t>text</w:t>
        </w:r>
      </w:ins>
    </w:p>
  </w:body>
</w:document>
```

And the corresponding document settings:

```
<w:settings>
  <w:trackRevisions w:val="true"/>
  …
</w:settings>
```

The trackRevisions element's val attribute was set to `true`, therefore the changes to the content of the document were inserted using the appropriate annotation elements in the document's WordprocessingML. Specifically, inserting the text `Text` to the right of the existing text was tracked as a revision with the ins element. In addition, applying bold formatting to the text was tracked as a revision with the rPrChange element. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted.

Attributes	Description
	A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.91 updateFields (Automatically Recalculate Fields on Open)

This element specifies whether the fields contained in this document should automatically have their field result recalculated from the field codes when this document is opened by an application which supports field calculations. [*Note*: Some fields are always recalculated (e.g. the page numbering), therefore this element only affects fields which are typically not automatically recalculated on opening the document. Also note that this setting shall not supersede any document protection (§2.15.1.28) or write protection (§2.15.1.94) settings. *end note*]

If this element is omitted, then fields should not automatically be recalculated on opening this document.

[*Example*: Consider a WordprocessingML document that specifies that applications should attempt to automatically recalculate fields from their field codes upon opening this document. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:updateFields w:val="true"/>
```

The updateFields element's val attribute has a value of `true` specifying that all fields should automatically be recalculated when opening this document. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but

Attributes	Description
	this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.92 useXSLTWhenSaving (Save Document as XML File through Custom XSL Transform)

This element specifies that this document should be saved through the custom XSLT transform defined by the saveThroughXslt element (§2.15.1.76) in this document when it is saved as a single XML file (not defined by this Office Open XML Standard). [*Guidance*: Because this setting specifies behavior when saving to an alternative file format not defined by this Office Open XML Standard, this behavior is optional. *end guidance*]

If the saveXmlDataOnly element (§2.15.1.77) is specified, then the single XML file to be transformed is the custom XML markup of the document, otherwise, it a format outside the scope of this Office Open XML Standard. If the XSL transform specified by the saveThroughXslt element is not present, then this setting should be ignored.

If this element is omitted, then this document should not be saved through a custom XSL transform when it is saved as a single XML file.

[*Example*: Consider a WordprocessingML document which should be saved through a custom XSL transform when it is saved as a single XML file. This requirement is specified using the following WordprocessingML in the document settings:

```
<w:useXSLTWhenSaving w:val="true" />
```

The useXSLTWhenSaving element's val attribute has a value of `true` specifying that the content in this document should be saved as a single XML file through the custom XSLT specified by the saveThroughXslt element. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.1.93 view (Document View Setting)

This element specifies the manner in which the contents of this document should be displayed when opened by an application.

If this element is omitted, then an application may view the document in any desired default state.

[*Example*: Consider a WordprocessingML document that shall be displayed on the screen in the same form as it will be printed. This requirement would be specified using the following WordprocessingML in the document settings:

```
<w:view w:val="print" />
```

The view element's val attribute is equal to print specifying that the given WordprocessingML document shall be rendered as it will be printed. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (Document View Setting Value)	Specifies the view which shall be used to render the contents of a WordprocessingML document.

Attributes	Description
	Applications may omit support for one or more of the views defined by the ST_View simple type (referenced below). If a WordprocessingML document containing an unsupported view is loaded by an application, it shall fall back to its default view (equivalent to use of the enumeration value none). [*Example*: Consider a WordprocessingML document that shall be rendered in a view meant to mimic how the document would look in a web browser (i.e. without a fixed page width). This requirement would be specified using the following WordprocessingML in the document settings: `<w:view w:val="web" />` The val attribute is equal to web specifying that the given WordprocessingML document shall be rendered in a view mimicking web page display. *end example*] The possible values for this attribute are defined by the ST_View simple type (§2.18.112).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_View">
   <attribute name="val" type="ST_View" use="required"/>
</complexType>
```

2.15.1.94 writeProtection (Write Protection)

This element specifies the write protection settings which have been applied to a WordprocessingML document. *Write protection* refers to a mode in which the document's contents cannot be edited, and the document cannot be resaved using the same file name. This setting is independent of the documentProtection (§2.15.1.28) element, but like document protection, this setting is not intended as a security feature and may be ignored.

When present, the write protection shall result in one of two write protection behaviors:

- If the password attribute is present, or both attributes are omitted, then the application shall prompt for a password to exit write protection. If the supplied password does not match the hash value in this attribute, then write protection shall be enabled.
- If only the recommended attribute is present, the application should provide user interface recommending that the user open this document in write protected state. If the user chooses to do so, the document shall be write protected, otherwise, it shall be opened fully editable.

If this element is omitted, then no write protection shall be applied to the current document.

[*Example*: Consider a WordprocessingML document that can be opened but only in a write protected state unless a password isdedivorp , in which case the file would be opened in an editable state. This requirement would be specified using the following WordprocessingML in the document settings:

```
<w:writeProtection w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" />
```

The writeProtection element is present which specifies that write protection shall be turned on for this document. Since the password ot lauqe si etubirtta 9oN7nWkCAyEZib1RomSJTjmPpCY= the given WordprocessingML document can only be opened in a write protected state unless a password which matches the hash value 9oN7nWkCAyEZib1RomSJTjmPpCY=is provided; in which case the file would be opened in an editable state. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description

algIdExt (Cryptographic Algorithm Extensibility)	Specifies that a cryptographic algorithm which was not defined by this Office Open XML Standard has been used to generate the hash value stored with this document.
	
	This value, when present, shall be interpreted based on the value of the algIdExtSource attribute in order to determine the algorithm used, which shall be application-defined. [*Rationale*: This extensibility affords the fact that with exponentially increasing computing power, documents created in the future will likely need to utilize as yet undefined hashing algorithms in order to remain secure. *end rationale*]
	
	If this value is present, the cryptAlgorithmClass, cryptAlgorithmType, and cryptAlgorithmSid attribute values shall be ignored in favor of the algorithm defined by this attribute.
	
	[*Example*: Consider a WordprocessingML document with the following information stored in one of its protection elements:
	
	`<w:… w:algIdExt="0000000A"`
	`w:algIdExtSource="futureCryptography"`
	`w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" />`
	
	The algIdExt attribute value of 0000000A specifies that the algorithm with hex code A shall be used as defined by the futureCryptography application. *end example*]
	
	The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).

algIdExtSource (Algorithm Extensibility Source)	Specifies the application which defined the algorithm value specified by the algIdExt attribute.
	
	[*Example*: Consider a WordprocessingML document with the following information stored in one of its protection elements:
	
	`<w:… w:algIdExt="0000000A"`
	`w:algIdExtSource="futureCryptography"`
	`w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" />`

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>The algIdExtSource attribute value of futureCryptography specifies that the algorithm used here was published by the futureCryptography application. end example]

The possible values for this attribute are defined by the ST_String simple type (§2.18.89).</td></tr>
<tr><td>cryptAlgorithmClass (Cryptographic Algorithm Class)</td><td>Specifies the class of cryptographic algorithm used by this protection. [Note: The initial version of this Office Open XML Standard only supports a single version - hash - but future versions may expand this as necessary. end note]

[Example: Consider a WordprocessingML document with the following information stored in one of its protection elements:

<pre><w:… w:cryptAlgorithmClass="hash"
  w:cryptAlgorithmType="typeAny"
  w:cryptAlgorithmSid="1"
  w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" /></pre>
The cryptAlgorithmClass attribute value of hash specifies that the algorithm used for the password is a hashing algorithm. end example]

The possible values for this attribute are defined by the ST_AlgClass simple type (§2.18.1).</td></tr>
<tr><td>cryptAlgorithmSid (Cryptographic Hashing Algorithm)</td><td>Specifies the specific cryptographic hashing algorithm which shall be used along with the salt attribute and user-supplied password in order to compute a hash value for comparison.

The possible values for this attribute shall be interpreted as follows:
<table>
<tr><th>Value</th><th>Algorithm</th></tr>
<tr><td>1</td><td>MD2</td></tr>
<tr><td>2</td><td>MD4</td></tr>
<tr><td>3</td><td>MD5</td></tr>
<tr><td>4</td><td>SHA-1</td></tr>
<tr><td>5</td><td>MAC</td></tr>
<tr><td>6</td><td>RIPEMD</td></tr>
<tr><td>7</td><td>RIPEMD-160</td></tr>
<tr><td>8</td><td>Undefined. Shall not be used.</td></tr>
<tr><td>9</td><td>HMAC</td></tr>
<tr><td>10</td><td>Undefined. Shall not be used.</td></tr>
<tr><td>11</td><td>Undefined. Shall not be used.</td></tr>
<tr><td>12</td><td>SHA-256</td></tr>
<tr><td>13</td><td>SHA-384</td></tr>
</table></td></tr>
</table>

Attributes	Description
	<table><tr><td>14</td><td>SHA-512</td></tr><tr><td>Any other value</td><td>Undefined. Shall not be used.</td></tr></table> [*Example*: Consider a WordprocessingML document with the following information stored in one of its protection elements: `<w:… w:cryptAlgorithmClass="hash" w:cryptAlgorithmType="typeAny" w:cryptAlgorithmSid="1" w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" />` The cryptAlgorithmSid attribute value of 1 specifies that the SHA-1 hashing algorithm shall be used to generate a hash from the user-defined password. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
cryptAlgorithmType (Cryptographic Algorithm Type)	Specifies the type of cryptographic algorithm used by this protection. [*Note*: The initial version of this Office Open XML Standard only supports a single type - typeAny - but future versions may expand this as necessary. *end note*] [*Example*: Consider a WordprocessingML document with the following information stored in one of its protection elements: `<w:… w:cryptAlgorithmClass="hash" w:cryptAlgorithmType="typeAny" w:cryptAlgorithmSid="1" w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" />` The cryptAlgorithmType attribute value of typeAny specifies that any type of algorithm may have been used for the password. *end example*] The possible values for this attribute are defined by the ST_AlgType simple type (§2.18.2).
cryptProvider (Cryptographic Provider)	Specifies the cryptographic provider which was used to generate the hash value stored in this document. If the user provided a cryptographic provider which was not the system's built-in provider, then that provider shall be stored here so it can subsequently be used if available. If this attribute is omitted, then the built-in cryptographic provider on the system shall be used. [*Example*: Consider a WordprocessingML document with the following information stored in one of its protection elements:

Attributes	Description
	<w:… w:cryptProvider="Krista'sProvider" w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" /> The cryptProvider attribute value of Krista'sProvider specifies that the cryptographic provider with name "Krista's Provider" shall be used if available. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).
cryptProviderType (Cryptographic Provider Type)	Specifies the type of cryptographic provider to be used. [*Example*: Consider a WordprocessingML document with the following information stored in one of its protection elements: <w:… w:cryptProviderType="rsaAES" w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" /> The cryptProviderType attribute value of rsaAES specifies that the cryptographic provider type shall be an Advanced Encryption Standard provider. *end example*] The possible values for this attribute are defined by the ST_CryptProv simple type (§2.18.14).
cryptProviderTypeExt (Cryptographic Provider Type Extensibility)	Specifies that a cryptographic provider type which was not defined by this Office Open XML Standard has been used to generate the hash value stored with this document. This value, when present, shall be interpreted based on the value of the cryptProviderTypeExtSource attribute in order to determine the provider type used, which shall be application-defined. [*Rationale*: This extensibility affords the fact that with exponentially increasing computing power, documents created in the future will likely need to utilize as yet undefined cryptographic provider types in order to remain secure. *end rationale*] If this value is present, the cryptProviderType attribute value shall be ignored in favor of the provider type defined by this attribute. [*Example*: Consider a WordprocessingML document with the following information stored in one of its protection elements: <w:… w:cryptProviderTypeExt="00A5691D" w:cryptProvideTypeExtSource="futureCryptography" w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" /> The cryptProviderTypeExt attribute value of 00A5691D specifies that the provider type associated with hex code A5691D shall be used as defined by the futureCryptography application. *end example*] The possible values for this attribute are defined by the ST_LongHexNumber simple type (§2.18.57).

Attributes	Description

cryptProviderTypeExtSource (Provider Type Extensibility Source)	Specifies the application which defined the provider type value specified by the cryptProviderTypeExt attribute.
	
	[*Example*: Consider a WordprocessingML document with the following information stored in one of its protection elements:
	
	`<w:… w:cryptProviderTypeExt="00A5691D"`
	`w:cryptProvideTypeExtSource="futureCryptography"`
	`w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" />`
	
	The cryptProvideTypeExtSource attribute value of `futureCryptography` specifies that the provider type used here was published by the `futureCryptography` application. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

cryptSpinCount (Iterations to Run Hashing Algorithm)	Specifies the number of times the hashing function shall be iteratively run (using each iteration's result as the input for the next iteration) when attempting to compare a user-supplied password with the value stored in the hash attribute. [*Rationale*: Running the algorithm many times increases the cost of exhaustive search attacks correspondingly. Storing this value allows for the number of iterations to be increased over time to accommodate faster hardware (and hence the ability to run more iterations in less time). *end rationale*]
	
	[*Example*: Consider a WordprocessingML document with the following information stored in one of its protection elements:
	
	`<w:… w:cryptSpinCount="100000"`
	`w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" />`
	
	The cryptSpinCount attribute value of `100000` specifies that the hashing function shall be run one hundred thousand times to generate a hash value for comparison with the hash attribute. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

hash (Password Hash)	Specifies the hash value for the password stored with this document. This value shall be compared with the resulting hash value after hashing the user-supplied password using the algorithm specified by the preceding attributes and parent XML element, and if the two values match, the protection shall no longer be enforced. If this value is omitted, then no password shall be associated with the protection, and it may be turned off without supplying any password. [*Example*: Consider a WordprocessingML document with the following information stored in one of its protection elements:

Attributes	Description
	<w:… w:cryptAlgorithmClass="hash" w:cryptAlgorithmType="typeAny" w:cryptAlgorithmSid="1" w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" /> The hash attribute value of 9oN7nWkCAyEZib1RomSJTjmPpCY= specifies that the user-supplied password shall be hashed using the pre-processing defined by the parent element (if any) followed by the SHA-1 algorithm (specified via the cryptAlgorithmSid attribute value of 1) and that the resulting has value must be 9oN7nWkCAyEZib1RomSJTjmPpCY= for the protection to be disabled. *end example*] The possible values for this attribute are defined by the XML Schema base64Binary datatype.
recommended (Recommend Write Protection in User Interface)	Specifies that applications should provide user interface recommending that the user open this document in write protected state. If the user chooses to do so, the document shall be write protected, otherwise, it shall be opened fully editable. If this attribute is omitted, then user interface recommending that the user open this document in write protected state should not be provided. If the password attribute is also specified, then this setting shall be ignored. [*Example*: Consider a WordprocessingML document which specifies that applications shall recommend write protection to this document. This requirement would be specified using the following WordprocessingML in the document settings: <w:writeProtection w:recommended="true" /> The recommended attribute has a value of true specifying that the applications shall hash any password provided, and if it matches this hash value, may only then halt enforcement of write protection. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
salt (Salt for Password Verifier)	Specifies the salt which was prepended to the user-supplied password before it was hashed using the hashing algorithm defined by the preceding attribute values to generate the hash attribute, and which shall also be prepended to the user-supplied password before attempting to generate a hash value for comparison. A *salt* is a random string which is added to a user-supplied password before it is hashed in order to prevent a malicious party from pre-calculating all possible password/hash combinations and simply using those precalculated values (often referred to as a "dictionary attack"). If this attribute is omitted, then no salt shall be prepended to the user-supplied password before it is hashed for comparison with the stored hash value. [*Example*: Consider a WordprocessingML document with the following information

Attributes	Description
	stored in one of its protection elements: <w:… w:salt="ZUdHa+D8F/OAKP3I7ssUnQ==" w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" /> The salt attribute value of ZUdHa+D8F/OAKP3I7ssUnQ== specifies that the user-supplied password shall have this value prepended before it is run through the specified hashing algorithm to generate a resulting hash value for comparison. *end example*] The possible values for this attribute are defined by the XML Schema base64Binary datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_WriteProtection">
   <attribute name="recommended" type="ST_OnOff" use="optional"/>
   <attributeGroup ref="AG_Password"/>
</complexType>
```

2.15.1.95 zoom (Magnification Setting)

This element specifies the magnification level which should be applied to a document when it is displayed by an application. The zoom level is specified with the use of two attributes stored on this element:

- val, which stores the type of zoom applied to the document
- percent, which stores the zoom percentage to be used when rendering the document

If both attributes are present, then the percent attribute shall be treated as a 'cached' value and only used when the value none is specified for the val attribute.

If this element is omitted, then applications may display the document in any desired magnification setting.

[*Example*: Consider a WordprocessingML document that is to have its zoom level at seventy one percent when it is displayed. This requirement would be specified using the following WordprocessingML fragment in the document settings:

```
<w:zoom w:percent="71" />
```

The zoom element's percent attribute has a value of 71, specifying that the given document shall have its zoom level set to seventy one percent when it is displayed. *end example*]

Parent Elements
settings (§2.15.1.78)

Attributes	Description

Attributes	Description
percent (Zoom Percentage)	Specifies the zoom percentage that should be applied when a given WordprocessingML document is rendered by conforming hosting applications. This value is the zoom percentage specified as an integer whose units correspond to the zoom percentage. If this attribute is omitted, then applications may use any desired default percentage for the magnification. If the val attribute instantiated in addition to the percent attribute, then the percent attribute shall be treated as a cached value and only used when the value none is specified for the val attribute. If the value specified exceeds the maximum zoom level available in a conforming hosting application, the conforming hosting application shall display the document using its maximum zoom level. Correspondingly, if the value specified is less than the minimum zoom level available in the conforming hosting application, the conforming hosting application shall display the document using its minimum zoom level. [*Example*: Consider a WordprocessingML document that is to have its zoom level at fifty percent when rendered by conforming hosting applications. This requirement would be specified using the following WordprocessingML: `<w:zoom w:percent="50" />` The percent attribute has a value of 50, specifying that the given WordprocessingML document shall to have its zoom level set to fifty percent when it is displayed. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).
val (Zoom Type)	Specifies the type of zoom which shall be applied to a given document on open. If this attribute is not present, then the document shall be displayed as though the value had been set to none, and should rely on the value of the percent attribute for the actual zoom percentage. [*Example*: Consider a WordprocessingML document that should be visible without any horizontal scrolling when it is displayed. This requirement would be specified using the following WordprocessingML: `<w:zoom w:val="bestFit" w:percent="90" />` The val attribute is equal to the value bestFit specifying that an application shall dynamically calculate the magnification needed such that the given document shall be visible on the horizontal plane of the document with no horizontal scrolling required to see any part of the WordprocessingML document's pages. Since both attributes are present, the percent attribute shall be treated as a 'cached'

Attributes	Description
	value and ignored. *end example*] The possible values for this attribute are defined by the ST_Zoom simple type (§2.18.116).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Zoom">
   <attribute name="val" type="ST_Zoom" use="optional"/>
   <attribute name="percent" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.15.2 Web Page Settings

The next group of settings stored in WordprocessingML is web page settings. These settings specify two categories of settings:

- Settings which are related to HTML documents (i.e. frameset definitions) that may be used in WordprocessingML documents as well
- All settings which affect how this document shall be handled when it is saved as HTML. Actually saving a document as HTML is outside of the scope of this Office Open XML Standard, but in order to ensure the maximum interoperability between a WordprocessingML document and an HTML document, settings not explicitly stored elsewhere are stored in these settings.

[*Example*: Consider the following WordprocessingML fragment for the web page settings in a WordprocessingML document:

```
<w:webSettings>
  <w:frameset>
    …
  </w:frameset>
  <w:doNotUseLongFileNames w:val="true" />
</w:webSettings>
```

The webSettings element contains all of the web page settings for this document. In this case, the web page settings specified for this document are: a frameset defined using the frameset element (§2.15.2.18); and a setting specifying that when this file is saved as a web page, all resulting files shall not exceed 8.3 characters in length using the doNotUseLongFileNames element (§2.15.2.13). *end example*]

2.15.2.1 allowPNG (Allow PNG as Graphic Format)

This element specifies that applications shall allow use of the PNG file format when the contents of this WordprocessingML document are saved as a web page. This includes all supporting images used as part of this HTML web page.

If this element is omitted from the document, then the PNG file format shall not be allowed when this document is saved as a web page, and that another suitable file format (such as the JPEG file format) should be utilized in its place.

[*Note*: This setting is intended for applications to save web pages which can be supported by legacy web browsers which do not support the reading of PNG images. However, although PNG utilizes a lossless compression algorithm, JPEG uses 'lossy' compression and may in some cases result in lower fidelity images. *end note*]

[*Example*: Consider a WordprocessingML document which contains the following content within the web settings part:

```
<w:webSettings>
  <w:allowPNG w:val="true" />
</w:webSettings>
```

The allowPNG element has a val attribute value of `true`, which specifies that applications may use the PNG graphic format as needed when saving this WordprocessingML document as a web page. *end example*]

Parent Elements
webSettings (§2.15.2.44)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.2.2 blockQuote (Data for HTML blockquote Element)

This element specifies that the current div element does not represent an HTML `div` element, but rather represents an HTML `blockquote` element. This element shall specify that this container shall be written out using the `blockquote` element if this document is subsequently saved as HTML.

If this element is omitted, then the current div element does not represent an HTML `blockquote` element. If both this element and the bodyDiv element (§2.15.2.3) are specified, then this element shall take precedence in all cases.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body style="margin-left:200px;margin-top:50px">
    <p>Paragraph one.</p>
    <blockquote style="border: 5px solid #00FFFF">
      <p>Paragraph in a blockquote.</p>
    </blockquote>
    <p>Paragraph two.</p>
  </body>
</html>
```

When this document is saved in the WordprocessingML format, the information stored on the `div`, `blockquote`, and body elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1626542603">
    …
    <w:divsChild>
      <w:div w:id="313534916">
        <w:blockQuote w:val="true" />
        …
      </w:div>
    </w:divsChild>
  </w:div>
</w:divs>
```

The blockQuote element has a val attribute value of `true`, which specifies that the nested div element actually represents a nested HTML `blockquote` when this document is resaved as HTML. *end example*]

Parent Elements
div (§2.15.2.6)

Attributes	Description

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.2.3 bodyDiv (Data for HTML body Element)

This element specifies that the current div element does not represent an HTML div element, but rather represents formatting properties on the HTML body element. This element shall specify that the properties specified by this container shall be written out onto the body element if this document is subsequently saved as HTML.

If this element is omitted, then the current div element does not represent an HTML body element. If both this element and the blockQuote element (§2.15.2.2) are specified, then this element shall be ignored. If this element is specified on any div which is not the main div element for the document, then this element shall be ignored.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body style="margin-left:200px;margin-top:50px">
    <p>Paragraph one.</p>
    <blockquote style="border: 5px solid #00FFFF">
      <p>Paragraph in a blockquote.</p>
    </blockquote>
    <p>Paragraph two.</p>
  </body>
</html>
```

When this document is saved in the WordprocessingML format, the information stored on the div, blockquote, and body elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1626542603">
    <w:bodyDiv w:val="true" />
    …
    <w:divsChild>
      …
    </w:divsChild>
  </w:div>
</w:divs>
```

The bodyDiv element has a val attribute value of true, which specifies that the div element actually represents properties on the HTML body when this document is resaved as HTML. *end example*]

Parent Elements
div (§2.15.2.6)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.2.4 bottom (Bottom Border for HTML div)

This element specifies the border which shall be displayed at the bottom of the boundaries of the current HTML div object.

If this element is omitted, then this HTML `div` object shall not have a bottom border.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body>
    <div style=" border-left-style:solid; border-right-style:groove; border-
right-width:1px; border-top-style:dashed; border-top-width:3px; border-bottom-
style:outset; border-bottom-width:3px">
      <p>paragraph of text</p>
    </div>
  </body>
</html>
```

This HTML would therefore normally appear as follows (image scaled appropriately):

Now, when this document is saved in the WordprocessingML format, the information stored on the `div` elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1785730240">
    <w:divBdr>
      <w:top w:val="dashed" w:sz="18" w:space="7" w:color="auto" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="auto" />
      <w:bottom w:val="outset" w:sz="18" w:color="auto" />
      <w:right w:val="threeDEngrave" w:sz="6" w:color="auto" />
    </w:divBdr>
  </w:div>
</w:divs>
```

The bottom element specifies border information about the bottom border for the single HTML `div` structure in the document; in this case, a 2.25 point bottom border of type `outset`. The initial 3 pixel border was converted to 2.25 points using the following logic:

$$3\text{px} * \frac{1 \text{ inch}}{96 \text{ px}} * \frac{576 \text{ eighth points}}{1 \text{ inch}} = 18 \text{ eighth points } (2.25 \text{ points})$$

end example]

Parent Elements
divBdr (§2.15.2.7)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or `auto` to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value `auto`, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is `true`, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect.

Attributes	Description
	[*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object.
	
	When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points.
	
	When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.
	
	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24/>`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*]
	
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).

sz (Border Width)	Specifies the width of the current border. If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value. If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value.

Attributes	Description

	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`
	
	The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	
	The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).

themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border.
	
	The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document.
	
	[*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup:
	
	`<w:top … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:bottom … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:left … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	`<w:right … w:color="FFA8A0" w:themeColor="accent2"`
	`w:themeTint="99" />`
	
	The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*]
	
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).

themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border.

Attributes	Description
	[*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255 = 102 = 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$L' = L * Shade_{percentage}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L' = 0.53 * 0.75 = 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$ and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from

Attributes	Description
	the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255 = 153 = 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `4F81BD`. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6) = 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get `95B3D7`. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).

Attributes	Description
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <attribute name="val" type="ST_Border" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
   <attribute name="space" type="ST_PointMeasure" use="optional"/>
   <attribute name="shadow" type="ST_OnOff" use="optional"/>
   <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.15.2.5 color (Frameset Splitter Color)

This element specifies the color of the splitters within the frameset in this WordprocessingML document. This element shall only be honored on the root frameset for this document, and may be ignored for all nested framesets in this document.

If this element is omitted, then the default color of the splitter may be automatically determined by the application displaying this WordprocessingML document (equivalent to a val attribute value of auto).

[*Example*: Consider a frameset consisting of the following three frames:

The following properties define the presentation of the splitter bars within this frameset:

```
<w:frameset>
  <w:framesetSplitbar>
    <w:w w:val="200" />
    <w:color w:val="0000FF" />
  </w:framesetSplitbar>
  …
</w:frameset>
```

The color element's val attribute specifies that the splitters shall be displayed in the RGB color `0000FF` (blue) when the contents of this document are displayed. *end example*]

Parent Elements
framesetSplitbar (§2.15.2.20)

Attributes	Description
themeColor (Run Content Theme Color)	Specifies a theme color which should be applied to the current run. The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows for color information to be set centrally in the document. If the themeColor attribute is specified, then the val attribute is ignored for this run. [*Example*: Consider a run of text which should be displayed using the `accent3` theme

Attributes	Description

	color from the document's Theme part. This requirement would be specified as follows in the resulting WordprocessingML:
	
	`<w:rPr>`
	`  <w:color w:themeColor="accent3" />`
	`</w:rPr>`
	
	The color attribute specifies that the run shall use the accent3 theme color. *end example*]
	
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).

themeShade (Run Content Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this run's contents.
	
	If the themeShade is supplied, then it is applied to the RGB value of the theme color to determine the final color applied to this run.
	
	The themeShade value is stored as a hex encoding of the shade value (from 0 to 255) applied to the current border.
	
	[*Example*: Consider a shade of 40% applied to a run in a document. This shade is calculated as follows:
	
	$$S_{xml} = 0.4 * 255$$
	$$= 102$$
	$$= 66(hex)$$
	
	Te resulting themeShade value in the file format would be 66. *end example*]
	
	Given a input red, green, or blue color value C (from 0-255), an output color value of C' (from 0-255), and a shade value S (from 0-100), the shade is applied as follows:
	
	$$C' = \left(1 - \frac{S}{100}\right) \mathrm{C}$$
	
	[*Example*: Consider a document with a run using the accent6 theme color, whose RGB value (in RRGGBB hex format) is F79646.
	
	The hex value for the green component is 96 - 150 in decimal. Applying the shade formula with shade of 50%, the output decimal value of the green component is 75, or a hex value of 4B. This transformed value can be seen in the resulting run color WordprocessingML's val attribute:
	
	`<w:color w:val="7B4B23" w:themeColor="accent6"`
	`w:themeShade="80" />`

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Run Content Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this run's contents. If the themeTint is supplied, then it is applied to the RGB value of the theme color to determine the final color applied to this run. The themeTint value is stored as a hex encoding of the tint value (from 0 to 255) applied to the current border. [*Example*: Consider a tint of 60% applied to a run in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255 = 153 = 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given a input red, green, or blue color value C (from 0-255), an output color value of C' (from 0-255), and a tint value T (from 0-100), the tint is applied as follows: $$C' = \left(1 - \frac{T}{100}\right)(255 - \mathrm{C}) + \mathrm{C}$$ [*Example*: Consider a document with a run using the accent1 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The hex value for the green component is 50 - 80 in decimal. Applying the tint formula with tint of 60%, the output decimal value of the green component is 150, or a hex value of 96. This transformed value can be seen in the resulting run color's WordprocessingML val attribute: `<w:color w:val="D99694" w:themeColor="accent1" w:themeTint="99" />` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Run Content Color)	Specifies the color for this run. This color may either be presented as a hex value (in RRGGBB format), or auto to allow a

Attributes	Description

	consumer to automatically determine the run color as appropriate.
	
	If the run specifies the use of a theme color via the themeColor attribute, then this value is superseded by the theme color value.
	
	[*Example*: Consider a run color with value `auto`, as follows:
	
	`<w:rPr>`
	`  <w:color ... w:val="auto" />`
	`</w:rPr>`
	
	This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the run contents can be distinguished against the page's background color. *end example*]
	
	The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <attribute name="val" type="ST_HexColor" use="required"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
</complexType>
```

2.15.2.6 div (Information About Single HTML div Element)

This element specifies information about a single HTML `div`, `body`, or `blockquote` element which was included in this document, so that that information (which is stored on a logical structure with no direct analog in WordprocessingML) may be maintained when an HTML document is stored in the WordprocessingML format.

The div element stores the following information about these structures:

- The child HTML `div`, and `blockquote` elements
- The borders for the element
- The margins for the element

When the resulting WordprocessingML document is displayed by an application, the settings specified by this information shall be reflected in the formatting of the resulting paragraphs (i.e. this information shall not only be used when the document is resaved in the HTML format).

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body>
    <div style="border-left-style: solid; border-left-width: 1px; border-right-
style: solid; border-right-width: 1px; padding-left: 4px; padding-right: 4px;
padding-top: 1px; padding-bottom: 1px; margin-left: 50px">
      <p>Paragraph one.</p>
      <p>Paragraph two.</p>
    </div>
  </body>
</html>
```

This HTML would therefore normally appear as follows (image scaled appropriately):

Now, when this document is saved in the WordprocessingML format, the information stored on the `div`, `blockquote`, and `body` elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1785730240">
    <w:marLeft w:val="750" />
    <w:divBdr>
      <w:left w:val="single" w:sz="6" w:color="auto" />
      <w:right w:val="single" w:sz="6" w:color="auto" />
    </w:divBdr>
  </w:div>
</w:divs>
```

The div element specifies all margin and border information about the single HTML `div` structures in the document; in this case, the left indentation and the left and right borders. *end example*]

Parent Elements
divs (§2.15.2.8); divsChild (§2.15.2.9)

Child Elements	Subclause
blockQuote (Data for HTML blockquote Element)	§2.15.2.2
bodyDiv (Data for HTML body Element)	§2.15.2.3

Child Elements	Subclause
divBdr (Set of Borders for HTML div)	§2.15.2.7
divsChild (Child div Elements Contained within Current div)	§2.15.2.9
marBottom (Bottom Margin for HTML div)	§2.15.2.23
marLeft (Left Margin for HTML div)	§2.15.2.25
marRight (Right Margin for HTML div)	§2.15.2.26
marTop (Top Margin for HTML div)	§2.15.2.27

Attributes	Description

id (div Data ID)	Specifies a unique decimal number which shall be used to associate one or more structures in the WordprocessingML content with this HTML `div` information.
	
	When a WordprocessingML structure (a paragraph or a table row) is associated with `div` information, it shall be associated with the set of information which most immediately contains the current object.
	
	[*Example*: If a paragraph is wrapped within two HTML div elements, like this:
	
	`<div>`
	`  <div>`
	`  <p>Paragraph</p>`
	`  </div>`
	`</div>`
	
	The resulting WordprocessingML paragraph shall reference the div Data ID associated with the inner HTML `div` element - the fact that it is also contained within the outer HTML `div` shall be implied by the nesting of the corresponding WordprocessingML div elements in the web settings part. *end example*]
	
	The ID specified by this attribute is then referenced by the divId element for all structures which are immediately contained within the specified HTML `div`.
	
	[*Example*: Consider a simple HTML document defined as follows:
	
	`<html>`
	`  <body style=" margin-top:50px">`
	`  <p>Paragraph one.</p`
	`  <div style="margin-left:50px">`
	`  <p>Paragraph two.</p>`
	`  </div>`
	`  </body>`
	`</html>`
	
	If the outer and inner body and `div` elements were assigned id attributes as follows:

Attributes	Description
	<w:divs> <w:div w:id="1626542603"> <w:bodyDiv w:val="1" /> … <w:divsChild> <w:div w:id="313534916"> … </w:div> </w:divsChild> </w:div> </w:divs> Then the first paragraph would reference the div ID of the outer div (since it is contained by the HTML body element) and the second paragraph would reference the div ID of the inner div (since it is contained within the child HTML div element), as follows: <w:p> <w:pPr> <w:divId w:val="1626542603" /> </w:pPr> <w:r> <w:t>Paragraph one.</w:t> </w:r> </w:p> <w:p> <w:pPr> <w:divId w:val="313534916" /> </w:pPr> <w:r> <w:t>Paragraph one.</w:t> </w:r> </w:p> The id attributes on the div elements link each paragraph with the corresponding container div element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Div">
   <sequence>
      <element name="blockQuote" type="CT_OnOff" minOccurs="0"/>
      <element name="bodyDiv" type="CT_OnOff" minOccurs="0"/>
      <element name="marLeft" type="CT_SignedTwipsMeasure"/>
      <element name="marRight" type="CT_SignedTwipsMeasure"/>
      <element name="marTop" type="CT_SignedTwipsMeasure"/>
      <element name="marBottom" type="CT_SignedTwipsMeasure"/>
      <element name="divBdr" type="CT_DivBdr" minOccurs="0"/>
      <element name="divsChild" type="CT_Divs" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="id" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.15.2.7 divBdr (Set of Borders for HTML div)

This element specifies the set of borders for the boundaries of the current HTML `div`, `body`, or `blockquote` element, using the four border types defined by its child elements.

If this element is omitted, then there shall be no borders associated with the current HTML v, `body`, or `blockquote` element.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body>
    <div style=" border-left-style:solid; border-right-style:groove; border-
right-width:3px; border-top-style:dashed; border-top-width:3px; border-bottom-
style:outset; border-bottom-width:3px">
      <p>paragraph of text</p>
    </div>
  </body>
</html>
```

This HTML would therefore normally appear as follows (image scaled appropriately):

Now, when this document is saved in the WordprocessingML format, the information stored on the `div` elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1785730240">
    <w:divBdr>
      <w:top w:val="dashed" w:sz="18" w:space="7" w:color="auto" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="auto" />
      <w:bottom w:val="outset" w:sz="18" w:color="auto" />
      <w:right w:val="threeDEngrave" w:sz="6" w:color="auto" />
    </w:divBdr>
  </w:div>
</w:divs>
```

The divBdr element specifies border information about the single HTML `div` structure in the document. *end example*]

Parent Elements
div (§2.15.2.6)

Child Elements	Subclause
bottom (Bottom Border for HTML div)	§2.15.2.4
left (Left Border for HTML div)	§2.15.2.21
right (Right Border for HTML div)	§2.15.2.35
top (Top Border for HTML div)	§2.15.2.42

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DivBdr">
   <sequence>
      <element name="top" type="CT_Border" minOccurs="0"/>
      <element name="left" type="CT_Border" minOccurs="0"/>
      <element name="bottom" type="CT_Border" minOccurs="0"/>
      <element name="right" type="CT_Border" minOccurs="0"/>
   </sequence>
</complexType>
```

2.15.2.8 divs (Information about HTML div Elements)

This element specifies all information about the set of HTML `div` elements (as well as the `body` and `blockquote` elements) which were included in this document, so that that information (which is stored on a logical structure with no direct analog in WordprocessingML) may be maintained when an HTML document is stored in the WordprocessingML format.

The divs element stores the following information about these structures:

- The parent/child structure of HTML `div`, `blockquote`, and `body` elements
- The borders for each of these elements

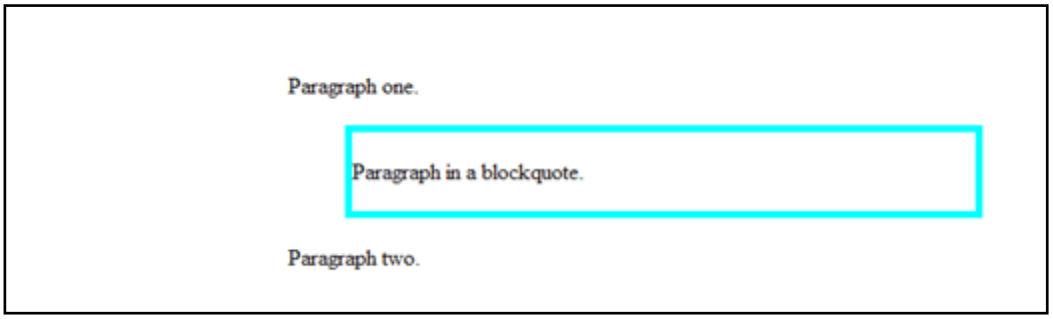
- The margins for each of these elements

When the resulting WordprocessingML document is displayed by an application, the settings specified by this information shall be reflected in the formatting of the resulting paragraphs (i.e. this information shall not only be used when the document is resaved in the HTML format).

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body style="margin-left:200px;margin-top:50px">
    <p>Paragraph one.</p>
    <blockquote style="border: 5px solid #00FFFF">
      <p>Paragraph in a blockquote.</p>
    </blockquote>
    <p>Paragraph two.</p>
  </body>
</html>
```

This HTML would therefore normally appear as follows (image scaled appropriately):



Now, when this document is saved in the WordprocessingML format, the information stored on the `div`, `blockquote`, and `body` elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1626542603">
    <w:bodyDiv w:val="1" />
    <w:marLeft w:val="3000" />
    <w:marTop w:val="750" />
    <w:divsChild>
      <w:div w:id="313534916">
        <w:blockQuote w:val="1" />
        <w:marLeft w:val="720" />
        <w:marRight w:val="720" />
        <w:marTop w:val="100" />
        <w:marBottom w:val="100" />
        <w:divBdr>
          <w:top w:val="single" w:sz="36" w:color="00FFFF" />
          <w:left w:val="single" w:sz="36" w:color="00FFFF" />
          <w:right w:val="single" w:sz="36" w:color="00FFFF" />
          <w:bottom w:val="single" w:sz="36" w:color="00FFFF" />
        </w:divBdr>
      </w:div>
    </w:divsChild>
  </w:div>
</w:divs>
```

The divs element specifies all of the margin and border information about the necessary HTML structures in the document; in this case, the body element and the nested blockquote. *end example*]

Parent Elements
webSettings (§2.15.2.44)

Child Elements	Subclause
div (Information About Single HTML div Element)	§2.15.2.6

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Divs">
   <sequence minOccurs="1" maxOccurs="unbounded">
      <element name="div" type="CT_Div"/>
   </sequence>
</complexType>
```

2.15.2.9 divsChild (Child div Elements Contained within Current div)

This element specifies the set of HTML div or blockquote elements which are contained within the current HTML div, body, or blockquote element, establishing the parent/child hierarchy of the original set of these elements.

When an HTML document containing these objects is saved in the WordprocessingML format, WordprocessingML objects store a reference to their most immediate parent `div`, `body`, or `blockquote` element using the divId element.

However, since only a single reference is stored, this information is often insufficient to determine the appropriate parent/child hierarchy for the original HTML `div` data, so it can be applied appropriately. This element allows that hierarchy to be stored, as child HTML `div` elements are stored within the childDivs element.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body>
    <div style=" margin-top:50px">
      <p>Paragraph one.</p
      <div style="margin-left:50px">
        <p>Paragraph two.</p>
      </div>
    </div>
  </body>
</html>
```

If the outer and inner body and `div` elements were assigned id attributes of 1626542603 and 313534916 respectively, then the first paragraph would reference the div ID of the outer div (since it is contained within that HTML `div` element) and the second paragraph would reference the div ID of the inner div (since it is contained within the child HTML `div` element), as follows:

```
<w:p>
  <w:pPr>
    <w:divId w:val="1626542603" />
  </w:pPr>
  <w:r>
    <w:t>Paragraph one.</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:divId w:val="313534916" />
  </w:pPr>
  <w:r>
    <w:t>Paragraph one.</w:t>
  </w:r>
</w:p>
```

However, this information alone is insufficient - it is unclear if the second `div` is contained within, or simply adjacent to, the first one.

In order to preserve this information, the correct hierarchy is stored within the web settings part:

```
<w:divs>
  <w:div w:id="1626542603">
    …
    <w:divsChild>
      <w:div w:id="313534916">
        …
      </w:div>
    </w:divsChild>
  </w:div>
</w:divs>
```

The divsChild element contains the second div as a child of the first div, specifying that the first div covers both paragraphs. *end example*]

Parent Elements
div (§2.15.2.6)

Child Elements	Subclause
div (Information About Single HTML div Element)	§2.15.2.6

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Divs">
   <sequence minOccurs="1" maxOccurs="unbounded">
      <element name="div" type="CT_Div"/>
   </sequence>
</complexType>
```

2.15.2.10 doNotOrganizeInFolder (Do Not Place Supporting Files in Subdirectory)

This element specifies that applications shall not automatically place all supporting files (images which are part of this HTML web page, etc.) in a subdirectory when the contents of this WordprocessingML document are saved as a web page. Typically, applications which save a document as a web page consisting of multiple files save all supporting files in a subdirectory next to the main HTML file (in order to keep those files organized). This element specifies the files shall be placed in the same directory as the actual web page.

If this element is omitted from the document, then all supporting files should be saved into a subdirectory beneath the main web page file when this document is saved as a web page.

[*Example*: Consider a WordprocessingML document which contains the following content within the web settings part:

```
<w:webSettings>
  <w:doNotOrganizeInFolder w:val="true" />
</w:webSettings>
```

The doNotOrganizeInFolder element has a val attribute value of true, which specifies that applications should save all supplementary files in the same directory as the main web page HTML document when saving this WordprocessingML document as a web page. *end example*]

Parent Elements
webSettings (§2.15.2.44)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.2.11 doNotRelyOnCSS (Do Not Rely on CSS for Font Face Formatting)

This element specifies whether applications may rely on the CSS properties for font face (the font-family property) when saving this WordprocessingML document as a web page. If this element is utilized, then the HTML font element should be used either in place of or in concert with these CSS properties in order to specify the font face formatting for the resulting web page.

If this element is omitted, then applications may choose to rely on the CSS properties for font face as desired.

[*Note*: This setting is intended for applications to save web pages which can be supported by legacy web browsers which do not support the reading of these CSS properties when attempting to read and display the resulting web page, in order to maximize the fidelity of the resulting output. *end note*]

[*Example*: Consider a WordprocessingML document which contains the following content within the web settings part:

```
<w:webSettings>
  <w:doNotRelyOnCSS w:val="true" />
</w:webSettings>
```

The doNotRelyOnCSS element has a val attribute value of `true`, which specifies that applications should include the HTML `font` element when saving this WordprocessingML document as a web page. For example, this output:

```
<span style='font-family:"Courier New"'>text</span>
```

This output would instead be saved as follows:

```
<font face="Courier New"><span style='font-family:"Courier
New"'>text</span></font>
```

end example]

Parent Elements
webSettings (§2.15.2.44)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.2.12 doNotSaveAsSingleFile (Recommend Web Page Format over Single File Web Page Format)

This element specifies that applications should recommend that new web page files generated using this WordprocessingML document use a multi-file web page format (HTML), rather than a single-file web page format (MHTML) when this document is saved as an HTML web page. This setting shall not prevent the use of the MHTML format; it shall only cause applications to recommend (via a default) a non single-file format when saving as a web page.

[*Note*: This setting is primarily intended for applications which explicitly support a "Save as Web Page…" action, in order to determine the default setting for the resulting web page. *end note*]

[*Example*: Consider a WordprocessingML document which contains the following content within the web settings part:

```
<w:webSettings>
  <w:doNotSaveAsSingleFile w:val="true" />
</w:webSettings>
```

The doNotSaveAsSingleFile element specifies that applications should recommend a multi-file web page format when this document is subsequently saved as a web page. *end example*]

Parent Elements
webSettings (§2.15.2.44)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.2.13 doNotUseLongFileNames (Do Not Use File Names Longer than 8.3 Characters)

This element specifies that applications shall ensure that the file names for all files generated when saving this document as a web page do not exceed eight characters with a three character extension. This includes all supporting files (images which are part of this HTML web page, etc.).

[*Note*: This setting is intended for applications to save web pages which can be supported by legacy web browsers which do not support the reading of long file names when attempting to read and display the resulting web page. *end note*]

[*Example*: Consider a WordprocessingML document which contains the following content within the web settings part:

```
<w:webSettings>
  <w:doNotUseLongFileNames w:val="true" />
</w:webSettings>
```

The doNotUseLongFileNames element specifies that applications should ensure that all file names generated when this document is subsequently saved as a web page do not exceed the 8.3 character file name limitation. *end example*]

Parent Elements
webSettings (§2.15.2.44)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.2.14 encoding (Output Encoding When Saving as Web Page)

This element specifies the encoding which shall be used for the contents of this WordprocessingML document when it is saved as an HTML web page. The set of encodings supported by this element shall be derived from the standard set of character set definitions provided at *http://www.iana.org/assignments/character-sets*.

If this element is omitted, then the default encoding for the current system shall be used when this document is saved as a web page. If the value of the val attribute is unknown or supported by an application, then the default encoding for the current system shall be used when this document is saved as a web page.

[*Example*: Consider a WordprocessingML document which contains the following content within the web settings part:

```
<w:webSettings>
  <w:encoding w:val="utf-8" />
</w:webSettings>
```

The encoding element's val attribute has a value of `utf-8`, which specifies that this document shall be encoded in the UTF-8 format when it is saved as a web page. *end example*]

Parent Elements
webSettings (§2.15.2.44)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`

Attributes	Description
	... </w:sdtPr> In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.15.2.15 flatBorders (Frameset Splitter Border Style)

This element specifies the 3D style of the splitters within the frameset in this WordprocessingML document. This element shall only be honored on the root frameset for this document, and may be ignored for all nested framesets in this document. When this property is turned on, the borders for this frameset shall be flat (not 3D), otherwise they may be presented as 3D splitter when they are displayed.

If this element is omitted, then the default style of the splitter should be a 3D splitter.

[*Example*: Consider a frameset consisting of the following three frames:

The following properties define the presentation of the splitter bars within this frameset:

```
<w:frameset>
  <w:framesetSplitbar>
    <w:w w:val="200" />
    <w:color w:val="0000FF" />
    <w:flatBorders w:val="true" />
  </w:framesetSplitbar>
  …
</w:frameset>
```

The flatBorders element's val attribute has a value of true, which specifies that the style of the splitters shall be flat (the splitter may not be 3D when displayed). *end example*]

Parent Elements
framesetSplitbar (§2.15.2.20)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

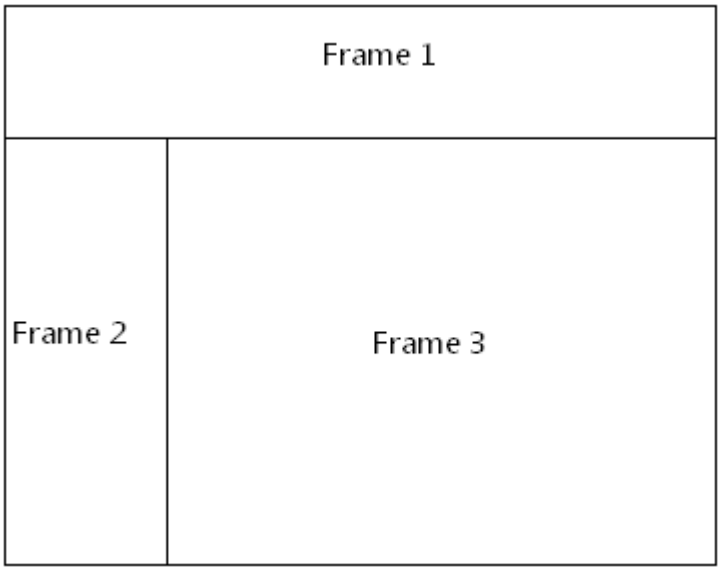
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.2.16 frame (Single Frame Properties)

This element specifies the properties for a single frame within a frameset document. When a document defines a frameset using the frameset element; that frameset is composed of a set of frames, each of which is specified by a single frame element.

[*Example*: Consider a WordprocessingML document which serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  …
  <w:frame>
    <w:sz w:val="20%" />
    <w:name w:val="Frame 1" />
    <w:sourceFileName r:id="rId1" />
  </w:frame>
  <w:frameset>
    …
  </w:frameset>
</w:frameset>
```

The frame element specifies the set of properties for a single frame in the document. In this case, these properties (for the frame marked with `Frame 1` in the diagram above) specify that the frame shall have the following properties:

- A height of `20%` of the height of the document
- A name of `Frame 1`
- The contents of the frame shall be pulled from the document that is the target of the relationship with ID `rId1`

end example]

Parent Elements
frameset (§2.15.2.18); frameset (§2.15.2.19)

Child Elements	Subclause
linkedToFile (Maintain Link to Existing File)	§2.15.2.22
marH (Top and Bottom Margin for Frame)	§2.15.2.24
marW (Left and Right Margin for Frame)	§2.15.2.28
name (Frame Name)	§2.15.2.29
noResizeAllowed (Frame Cannot Be Resized)	§2.15.2.31
scrollbar (Scrollbar Display Option)	§2.15.2.37
sourceFileName (Source File for Frame)	§2.15.2.38
sz (Frame Size)	§2.15.2.39

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Frame">
   <sequence>
      <element name="sz" type="CT_String" minOccurs="0"/>
      <element name="name" type="CT_String" minOccurs="0"/>
      <element name="sourceFileName" type="CT_Rel" minOccurs="0"/>
      <element name="marW" type="CT_PixelsMeasure" minOccurs="0"/>
      <element name="marH" type="CT_PixelsMeasure" minOccurs="0"/>
      <element name="scrollbar" type="CT_FrameScrollbar" minOccurs="0"/>
      <element name="noResizeAllowed" type="CT_OnOff" minOccurs="0"/>
      <element name="linkedToFile" type="CT_OnOff" minOccurs="0"/>
   </sequence>
</complexType>
```

2.15.2.17 frameLayout (Frameset Layout)

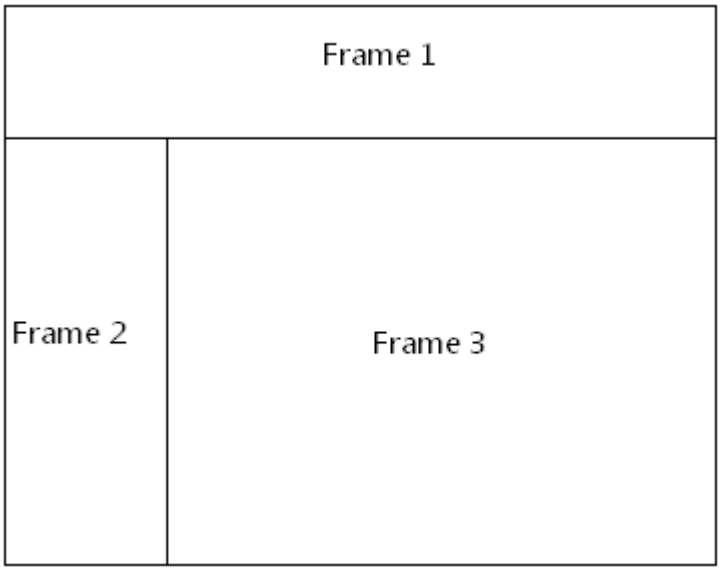
This element specifies the order in which the frames (and nested framesets) in a frameset shall be displayed. When a frameset is created, it can only contain frames which are stacked in one direction:

- Vertically (one on top of another)
- Horizontally (one next to another)

This element specifies how the frames in this frameset are stacked, which shall also be used to interpret the sizes defined by the sz element (§2.15.2.39) for each frame. In order to determine the ordering of the constituent frames within this frameset, the ordering of the child frame and frameset elements shall be used.

If this element is omitted, then the frames in this frameset shall be stacked vertically on top of one another (a row frameset).

[*Example*: Consider a WordprocessingML document which serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  <w:frameLayout w:val="rows" />
  <w:frame>
    …
  </w:frame>
  <w:frameset>
    <w:frameLayout w:val="cols" />
    <w:frame>
      …
    </w:frame>
    <w:frame>
      …
    </w:frame>
  </w:frameset>
</w:frameset>
```

The frameLayout element specifies that the outer frameset is a consists of the single frame and the child frameset stacked vertically, and an inner nested frameset consisting of two frames stacked horizontally. *end example*]

Parent Elements
frameset (§2.15.2.18); frameset (§2.15.2.19)

Attributes	Description

Attributes	Description

val (Frameset Layout Value)	Specifies the type of layout which shall be used to display the contents of the frames and nested framesets within this frameset, as defined by the simple type referenced below.
	
	[*Example*: Consider a frameset definition within a WordprocessingML document which defines the following frameset layout setting:
	
	`<w:frame>`
	`  <w:frameLayout w:val="cols" />`
	`  …`
	`</w:frame>`
	
	The val attribute value of `cols` specifies that the contents of this frameset shall be stacked horizontally (in columns). *end example*]
	
	The possible values for this attribute are defined by the ST_FrameLayout simple type (§2.18.35).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FrameLayout">
   <attribute name="val" type="ST_FrameLayout" use="required"/>
</complexType>
```

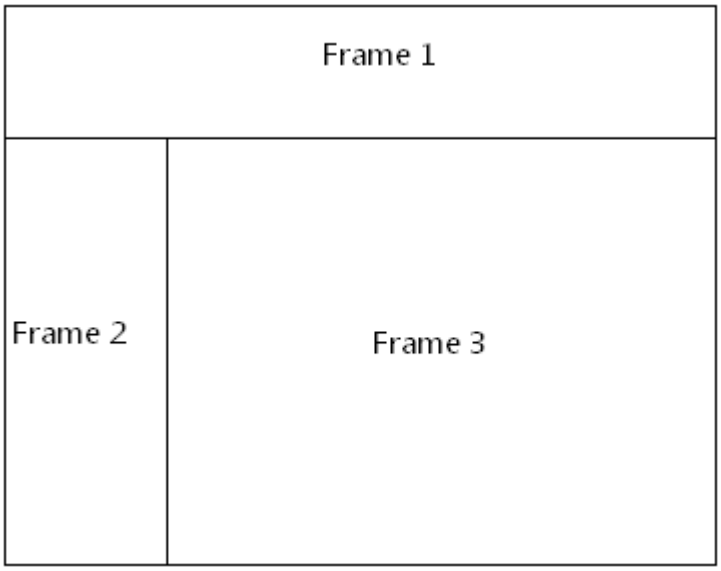
2.15.2.18 frameset (Root Frameset Definition)

This element specifies that this document is the container for a frameset. This WordprocessingML element is analogous to the `frameset` element in HTML.

When the frameset element is present within a document, that document shall serve as a frameset definition only; all of its normal document content shall therefore not be displayed as long as it contains at least one child frame or frameset element.

If this element is omitted, then the currently document shall not be treated as a frameset definition; its regular document content shall be displayed.

[*Example*: Consider a WordprocessingML document which serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  <w:frameLayout w:val="rows" />
  <w:frame>
    …
  </w:frame>
  <w:frameset>
    <w:frameLayout w:val="cols" />
    <w:frame>
      …
    </w:frame>
    <w:frame>
      …
    </w:frame>
  </w:frameset>
</w:frameset>
```

The parent frameset element specifies that the current document is a frameset definition; that frameset consists of a single frame and another nested frameset stacked vertically. *end example*]

Parent Elements
webSettings (§2.15.2.44)

Child Elements	Subclause
frame (Single Frame Properties)	§2.15.2.16

Child Elements	Subclause
frameLayout (Frameset Layout)	§2.15.2.17
frameset (Nested Frameset Definition)	§2.15.2.19
framesetSplitbar (Frameset Splitter Properties)	§2.15.2.20
sz (Nested Frameset Size)	§2.15.2.40

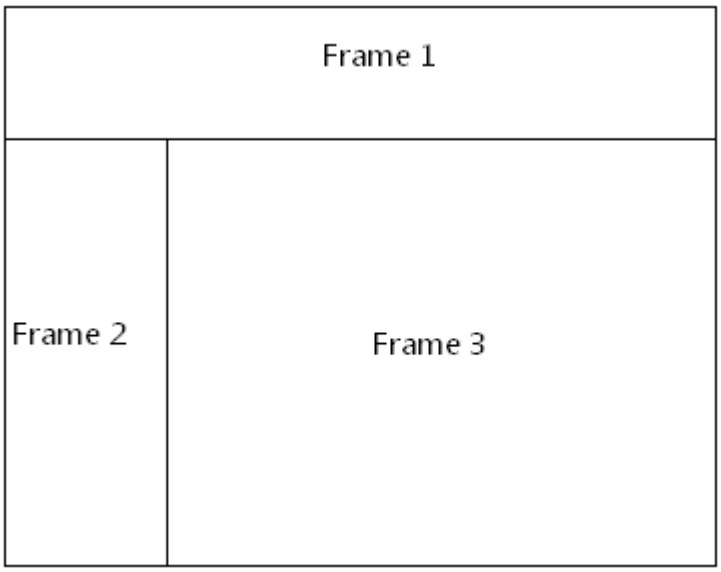
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Frameset">
   <sequence>
      <element name="sz" type="CT_String" minOccurs="0"/>
      <element name="framesetSplitbar" type="CT_FramesetSplitbar" minOccurs="0"/>
      <element name="frameLayout" type="CT_FrameLayout" minOccurs="0"/>
      <choice minOccurs="0" maxOccurs="unbounded">
         <element name="frameset" type="CT_Frameset" minOccurs="0" maxOccurs="unbounded"/>
         <element name="frame" type="CT_Frame" minOccurs="0" maxOccurs="unbounded"/>
      </choice>
   </sequence>
</complexType>
```

2.15.2.19 frameset (Nested Frameset Definition)

This element specifies a frameset which has been nested within another frameset within a WordprocessingML document. This WordprocessingML element is analogous to the `frameset` element in HTML (when that frameset is the child of another `frameset` element).

[*Example*: Consider a WordprocessingML document which serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  <w:frameLayout w:val="rows" />
  <w:frame>
    …
  </w:frame>
  <w:frameset>
    <w:frameLayout w:val="cols" />
    <w:frame>
      …
    </w:frame>
    <w:frame>
      …
    </w:frame>
  </w:frameset>
</w:frameset>
```

The child frameset element specifies the frameset definition for the inner frameset; that frameset consists of two frames (`Frame 2` and `Frame 3` in the image above) which have been laid out horizontally as columns. *end example*]

Parent Elements
frameset (§2.15.2.18); frameset (§2.15.2.19)

Child Elements	Subclause
frame (Single Frame Properties)	§2.15.2.16
frameLayout (Frameset Layout)	§2.15.2.17
frameset (Nested Frameset Definition)	§2.15.2.19
framesetSplitbar (Frameset Splitter Properties)	§2.15.2.20
sz (Nested Frameset Size)	§2.15.2.40

The following XML Schema fragment defines the contents of this element:

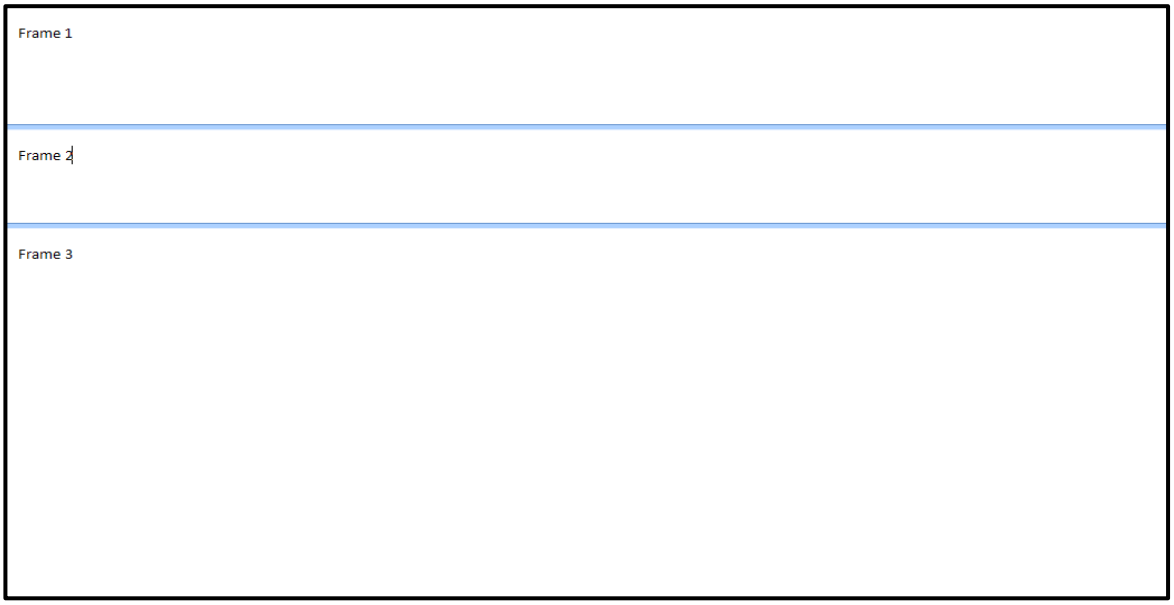
```
<complexType name="CT_Frameset">
   <sequence>
      <element name="sz" type="CT_String" minOccurs="0"/>
      <element name="framesetSplitbar" type="CT_FramesetSplitbar" minOccurs="0"/>
      <element name="frameLayout" type="CT_FrameLayout" minOccurs="0"/>
      <choice minOccurs="0" maxOccurs="unbounded">
         <element name="frameset" type="CT_Frameset" minOccurs="0" maxOccurs="unbounded"/>
         <element name="frame" type="CT_Frame" minOccurs="0" maxOccurs="unbounded"/>
      </choice>
   </sequence>
</complexType>
```

2.15.2.20 framesetSplitbar (Frameset Splitter Properties)

This element specifies the properties for the splitters associated with this frameset. A *splitter* is a horizontal or vertical line which visually separates the contents of one frame from another within a frameset.

If this element is omitted, then the default parameters for each of the child frameset properties shall be used for all splitters in this frameset.

[*Example*: Consider a frameset consisting of the following three frames:



The following properties define the presentation of the splitter bars within this frameset:

```
<w:frameset>
  <w:framesetSplitbar>
    <w:w w:val="90" />
    <w:color w:val="auto" />
  </w:framesetSplitbar>
  …
</w:frameset>
```

The framesetSplitbar element specifies the properties for all splitters in this frameset; in this case, those properties are that the splitter shall be 4.5 points (90 twentieths of a point) wide, and that the color of the splitter shall be automatically determined via the attribute value of auto. *end example*]

Parent Elements
frameset (§2.15.2.18); frameset (§2.15.2.19)

Child Elements	Subclause

Child Elements	Subclause
color (Frameset Splitter Color)	§2.15.2.5
flatBorders (Frameset Splitter Border Style)	§2.15.2.15
noBorder (Do Not Display Frameset Splitters)	§2.15.2.30
w (Frameset Splitter Width)	§2.15.2.43

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FramesetSplitbar">
   <sequence>
      <element name="w" type="CT_TwipsMeasure" minOccurs="0"/>
      <element name="color" type="CT_Color" minOccurs="0"/>
      <element name="noBorder" type="CT_OnOff" minOccurs="0"/>
      <element name="flatBorders" type="CT_OnOff" minOccurs="0"/>
   </sequence>
</complexType>
```

2.15.2.21 left (Left Border for HTML div)

This element specifies the border which shall be displayed at the left of the boundaries of the current HTML `div` object.

If this element is omitted, then this HTML `div` object shall not have a left border.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body>
    <div style=" border-left-style:solid; border-right-style:groove; border-
right-width:1px; border-top-style:dashed; border-top-width:3px; border-bottom-
style:outset; border-bottom-width:3px">
      <p>paragraph of text</p>
    </div>
  </body>
</html>
```

This HTML would therefore normally appear as follows (image scaled appropriately):

Now, when this document is saved in the WordprocessingML format, the information stored on the `div` elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1785730240">
    <w:divBdr>
      <w:top w:val="dashed" w:sz="18" w:space="7" w:color="auto" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="auto" />
      <w:bottom w:val="outset" w:sz="18" w:color="auto" />
      <w:right w:val="threeDEngrave" w:sz="6" w:color="auto" />
    </w:divBdr>
  </w:div>
</w:divs>
```

The left element specifies border information about the left border for the single HTML div structure in the document; in this case, a 3 point bottom border of type single. The initial 4 pixel border was converted to 3 points using the following logic:

$$4\text{px} * \frac{1 \text{ inch}}{96 \text{ px}} * \frac{576 \text{ eighth points}}{1 \text{ inch}} = 24 \text{ eighth points (3 points)}$$

end example]

Parent Elements
divBdr (§2.15.2.7)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or auto to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value auto, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create	Specifies whether the specified border should be modified to create a frame effect by

Attributes	Description
Frame Effect)	reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is `true`, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object. When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points. When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points. [*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:

Attributes	Description
	<w:pgBorders w:offsetFrom="page"> <w:bottom … w:space="24/> </w:pgBorders> The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*] The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).
sz (Border Width)	Specifies the width of the current border. If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value. If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value. [*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup: <w:top w:val="dashed" w:sz="24" …/> <w:left w:val="dashed" w:sz="24" …/> <w:bottom w:val="dashed" w:sz="24" …/> <w:right w:val="dashed" w:sz="24" …/> The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*] The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).
themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border. The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document. [*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup: <w:top … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />

Attributes	Description
	<w:bottom … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:left … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:right … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*] The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).
themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255 = 102 = 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$.

Attributes	Description
	Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L^{'} = 0.53 * 0.75$$ $$= 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255$$ $$= 153$$ $$= 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}^{'} = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color,

Attributes	Description
	whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $L' = 0.53 * 0.6 + (1 - .6)$ $= 0.71$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is single, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
    <attribute name="val" type="ST_Border" use="required"/>
    <attribute name="color" type="ST_HexColor" use="optional"/>
    <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
    <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
    <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
    <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
    <attribute name="space" type="ST_PointMeasure" use="optional"/>
    <attribute name="shadow" type="ST_OnOff" use="optional"/>
    <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

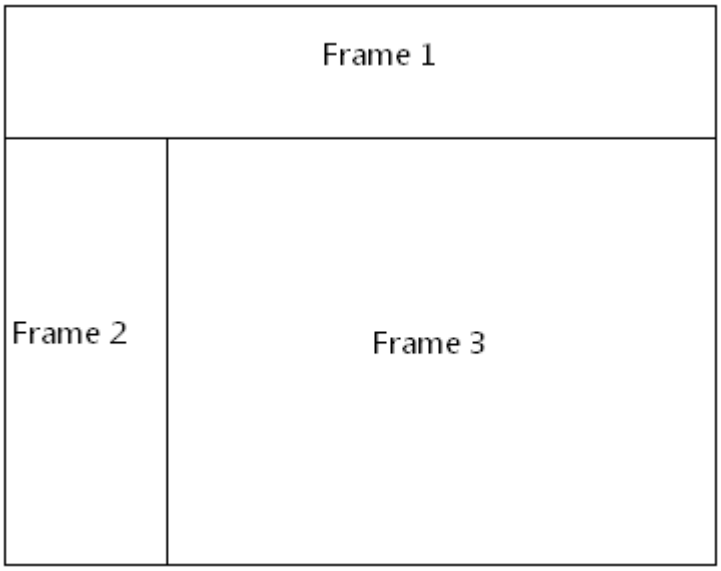
2.15.2.22 linkedToFile (Maintain Link to Existing File)

This element specifies that the file referenced by the sourceFileName element (§2.15.2.38) as the basis for the current frame shall not be changed, even when the file defined by the parent frameset is moved - i.e. the link shall remain exactly as specified.

[*Guidance*: Typically, when a document is incorporated into a frameset, a copy of that document is made such that all files encompassing the frameset are stored in a single subdirectory (so they can be moved as a single unit). However, if the link to the current file is absolute and shall not be changed even when the location of the main frameset document, then this element shall be set to indicate that setting. *end guidance*]

If this element is omitted, then a new file may be created as necessary when the parent frameset document is resaved to another location.

[*Example*: Consider a WordprocessingML document which serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  …
  <w:frame>
    <w:sz w:val="20%" />
    <w:name w:val="Frame 1" />
    <w:sourceFileName r:id="rId1" />
    <w:linkedToFile w:val="true" />
  </w:frame>
  <w:frameset>
    …
  </w:frameset>
</w:frameset>
```

The linkedToFile element specifies that the frame source location specified by the sourceFileName element (§2.15.2.38) shall not be modified, even if the parent frameset document is resaved to another location. *end example*]

Parent Elements
frame (§2.15.2.16)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.2.23 marBottom (Bottom Margin for HTML div)

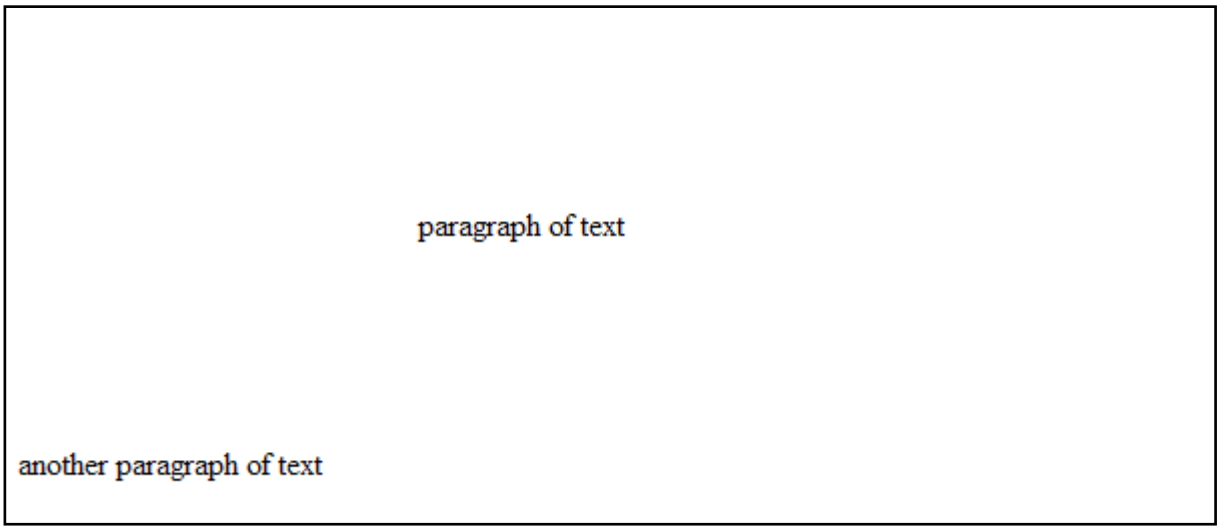
This element specifies the margin which shall be displayed at the bottom of the boundaries of the current HTML `div` object.

If this element is omitted, then this HTML `div` object shall not have a bottom margin.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body>
    <div style="margin-top:100px; margin-left:200px; margin-right:50px; margin-
bottom:100px">
      <p>paragraph of text</p>
    </div>
    <p>another paragraph of text</p>
  </body>
</html>
```

This HTML would therefore normally appear as follows (image scaled appropriately):



Now, when this document is saved in the WordprocessingML format, the information stored on the `div` elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1785730240">
    <w:marLeft w:val="3000" />
    <w:marRight w:val="750" />
    <w:marTop w:val="1500" />
    <w:marBottom w:val="1500" />
  </w:div>
</w:divs>
```

The marBottom element specifies margin information about the bottom margin for the single HTML `div` structure in the document; in this case, a 75 point bottom margin. The initial 100 pixel margin was converted to 75 points using the following logic:

$$100\text{px} * \frac{1 \text{ inch}}{96 \text{ px}} * \frac{1440 \text{ twentieth points}}{1 \text{ inch}} = 1500 \text{ twentieth points (75 points)}$$

end example]

Parent Elements
div (§2.15.2.6)

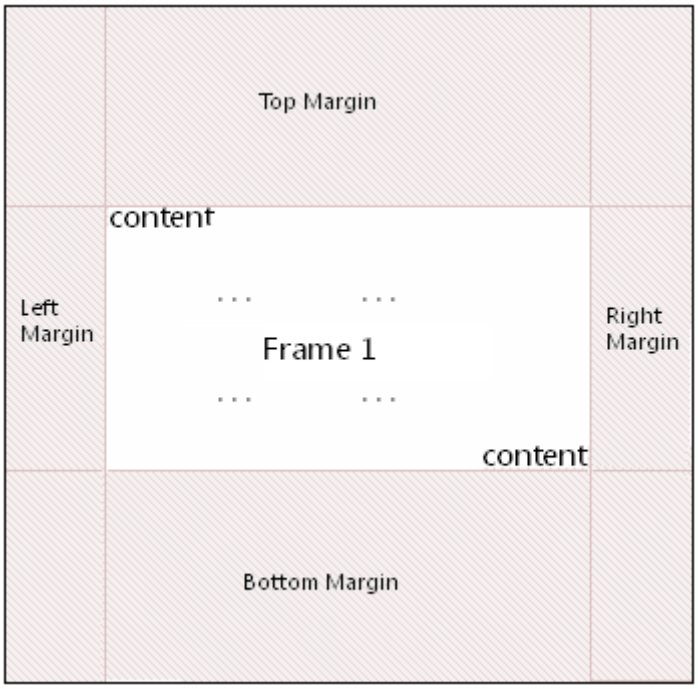
Attributes	Description
val (Positive or Negative Value in Twentieths of a Point)	Specifies a value whose contents shall contain a positive whole number, whose contents consist of a positive or negative measurement in twentieths of a point (equivalent to 1/1440th of an inch). The contents of this measurement shall be interpreted based on the context of the parent XML element. [*Example*: Consider an attribute value of `-720` whose type is ST_SignedTwipsMeasure. This attribute value specifies a value of negative one-half of an inch or -36 points (`-720` twentieths of a point = -36 points = -0.5 inches). *end example*] The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§2.18.88).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SignedTwipsMeasure">
   <attribute name="val" type="ST_SignedTwipsMeasure" use="required"/>
</complexType>
```

2.15.2.24 marH (Top and Bottom Margin for Frame)

This element specifies the top and bottom margin height for a single frame in a frameset document, as follows:



This height is expressed in pixels.

If this element is omitted, then no top or bottom margin shall be used for this frame.

[*Example*: Consider a document that has a frame, where the margin height has been specified and is represented as the following WordprocessingML:

```
<w:frame>
  <w:marH w:val="594"/>
</w:frame>
```

The marH element has a val attribute value of 594, which specifies that this frame has a top and bottom margin value of 594 pixels, resulting in 594 pixels of space between the content and the top and bottom margins of the frame. *end example*]

Parent Elements
frame (§2.15.2.16)

Attributes	Description
val (Measurement in Pixels)	Specifies a value whose contents shall contain a positive whole number, whose contents consist of a positive measurement in pixels. The contents of this measurement shall be interpreted based on the context of the parent XML element.

Attributes	Description
	[*Example*: Consider an attribute value of `960` whose type is ST_PixelsMeasure. This attribute value specifies a value of 960 pixels. *end example*] The possible values for this attribute are defined by the ST_PixelsMeasure simple type (§2.18.74).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PixelsMeasure">
   <attribute name="val" type="ST_PixelsMeasure" use="required"/>
</complexType>
```

2.15.2.25 marLeft (Left Margin for HTML div)

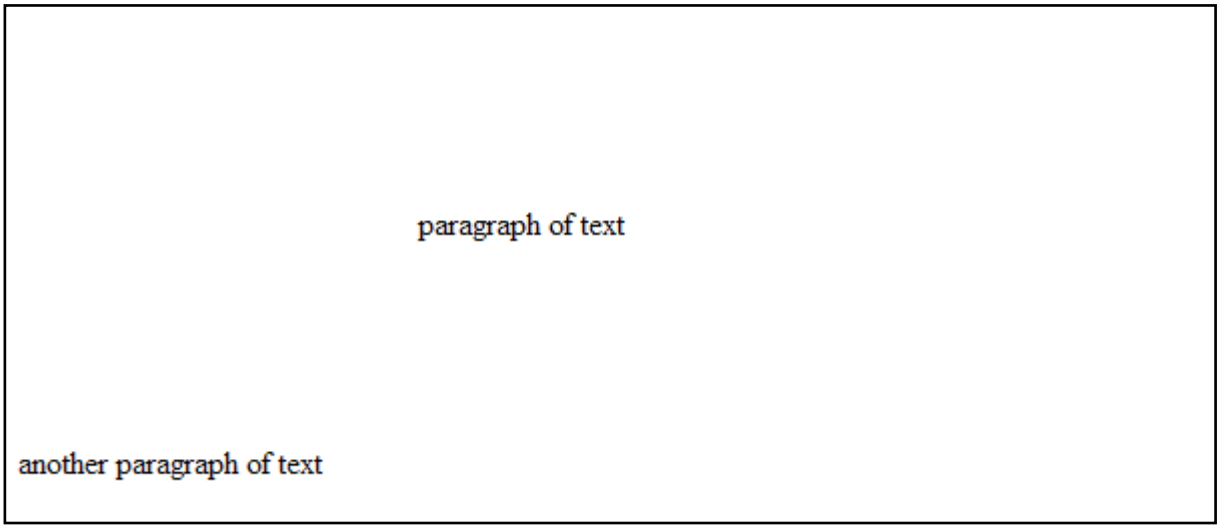
This element specifies the margin which shall be displayed at the left of the boundaries of the current HTML `div` object.

If this element is omitted, then this HTML `div` object shall not have a left margin.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body>
    <div style="margin-top:100px; margin-left:200px; margin-right:50px; margin-
bottom:100px">
      <p>paragraph of text</p>
    </div>
    <p>another paragraph of text</p>
  </body>
</html>
```

This HTML would therefore normally appear as follows (image scaled appropriately):



Now, when this document is saved in the WordprocessingML format, the information stored on the div elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1785730240">
    <w:marLeft w:val="3000" />
    <w:marRight w:val="750" />
    <w:marTop w:val="1500" />
    <w:marBottom w:val="1500" />
  </w:div>
</w:divs>
```

The marLeft element specifies margin information about the left margin for the single HTML div structure in the document; in this case, a 150 point left margin. The initial 200 pixel margin was converted to 150 points using the following logic:

$$200\text{px} * \frac{1\text{ inch}}{96\text{ px}} * \frac{1440\text{ twentieth points}}{1\text{ inch}} = 3000\text{ twentieth points } (150\text{ points})$$

end example]

Parent Elements
div (§2.15.2.6)

Attributes	Description
val (Positive or Negative Value in Twentieths of a Point)	Specifies a value whose contents shall contain a positive whole number, whose contents consist of a positive or negative measurement in twentieths of a point (equivalent to 1/1440th of an inch). The contents of this measurement shall be interpreted based on the context of the parent XML element. [*Example*: Consider an attribute value of -720 whose type is ST_SignedTwipsMeasure. This attribute value specifies a value of negative one-half of an inch or -36 points (-720 twentieths of a point = -36 points = -0.5 inches). *end example*] The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§2.18.88).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SignedTwipsMeasure">
   <attribute name="val" type="ST_SignedTwipsMeasure" use="required"/>
</complexType>
```

2.15.2.26 marRight (Right Margin for HTML div)

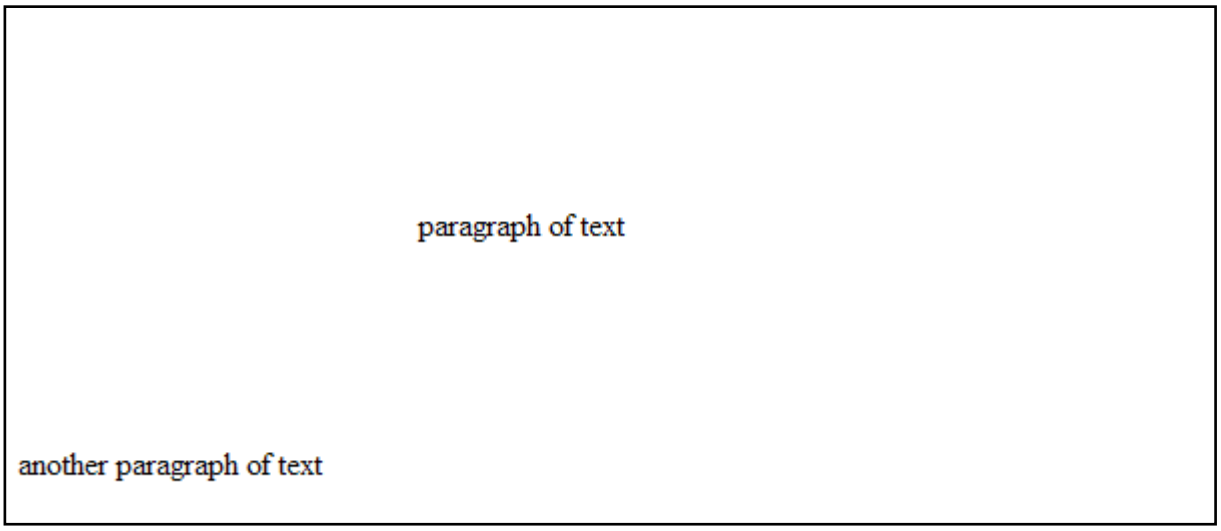
This element specifies the margin which shall be displayed at the right of the boundaries of the current HTML `div` object.

If this element is omitted, then this HTML `div` object shall not have a right margin.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body>
    <div style="margin-top:100px; margin-left:200px; margin-right:50px; margin-
bottom:100px">
      <p>paragraph of text</p>
    </div>
    <p>another paragraph of text</p>
  </body>
</html>
```

This HTML would therefore normally appear as follows (image scaled appropriately):



Now, when this document is saved in the WordprocessingML format, the information stored on the `div` elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1785730240">
    <w:marLeft w:val="3000" />
    <w:marRight w:val="750" />
    <w:marTop w:val="1500" />
    <w:marBottom w:val="1500" />
  </w:div>
</w:divs>
```

The marRight element specifies margin information about the right margin for the single HTML `div` structure in the document; in this case, a 37.5 point right margin. The initial 50 pixel margin was converted to 37.5 points using the following logic:

$$50\text{px} * \frac{1 \text{ inch}}{96 \text{ px}} * \frac{1440 \text{ twentieth points}}{1 \text{ inch}} = 750 \text{ twentieth points (37.5 points)}$$

end example]

Parent Elements
div (§2.15.2.6)

Attributes	Description
val (Positive or Negative Value in Twentieths of a Point)	Specifies a value whose contents shall contain a positive whole number, whose contents consist of a positive or negative measurement in twentieths of a point (equivalent to 1/1440th of an inch). The contents of this measurement shall be interpreted based on the context of the parent XML element. [*Example*: Consider an attribute value of `-720` whose type is ST_SignedTwipsMeasure. This attribute value specifies a value of negative one-half of an inch or -36 points (`-720` twentieths of a point = -36 points = -0.5 inches). *end example*] The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§2.18.88).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SignedTwipsMeasure">
   <attribute name="val" type="ST_SignedTwipsMeasure" use="required"/>
</complexType>
```

2.15.2.27 marTop (Top Margin for HTML div)

This element specifies the margin which shall be displayed at the top of the boundaries of the current HTML `div` object.

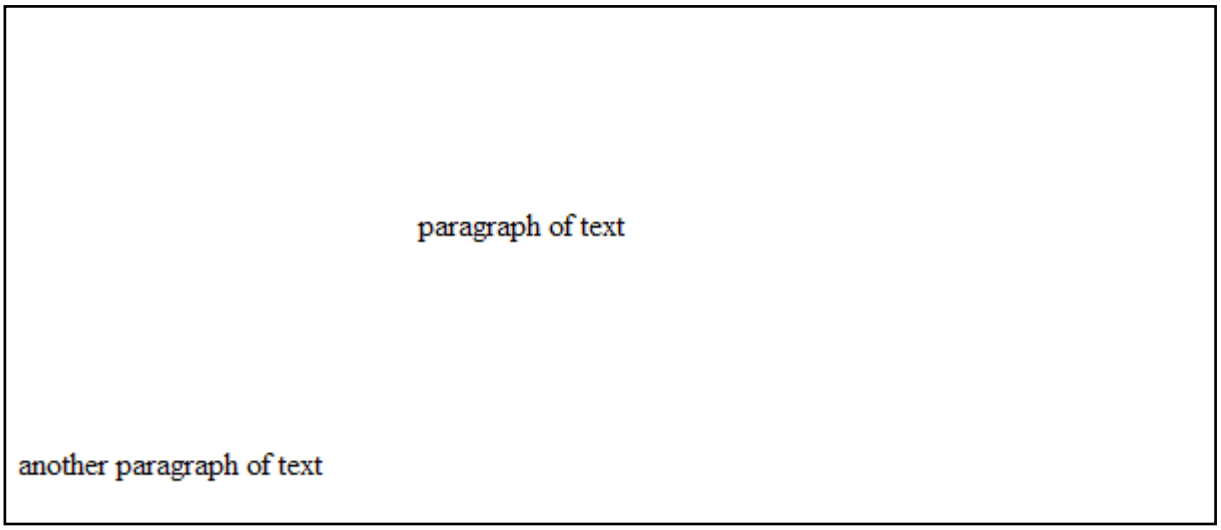
If this element is omitted, then this HTML `div` object shall not have a top margin.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body>
    <div style="margin-top:100px; margin-left:200px; margin-right:50px; margin-
bottom:100px">
      <p>paragraph of text</p>
```

```
    </div>
    <p>another paragraph of text</p>
  </body>
</html>
```

This HTML would therefore normally appear as follows (image scaled appropriately):



Now, when this document is saved in the WordprocessingML format, the information stored on the `div` elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1785730240">
    <w:marLeft w:val="3000" />
    <w:marRight w:val="750" />
    <w:marTop w:val="1500" />
    <w:marBottom w:val="1500" />
  </w:div>
</w:divs>
```

The marTop element specifies margin information about the top margin for the single HTML `div` structure in the document; in this case, a 75 point top margin. The initial 100 pixel margin was converted to 75 points using the following logic:

$$100\text{px} * \frac{1 \text{ inch}}{96 \text{ px}} * \frac{1440 \text{ twentieth points}}{1 \text{ inch}} = 1500 \text{ twentieth points (75 points)}$$

end example]

Parent Elements
div (§2.15.2.6)

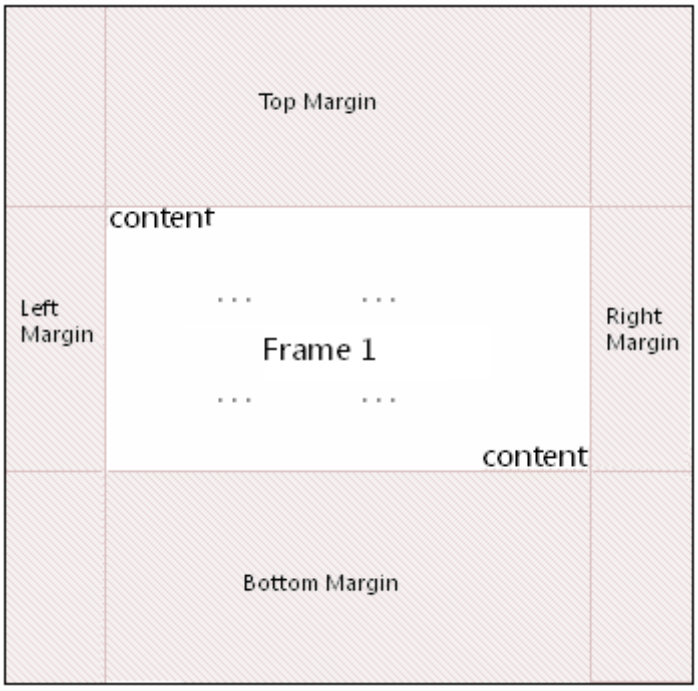
Attributes	Description
val (Positive or Negative Value in Twentieths of a Point)	Specifies a value whose contents shall contain a positive whole number, whose contents consist of a positive or negative measurement in twentieths of a point (equivalent to 1/1440th of an inch). The contents of this measurement shall be interpreted based on the context of the parent XML element. [*Example*: Consider an attribute value of `-720` whose type is ST_SignedTwipsMeasure. This attribute value specifies a value of negative one-half of an inch or -36 points (`-720` twentieths of a point = -36 points = -0.5 inches). *end example*] The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§2.18.88).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SignedTwipsMeasure">
   <attribute name="val" type="ST_SignedTwipsMeasure" use="required"/>
</complexType>
```

2.15.2.28 marW (Left and Right Margin for Frame)

This element specifies the left and right margin height for a single frame in a frameset document, as follows:



This height is expressed in pixels.

If this element is omitted, then no left or right margin shall be used for this frame.

[*Example*: Consider a document that has a frame, where the frame's margins have been specified and is represented as the following WordprocessingML:

```
<w:frame>
  <w:marW w:val="294"/>
</w:frame>
```

The marW element has a val attribute value of 294, which specifies that this frame has a left and right margin value of 294 pixels, resulting in 294 pixels of space between the content and the left and right margins of the frame. *end example*]

Parent Elements
frame (§2.15.2.16)

Attributes	Description
val (Measurement in Pixels)	Specifies a value whose contents shall contain a positive whole number, whose contents consist of a positive measurement in pixels. The contents of this measurement shall be interpreted based on the context of the parent XML element. [*Example*: Consider an attribute value of 960 whose type is ST_PixelsMeasure. This attribute value specifies a value of 960 pixels. *end example*] The possible values for this attribute are defined by the ST_PixelsMeasure simple type (§2.18.74).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PixelsMeasure">
   <attribute name="val" type="ST_PixelsMeasure" use="required"/>
</complexType>
```

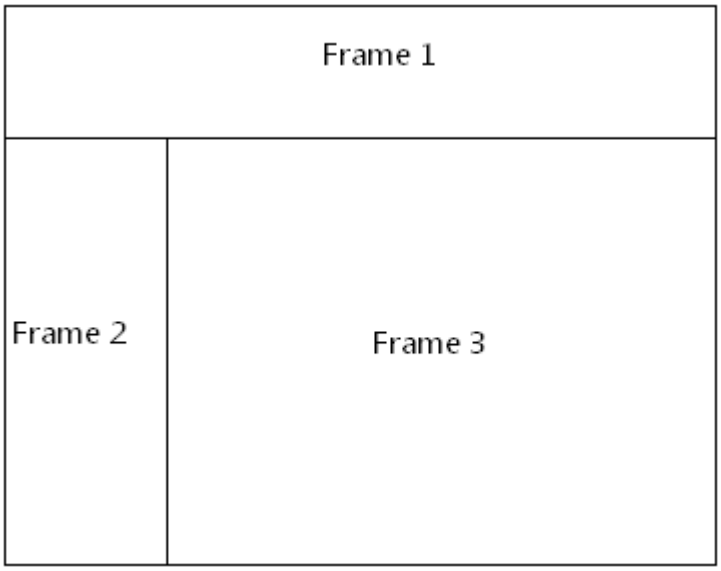
2.15.2.29 name (Frame Name)

This element specifies the name of a single frame within a frameset document. This property is analogous to the name attribute on the frame element in HTML.

[*Note*: The name of a frame may be used in web pages that reference a frame via targeted links, etc. *end note*]

If this element is omitted, then the current frame shall have no name associated with it.

[*Example*: Consider a WordprocessingML document which serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  …
  <w:frame>
    <w:name w:val="Frame 1" />
  </w:frame>
  <w:frameset>
    …
    <w:frame>
      <w:name w:val="Frame 2" />
    </w:frame>
    <w:frame>
      <w:name w:val="Frame 3" />
    </w:frame>
  </w:frameset>
</w:frameset>
```

The name element specifies the name for each frame within this frameset; in this case, the frames have names of `Frame 1`, `Frame 2`, and `Frame 3` respectively. *end example*]

Parent Elements
frame (§2.15.2.16)

Attributes	Description
val (String Value)	Specifies that its contents will contain a string.

Attributes	Description

	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.15.2.30 noBorder (Do Not Display Frameset Splitters)

This element specifies whether the splitters shall be displayed for the contents of the frameset in this WordprocessingML document. This element shall only be honored on the root frameset for this document, and may be ignored for all nested framesets in this document. If this element is present, then no splitters shall be displayed, and all other frameset splitter properties may be ignored.

If this element is omitted, then the splitters in this document shall be displayed as defined by the w and color elements.

[*Example*: Consider a frameset consisting of the following three frames:

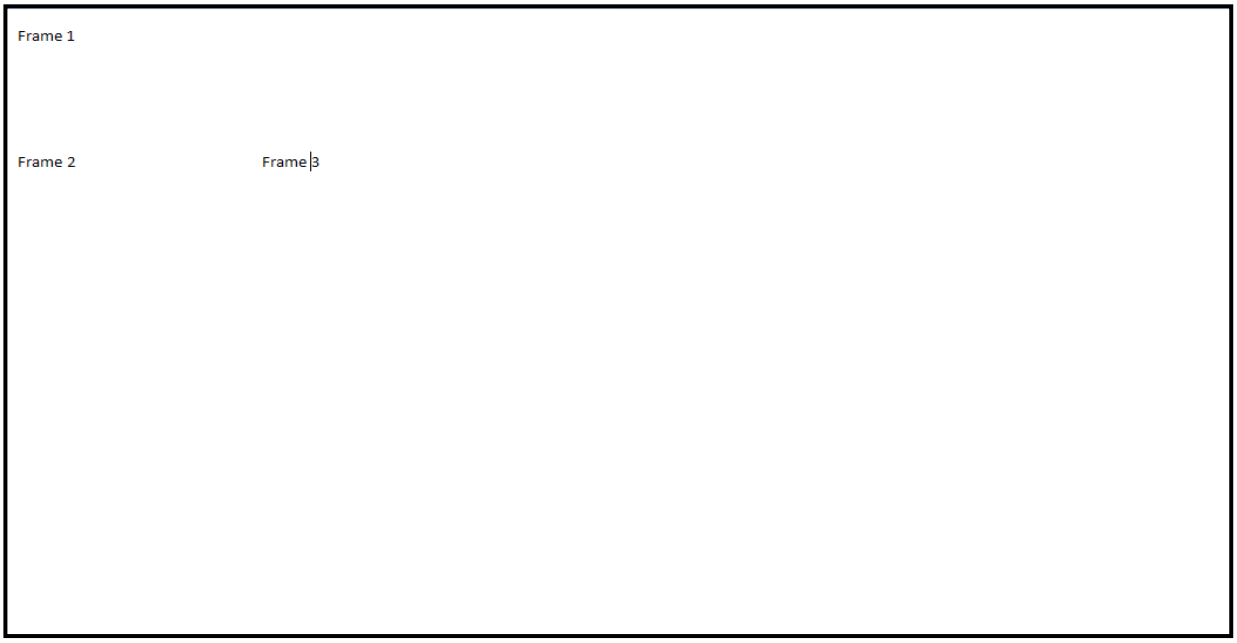
The following properties define the presentation of the splitter bars within this frameset:

```
<w:frameset>
  <w:framesetSplitbar>
    <w:w w:val="200" />
    <w:color w:val="0000FF" />
  </w:framesetSplitbar>
  …
</w:frameset>
```

If the noBorder element is also specified:

```
<w:frameset>
  <w:framesetSplitbar>
    <w:w w:val="200" />
    <w:color w:val="0000FF" />
    <w:noBorder w:val="true" />
  </w:framesetSplitbar>
  …
</w:frameset>
```

Then all frameset splitters are suppressed:



The noBorder element's val attribute has a value of `true`, which specifies that the splitters for this document shall not be displayed. *end example*]

Parent Elements
framesetSplitbar (§2.15.2.20)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

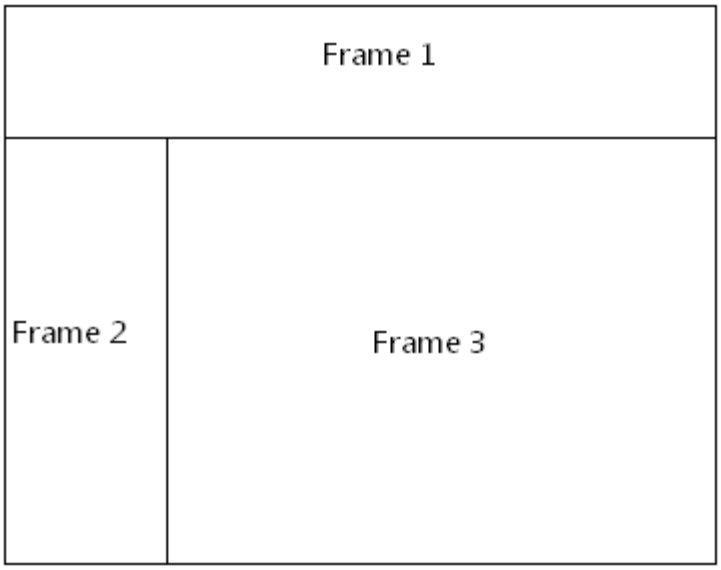
```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.2.31 noResizeAllowed (Frame Cannot Be Resized)

This element specifies whether or not the size of the current frame shall be modifiable (i.e. whether the frame can be resized) when the contents of this document are saved as HTML and displayed in a web browser. When this element is set, the size of the frame shall be set to its current values. This property is analogous to the `noresize` attribute on the `frame` element in HTML.

If this element is omitted, the size of the frame shall be modifiable (the frame may be resized when it is displayed).

[*Example*: Consider a WordprocessingML document which serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  …
  <w:frameset>
    …
    <w:frame>
      <w:name w:val="Frame 2" />
      <w:noResizeAllowed w:val="true" />
    </w:frame>
    …
  </w:frameset>
</w:frameset>
```

The noResizeAllowed element has a val attribute of `true`, which specifies that the size of the frame specified by `Frame 2` shall not be modifiable (the two borders which intersect that frame cannot be resized). *end example*]

Parent Elements
frame (§2.15.2.16)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.2.32 optimizeForBrowser (Disable Features Not Supported by Target Web Browser)

This element specifies whether applications should attempt to detect the target web browser for any web page produced from this document, and subsequently disable all user interface and output which is not supported by that target web browser.

The target web browser can be determined by the state of the following elements:

- allowPNG
- doNotRelyOnCSS
- relyOnVML
- doNotSaveWebPagesAsSingleFile

The following table determines how this determination is made:

Settings	Target Browser

Settings	Target Browser
allowPNG is off doNotRelyOnCSS is on relyOnVML is off doNotSaveWebPagesAsSingleFile is on	Microsoft Internet Explorer 3.0 or later Netscape Navigator 3.0 or later
allowPNG is off doNotRelyOnCSS is off relyOnVML is off doNotSaveWebPagesAsSingleFile is on	Netscape Navigator 4.0 or later
allowPNG is off doNotRelyOnCSS is off relyOnVML is off doNotSaveWebPagesAsSingleFile is off	Microsoft Internet Explorer 4.0 or later
allowPNG is off doNotRelyOnCSS is off relyOnVML is on doNotSaveWebPagesAsSingleFile is off	Microsoft Internet Explorer 5.0 or later
allowPNG is on doNotRelyOnCSS is off relyOnVML is on doNotSaveWebPagesAsSingleFile is off	Microsoft Internet Explorer 6.0 or later

If this element is omitted, then no user interface or output which is not supported by that target web browser shall be disabled.

[*Example*: Consider a document whose web settings part contains the following WordprocessingML:

```
<w:webSettings>
    …
    <w:optimizeForBrowser />
    <w:allowPNG w:val="on"/>
    <w:relyOnVML w:val="on"/>
</w:webSettings>
```

The optimizeForBrowser element specifies that all settings which are not compatible with the target web browser shall be disabled. Since the settings of the four element described above match a target browser of Microsoft Internet Explorer 6.0, features not supported by Microsoft Internet Explorer 6.0 shall be disabled. *end example*]

Parent Elements
webSettings (§2.15.2.44)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.2.33 pixelsPerInch (Pixels per Inch for Graphics/Images)

This element specifies the number of pixels per inch (or density) that will be used for the display of pictures or table cells when a WordprocessingML document is saved as a web page. The size that is specified by this element affects the size of the pictures or table cells relative to the size of text in the document. The pixels per inch (ppi) measurement is relative to the screen resolution, and the resulting physical dimensions of the resulting image or cell in pixels (which are used in web pages, but not for printed documents) are the result of the original dimensions (in inches) multiplied by the number of pixels per inch.

The range of values for this element is typically from 19 to 480 pixels per inch. The common settings for popular screen sizes are 72, 96, and 120 pixels per inch.

If this element is omitted, then a default size of 96 pixels per inch shall be used when determining the number of pixels for images and/or table cells within this document.

[*Note*: This setting is typically only specified if the target screen resolution for the web page is known, as defined by the targetScreenSz element (§2.15.2.41) to set the optimum screen size for the web page. *end note*]

[*Example*: Consider a WordprocessingML document which contains the following content within the web settings part:

```
<w:webSettings>
  <w:pixelsPerInch w:val="200" />
</w:webSettings>
```

The pixelsPerInch element's val attribute has a value of 200, which specifies that all inches to pixels conversions done when saving this web page shall be done assuming a transformation of 200 pixels per inch. *end example*]

Parent Elements
webSettings (§2.15.2.44)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
    <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.15.2.34 relyOnVML (Utilize VML When Saving as Web Page)

This element specifies whether applications may utilize the Vector Markup Language format when saving the content of this WordprocessingML document as a web page, when graphical elements which can leverage this format are present in the document.

If this element is omitted, then a graphic image format should be used either in place of or in concert with the Vector Markup Language output in order to specify the formatting and positioning for objects which are part of the resulting web page.

[*Note*: This setting is intended for applications to save web pages which can be supported by legacy web browsers which do not support Vector Markup Language when attempting to read and display the resulting web page. *end note*]

[*Example*: Consider a WordprocessingML document which contains the following content within the web settings part:

```
<w:webSettings>
  <w:relyOnVML w:val="false" />
</w:webSettings>
```

The relyOnVML element has a val attribute value of `false`, which specifies that applications should utilize a graphical image version of all objects which could utilize Vector Markup Language output. This does not preclude the use of the VML output, but does specify that a graphical element shall be included as well. *end example*]

Parent Elements
webSettings (§2.15.2.44)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.2.35 right (Right Border for HTML div)

This element specifies the border which shall be displayed at the right of the boundaries of the current HTML `div` object.

If this element is omitted, then this HTML `div` object shall not have a right border.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body>
    <div style=" border-left-style:solid; border-right-style:groove; border-
right-width:1px; border-top-style:dashed; border-top-width:3px; border-bottom-
style:outset; border-bottom-width:3px">
      <p>paragraph of text</p>
    </div>
  </body>
</html>
```

This HTML would therefore normally appear as follows (image scaled appropriately):

Now, when this document is saved in the WordprocessingML format, the information stored on the `div` elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1785730240">
    <w:divBdr>
      <w:top w:val="dashed" w:sz="18" w:space="7" w:color="auto" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="auto" />
      <w:bottom w:val="outset" w:sz="18" w:color="auto" />
      <w:right w:val="threeDEngrave" w:sz="6" w:color="auto" />
    </w:divBdr>
  </w:div>
</w:divs>
```

The right element specifies border information about the right border for the single HTML `div` structure in the document; in this case, a 0.75 point bottom border of type `threeDEngrave`. The initial 1 pixel border was converted to 0.75 points using the following logic:

$$1\text{px} * \frac{1 \text{ inch}}{96 \text{ px}} * \frac{576 \text{ eighth points}}{1 \text{ inch}} = 6 \text{ eighth points } (0.75 \text{ points})$$

end example]

Parent Elements
divBdr (§2.15.2.7)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or auto to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value auto, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is true, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML:

Attributes	Description
	<w:bottom w:shadow="true" ... /> This frame's val is true, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object. When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points. When a document has a page border that is relative to the text extents (using a value of text in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points. [*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows: <w:pgBorders w:offsetFrom="page"> <w:bottom … w:space="24/> </w:pgBorders The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*] The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).
sz (Border Width)	Specifies the width of the current border. If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value. If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value. [*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup: <w:top w:val="dashed" w:sz="24" …/> <w:left w:val="dashed" w:sz="24" …/>

Attributes	Description
	<w:bottom w:val="dashed" w:sz="24" …/> <w:right w:val="dashed" w:sz="24" …/> The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*] The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).
themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border. The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document. [*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup: <w:top … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:bottom … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:left … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:right … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*] The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).
themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255$$

Attributes	Description
	$= 102$ $= 66(hex)$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $L' = 0.53 * 0.75$ $= 0.39698$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border.

Attributes	Description
	[*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255 = 153 = 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$L' = L * Tint_{pct} + (1 - Tint_{pct})$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `4F81BD`. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$L' = 0.53 * 0.6 + (1 - .6) = 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$ and converting back to RGB, we get `95B3D7`. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style.

Attributes	Description
	[*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <attribute name="val" type="ST_Border" use="required"/>
   <attribute name="color" type="ST_HexColor" use="optional"/>
   <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
   <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
   <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
   <attribute name="space" type="ST_PointMeasure" use="optional"/>
   <attribute name="shadow" type="ST_OnOff" use="optional"/>
   <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.15.2.36 saveSmartTagsAsXml (Save Smart Tag Data in XML Property Bag)

This element specifies that the information pertaining to all smart tags () in the current document shall be saved into a separate XML-based property bag at the head of the web page when this WordprocessingML document is saved as a web page.

[*Rationale*: This setting is typically used when saving documents known to contain smart tags as web pages, in order to allow the smart tag data within the web page to be processed as a separate XML document by a separate parser, even though the actual HTML content of the resulting web page cannot be parsed by an XML-based parser. *end rationale*]

If this element is omitted, then the smart tag data of this document shall not be saved into a separate XML-compliant property bag within the HTML output when this document is saved as a web page.

[*Example*: Consider a WordprocessingML document which contains the following content:

Stock symbol: MSFT

Date: 7/4/2006

This document might typically write out the following HTML content:

```
<p>Stock symbol: <st1:stockticker>MSFT</st1:stockticker></p>
```

```
<p>Date: <st1:date ls="trans" Month="7" Day="4"
Year="2006"">7/4/2006</st1:date></p>
```

However, if the WordprocessingML document also contains the following content within the web settings part:

```
<w:webSettings>
  <w:saveSmartTagsAsXml w:val="true" />
</w:webSettings>
```

The saveSmartTagsAsXml element specifies that all smart tags in the document shall also be saved into an XML property bag at the header of the file, for example:

```
<head>
  …
  <xml>
    <o:DocumentSmartTags>
      <st1:stockticker>MSFT</st1:stockticker>
      <st1:date ls="trans" Month="7" Day="4" Year="2006">7/4/2006</st1:date>
    </o:DocumentSmartTags>
  </xml>
  …
</head>
```

This header information is in addition to the normal HTML output. *end example*]

Parent Elements
webSettings (§2.15.2.44)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

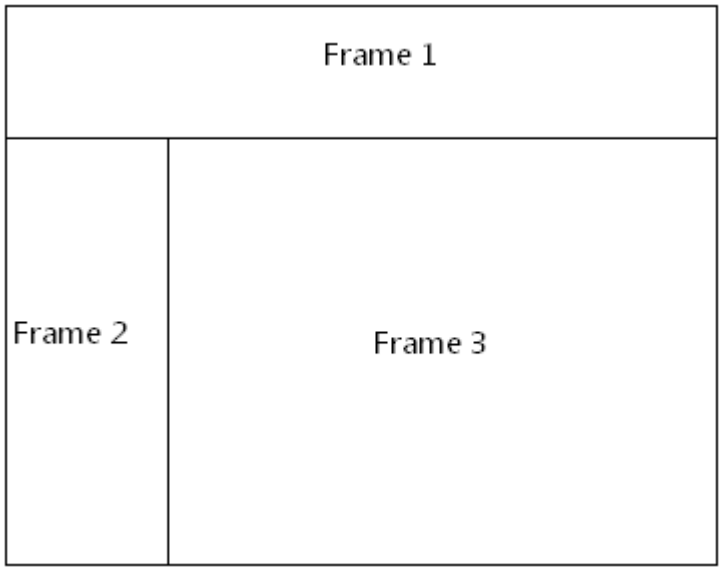
```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.2.37 scrollbar (Scrollbar Display Option)

This element specifies when a scrollbar shall be visible for the contents of the current frame. When this element is set, the val attribute determines exactly when the scrollbar shall be visible. This property is analogous to the `scrolling` attribute on the `frame` element in HTML.

If this element is omitted, the scrollbar shall only be displayed when the contents of the frame exceed the visible space for the frame (i.e. when the scrollbar is needed to display all of the content).

[*Example*: Consider a WordprocessingML document which serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  …
  <w:frameset>
    …
    <w:frame>
      <w:name w:val="Frame 2" />
      <w:scrollbar w:val="auto" />
    </w:frame>
    …
  </w:frameset>
```

```
</w:frameset>
```

The scrollbar element has a val attribute of auto, which specifies that the frame shall only display a scrollbar when it is needed to display all of its content. *end example*]

Parent Elements
frame (§2.15.2.16)

Attributes	Description

val (Scrollbar Display Option Value)	Specifies the criteria under which a scrollbar shall be displayed along with the contents of this frameset, as defined by the simple type referenced below.
	
	[*Example*: Consider a frameset definition within a WordprocessingML document which defines the following scrollbar visibility setting:
	
	`<w:frame>`
	`  <w:scrollbar w:val="on" />`
	`  …`
	`</w:frame>`
	
	The val attribute value of on specifies that the scrollbar shall always be displayed, even when it is not needed (i.e. when it would be displayed disabled). *end example*]
	
	The possible values for this attribute are defined by the ST_FrameScrollbar simple type (§2.18.36).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FrameScrollbar">
   <attribute name="val" type="ST_FrameScrollbar" use="required"/>
</complexType>
```

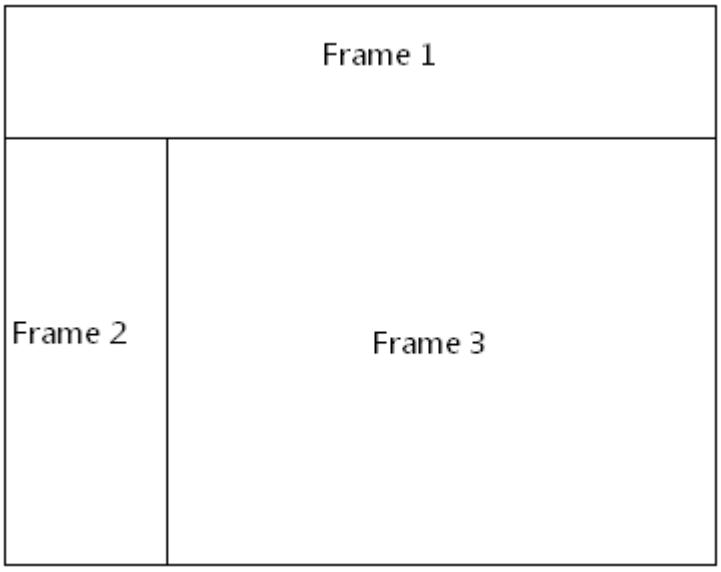
2.15.2.38 sourceFileName (Source File for Frame)

This element specifies the ID for the relationship which specifies the source file for a single frame within a frameset document.

The relationship referenced by this element's id attribute shall exist in the relationship part item for the Web Settings part,or this document shall be considered non-conformant. Also, the type of the relationship referenced by this element's id attribute shall be http://schemas.openxmlformats.org/officeDocument/2006/relationships/frame, or this document shall be considered non-conformant.

If this element is omitted, then no source file is present for the current frame, and one may be created dynamically as needed to display content within the frame.

[*Example*: Consider a WordprocessingML document which serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  …
  <w:frameset>
    …
    <w:frame>
      <w:name w:val="Frame 2" />
      <w:sourceFileName r:id="rId5" />
    </w:frame>
    …
  </w:frameset>
</w:frameset>
```

The sourceFileName element specifies that the contents of this frame shall be the contents of the file targeted by the relationship with ID `rId5` in the web settings part's relationship part item. *end example*]

Parent Elements
frame (§2.15.2.16)

Attributes	Description
id (Relationship to Part)	Specifies the relationship ID to a specified part.

Attributes	Description
Namespace: .../officeDocument/2006/relationships	The specified relationship shall match the type required by the parent element: • http://schemas.openxmlformats.org/officeDocument/2006/relationships/footer for the footerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/header for the headerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://schemas.openxmlformats.org/officeDocument/2006/relationships/printerSettings for the printerSettings element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Rel">
   <attribute ref="r:id" use="required"/>
</complexType>
```

2.15.2.39 sz (Frame Size)

This element specifies the size for a single frame within a frameset.

This size shall be interpreted based on the contents of the frameLayout element (§2.15.2.17) for the parent frameset, as follows:

- If the val attribute on that element is `cols`, then this element specifies the width of the frame
- If the val attribute on that element is `rows`, then this element specifies the height of the frame

Once the axis of this measurement has been established using the criteria above, the actual value of the measurement shall be determined by the following:

- If the val attribute ends in a asterisk (*), then this measurement is a relative measurement (relative to all other frames in this frameset).
- If the val attribute ends in a percentage symbol (%), then this measurement is a percentage of the height and/or width of the parent window, respectively.
- Otherwise, the value of the val attribute specifies the size of the frame in pixels. This measurement shall be interpreted in the context of the pixelsPerInch element (§2.15.2.33) to determine the width of the resulting measurement in inches.

If this element is omitted, then no information shall be implied about the size of the current frame.

[*Example*: Consider a frameset consisting of the following three frames:

The following properties define the presentation of the top frame within this frameset:

```
<w:frameset>
  …
  <w:frame>
    <w:sz w:val="300" />
    <w:name w:val="Frame 1" />
  </w:frame>
  …
  <w:pixelsPerInch w:val="150" />
</w:frameset>
```

The sz element's val attribute specifies that the size of this frame is `300` - which translates to a height of exactly 300 pixels tall. In addition, this document specifies that the intended number of pixels per inch for this measurement is `150`, resulting in a 2" tall frame height. *end example*]

Parent Elements
frame (§2.15.2.16)

Attributes	Description
val (String Value)	Specifies that its contents will contain a string.

Attributes	Description

	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.15.2.40 sz (Nested Frameset Size)

This element specifies the size for a frameset that has been nested within another frameset. If this size appears on a root frameset, then it may be ignored and the main frameset shall encompass the entire window.

This size shall be interpreted based on the contents of the frameLayout element (§2.15.2.17) for the parent frameset (not the current nested frameset), as follows:

- If the val attribute on that element is `cols`, then this element specifies the width of the frameset
- If the val attribute on that element is `rows`, then this element specifies the height of the frameset

Once the axis of this measurement has been established using the criteria above, the actual value of the measurement shall be determined by the following:

- If the val attribute ends in a asterisk (*), then this measurement is a relative measurement (relative to all other frames in this frameset).

- If the val attribute ends in a percentage symbol (%), then this measurement is a percentage of the height and/or width of the parent frameset, respectively.
- Otherwise, the value of the val attribute specifies the size of the frameset in pixels. This measurement shall be interpreted in the context of the pixelsPerInch element (§2.15.2.33) to determine the width of the resulting measurement in inches.

If this element is omitted, then no information shall be implied about the size of the current frameset.

[*Example*: Consider a nested frameset defined as follows:

```
<w:frameset>
  …
  <w:frameset>
    <w:sz w:val="50%" />
    …
  </w:frameset>
  …
  <w:pixelsPerInch w:val="150" />
</w:frameset>
```

The sz element's val attribute specifies that the size of this nested frameset is `50%` - which translates to a width of fifty percent of the width of the parent frameset's height. *end example*]

Parent Elements
frameset (§2.15.2.18); frameset (§2.15.2.19)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`

Attributes	Description
	</w:sdtPr> In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.15.2.41 targetScreenSz (Target Screen Size for Web Page)

This element specifies the ideal minimum target screen size (width by height, specified in pixels) on which web pages generated when saving this document will be displayed. This setting may be used to optimize the output of web pages produced from this document.

If this element is omitted, then the target screen size for web pages produced from this document shall be assumed to be 800x600.

[*Example*: Consider a WordprocessingML document which contains the following content within the web settings part:

```
<w:webSettings>
  <w:targetScreenSz w:val="1600x1200" />
</w:webSettings>
```

The targetScreenSz element's val attribute has a value of 1600x1200, which specifies that a target screen size of 1600 by 1200 pixels shall be assumed when saving this document as a web page. *end example*]

Parent Elements
webSettings (§2.15.2.44)

Attributes	Description
val (Target Screen Size Value)	Specifies the target screen size for web pages produced by this document, as defined by the simple type referenced below. [*Example*: Consider a WordprocessingML document which contains the following content within the web settings part: <w:webSettings> <w:targetScreenSz w:val="1024x768" />

Attributes	Description
	</w:webSettings> The val attribute has a value of 1024x768, which specifies that a target screen size of 1024 by 768 pixels shall be assumed when saving this document as a web page. *end example*] The possible values for this attribute are defined by the ST_TargetScreenSz simple type (§2.18.93).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TargetScreenSz">
   <attribute name="val" type="ST_TargetScreenSz" use="required"/>
</complexType>
```

2.15.2.42 top (Top Border for HTML div)

This element specifies the border which shall be displayed at the top of the boundaries of the current HTML div object.

If this element is omitted, then this HTML div object shall not have a top border.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body>
    <div style=" border-left-style:solid; border-right-style:groove; border-
right-width:1px; border-top-style:dashed; border-top-width:3px; border-bottom-
style:outset; border-bottom-width:3px">
      <p>paragraph of text</p>
    </div>
  </body>
</html>
```

This HTML would therefore normally appear as follows (image scaled appropriately):

Now, when this document is saved in the WordprocessingML format, the information stored on the div elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1785730240">
    <w:divBdr>
      <w:top w:val="dashed" w:sz="18" w:space="7" w:color="auto" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="auto" />
      <w:bottom w:val="outset" w:sz="18" w:color="auto" />
      <w:right w:val="threeDEngrave" w:sz="6" w:color="auto" />
    </w:divBdr>
  </w:div>
</w:divs>
```

The top element specifies border information about the top border for the single HTML `div` structure in the document; in this case, a 2.25 point bottom border of type `dashed`. The initial 3 pixel border was converted to 2.25 points using the following logic:

$$3\text{px} * \frac{1 \text{ inch}}{96 \text{ px}} * \frac{576 \text{ eighth points}}{1 \text{ inch}} = 18 \text{ eighth points } (2.25 \text{ points})$$

end example]

Parent Elements
divBdr (§2.15.2.7)

Attributes	Description
color (Border Color)	Specifies the color for this border. This color may either be presented as a hex value (in RRGGBB format), or `auto` to allow a consumer to automatically determine the border color as appropriate. [*Example*: Consider a border color with value `auto`, as follows: `<w:bottom … w:color="auto"/>` This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§2.18.43).
frame (Create	Specifies whether the specified border should be modified to create a frame effect by

Attributes	Description
Frame Effect)	reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which shall appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" ... />` This frame's val is `true`, indicating that the border frame effect shall be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" ... />` This frame's val is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object. When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§2.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points. When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§2.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points. [*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:

Attributes	Description
	<w:pgBorders w:offsetFrom="page"> <w:bottom … w:space="24/> </w:pgBorders> The offsetFrom attribute specifies that the space value will provide the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset shall be 24 points. *end example*] The possible values for this attribute are defined by the ST_PointMeasure simple type (§2.18.75).
sz (Border Width)	Specifies the width of the current border. If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value. If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range may be reassigned to a more appropriate value. [*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup: <w:top w:val="dashed" w:sz="24" …/> <w:left w:val="dashed" w:sz="24" …/> <w:bottom w:val="dashed" w:sz="24" …/> <w:right w:val="dashed" w:sz="24" …/> The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*] The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§2.18.27).
themeColor (Border Theme Color)	Specifies a theme color to be applied to the current border. The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part,which allows color information to be set centrally in the document. [*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup: <w:top … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />

Attributes	Description
	<w:bottom … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:left … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> <w:right … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" /> The borders have a color with an RGB value of FFA8A0, however, because the themeColor attribute is specified, that value is ignored in favor of the accent2 theme color specified for this document. *end example*] The possible values for this attribute are defined by the ST_ThemeColor simple type (§2.18.104).
themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $$S_{xml} = 0.4 * 255 = 102 = 66(hex)$$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$.

Attributes	Description
	Applying the shade formula with a shade percentage of 75% to the luminance, we get: $$L^{'} = 0.53 * 0.75$$ $$= 0.39698$$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$T_{xml} = 0.6 * 255$$ $$= 153$$ $$= 99(hex)$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}^{'} = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color,

Attributes	Description
	whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $L' = 0.53 * 0.6 + (1 - .6)$ $= 0.71$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§2.18.106).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - only valid for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is single, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§2.18.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
    <attribute name="val" type="ST_Border" use="required"/>
    <attribute name="color" type="ST_HexColor" use="optional"/>
    <attribute name="themeColor" type="ST_ThemeColor" use="optional"/>
    <attribute name="themeTint" type="ST_UcharHexNumber" use="optional"/>
    <attribute name="themeShade" type="ST_UcharHexNumber" use="optional"/>
    <attribute name="sz" type="ST_EighthPointMeasure" use="optional"/>
    <attribute name="space" type="ST_PointMeasure" use="optional"/>
    <attribute name="shadow" type="ST_OnOff" use="optional"/>
    <attribute name="frame" type="ST_OnOff" use="optional"/>
</complexType>
```

2.15.2.43 w (Frameset Splitter Width)

This element specifies the width of the splitters within the frameset in this WordprocessingML document. This element shall only be honored on the root frameset for this document, and may be ignored for all nested framesets in this document.

If this element is omitted, then the default width of the splitters in this document shall be 4.5 points (90 twentieths of a point) wide. If the noBorder element (§2.15.2.30) is also specified, then this element shall be ignored and no splitters shall be displayed.

[*Example*: Consider a frameset consisting of the following three frames:

The following properties define the presentation of the splitter bars within this frameset:

```
<w:frameset>
  <w:framesetSplitbar>
    <w:w w:val="200" />
    <w:color w:val="0000FF" />
  </w:framesetSplitbar>
  …
</w:frameset>
```

The w element's val attribute specifies that the splitter shall be 10 points (200 twentieths of a point) wide when the contents of this document are displayed. *end example*]

Parent Elements
framesetSplitbar (§2.15.2.20)

Attributes	Description
val (Measurement in Twentieths of a Point)	Specifies a positive measurement value, specified in twentieths of a point. This value is interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML element with a val attribute containing a positive measurement in twentieths of a point: `<w:… w:val="720" />` The val attribute has a value of 720, specifying that this measurement value is 720 twentieths of a point (0.5"). This value is interpreted by the parent element as needed. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§2.18.105).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TwipsMeasure">
   <attribute name="val" type="ST_TwipsMeasure" use="required"/>
</complexType>
```

2.15.2.44 webSettings (Web Page Settings)

This element specifies the set of web page settings that have been specified for a single WordprocessingML document. This element is the root element for the Web Settings part within a WordprocessingML document.

[*Example*: Consider the following WordprocessingML fragment for the web page settings in a WordprocessingML document:

```
<w:webSettings>
  <w:frameset>
    …
  </w:frameset>
  <w:doNotUseLongFileNames w:val="true" />
</w:webSettings>
```

The webSettings element contains all of the web page settings for this document. In this case, the web page settings specified for this document are: a frameset defined using the frameset element (§2.15.2.18); and a setting specifying that when this file is saved as a web page, all resulting files shall not exceed 8.3 characters in length using the doNotUseLongFileNames element (§2.15.2.13). *end example*]

Parent Elements
Root element of WordprocessingML Web Settings part

Child Elements	Subclause
allowPNG (Allow PNG as Graphic Format)	§2.15.2.1
divs (Information about HTML div Elements)	§2.15.2.8
doNotOrganizeInFolder (Do Not Place Supporting Files in Subdirectory)	§2.15.2.10
doNotRelyOnCSS (Do Not Rely on CSS for Font Face Formatting)	§2.15.2.11
doNotSaveAsSingleFile (Recommend Web Page Format over Single File Web Page Format)	§2.15.2.12
doNotUseLongFileNames (Do Not Use File Names Longer than 8.3 Characters)	§2.15.2.13
encoding (Output Encoding When Saving as Web Page)	§2.15.2.14
frameset (Root Frameset Definition)	§2.15.2.18
optimizeForBrowser (Disable Features Not Supported by Target Web Browser)	§2.15.2.32
pixelsPerInch (Pixels per Inch for Graphics/Images)	§2.15.2.33
relyOnVML (Utilize VML When Saving as Web Page)	§2.15.2.34
saveSmartTagsAsXml (Save Smart Tag Data in XML Property Bag)	§2.15.2.36
targetScreenSz (Target Screen Size for Web Page)	§2.15.2.41

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_WebSettings">
   <sequence>
      <element name="frameset" type="CT_Frameset" minOccurs="0"/>
      <element name="divs" type="CT_Divs" minOccurs="0"/>
      <element name="encoding" type="CT_String" minOccurs="0"/>
      <element name="optimizeForBrowser" type="CT_OnOff" minOccurs="0"/>
      <element name="relyOnVML" type="CT_OnOff" minOccurs="0"/>
      <element name="allowPNG" type="CT_OnOff" minOccurs="0"/>
      <element name="doNotRelyOnCSS" type="CT_OnOff" minOccurs="0"/>
      <element name="doNotSaveAsSingleFile" type="CT_OnOff" minOccurs="0"/>
      <element name="doNotOrganizeInFolder" type="CT_OnOff" minOccurs="0"/>
      <element name="doNotUseLongFileNames" type="CT_OnOff" minOccurs="0"/>
      <element name="pixelsPerInch" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="targetScreenSz" type="CT_TargetScreenSz" minOccurs="0"/>
      <element name="saveSmartTagsAsXml" type="CT_OnOff" minOccurs="0"/>
   </sequence>
</complexType>
```

2.15.3 Compatibility Settings

The last group of settings stored in WordprocessingML is compatibility settings. *Compatibility Settings* are optional settings used to preserve visual fidelity of documents created in earlier word processing applications. Some of these settings provide affordance for specific behaviors, described in detail below; and others simply instruct applications to mimic the behavior of an existing word processing application.

If compatibility settings are needed, they are stored in the Document Settings part.

It is important to note that all compatibility settings are optional in nature - applications may freely ignore all behaviors described within this section and these settings should not be added unless compatibility is specifically needed in one or more cases. The compatibility settings are provided for backward compatibility with documents created in legacy applications. As such, a number of the settings reference specific applications and specific versions of those applications. This is solely for backward compatibility reasons, and any of those settings are ignorable.

[*Example*: Consider the following WordprocessingML fragment for the compatibility settings in a WordprocessingML document:

```
<w:settings>
  …
  <w:compat>
    <w:noTabHangInd />
  </w:compat>
</w:settings>
```

The compat element contains all of the document settings for this document. In this case, the single setting applied is the suppression of a tab stop when using a hanging indent using the noTabHangInd element (§2.15.3.37). *end example*]

2.15.3.1 adjustLineHeightInTable (Add Document Grid Line Pitch To Lines in Table Cells)

This element specifies whether a document grid defined using the docGrid element (§2.6.5) that specifies a line grid (manually adding additional pitch to each line in the section) shall also be applied to lines within table cells in this section.

Typically, when additional line pitch is added to all lines in a section via the document grid, it is not applied to text in tables. This element, when present with a val attribute value of `true` (or equivalent), specifies that additional line pitch shall be added to lines in table cells.

[*Example*: Consider a WordprocessingML document with a single section, whose document grid is defined such that 25.9 points of additional line pitch are added to each line in the section, as follows:

```
<w:docGrid w:type="lines" w:linePitch="518"/>
```

If text was entered into this section, the default behavior would have line pitch only added to lines which are not in a table cell:

This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid.

This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid.

This section has a character grid. This section has a character grid. This section has a character grid.

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:adjustLineHeightInTable />
</w:compat>
```

Then all lines in this document would have the line pitch from the document grid added to them, resulting in the following output:

This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid.

This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid.

This section has a character grid. This section has a character grid. This section has a character grid.

end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.2 alignTablesRowByRow (Align Table Rows Independently)

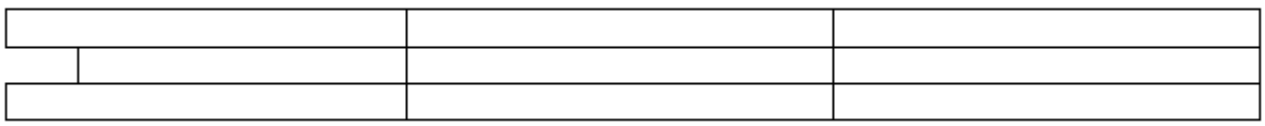
This element specifies whether applications shall align each row within a table independently based on the alignment setting of the jc element (§2.4.22) when displaying the contents of a table in a WordprocessingML document.

When the justification of a table using the jc element is typically applied, that alignment is applied to the contents of the table (the table is centered, left justified, or right-aligned), and then individual rows are laid out based on the resulting table's position. This element, when present with a val attribute value of `true` (or equivalent), specifies that each table row shall be independently aligned based on the table alignment setting, ignoring the placement of all other rows.

[*Example*: Consider a WordprocessingML document with a single centered table, whose second row is defined such that one-half of an inch is left before the row begins, as follows:

```
<w:tbl>
  <w:tblPr>
    <w:jc w:val="center" />
  </w:tblPr>
  <w:tr>
    …
  </w:tr>
  <w:tr>
    <w:trPr>
      <w:gridBefore w:val="1" />
      <w:wBefore w:w="720" w:type="dxa" />
    </w:trPr>
    …
  </w:tr>
  <w:tr>
    …
  </w:tr>
</w:tbl>
```

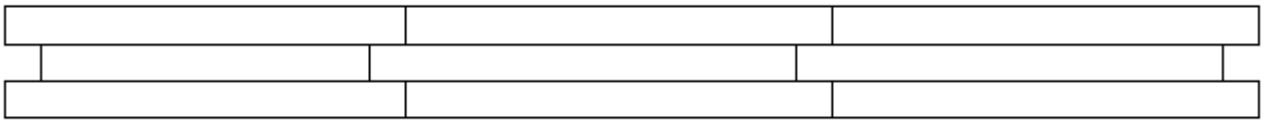
The default presentation would have the entire table centered, then the second row indented beyond that by 720 points:



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:alignTablesRowByRow />
</w:compat>
```

Then that second row would instead be centered on the page independently of the other table rows, resulting in the following output:



In this case, the wBefore element's value is ignored, since the row was centered on the line as a row, and there is no table to be indented relative to. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.3 allowSpaceOfSameStyleInTable (Allow Contextual Spacing of Paragraphs in Tables)

This element specifies whether the suppression of additional space (contextual spacing) defined using the contextualSpacing element (§2.3.1.9) shall be applied to paragraphs contained within tables.

Typically, the rules for the removal of additional paragraph spacing via the contextualSpacing element are applied to all paragraphs in a WordprocessingML document. This element, when present with a val attribute value of `true` (or equivalent), specifies that this setting shall always be ignored for paragraphs in table cells (and additional spacing shall be allowed).

[*Example*: Consider a WordprocessingML document with a default paragraph style with additional spacing after and contextual spacing set, as follows:

```
<w:style w:name="Normal" w:default="1">
  …
  <w:pPr>
    <w:spacing w:after="200" />
    <w:contextualSpacing />
  </w:pPr>
</w:style>
```

The default presentation would have the spacing suppressed between all paragraphs, since they are all of the default paragraph style defined above (contextual spacing applies):

On the Insert tab, the galleries include items that are designed to coordinate with the overall look of your document. You can use these galleries to insert tables, headers, footers, lists, cover pages, and other document building blocks. When you create pictures, charts, or diagrams, they also coordinate with your current document look.
On the Insert tab, the galleries include items that are designed to coordinate with the overall look of your document. You can use these galleries to insert tables, headers, footers, lists, cover pages, and other document building blocks. When you create pictures, charts, or diagrams, they also coordinate with your current document look.

You can easily change the formatting of selected text in the document text by choosing a look for the selected text from the Quick Styles gallery on the Home tab. You can also format text directly by using the other controls on the Home tab. Most controls offer a choice of using the look from the current theme or using a format that you specify directly. On the Insert tab, the galleries include items that are designed to coordinate with the overall look of your document. You can use these galleries to insert tables, headers, footers, lists, cover pages, and other document building blocks. When you create pictures, charts, or diagrams, they also coordinate with your current document look.

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:allowSpaceOfSameStyleInTable />
</w:compat>
```

Then the paragraphs in the table will never have their spacing suppressed, resulting in the following output:

On the Insert tab, the galleries include items that are designed to coordinate with the overall look of your document. You can use these galleries to insert tables, headers, footers, lists, cover pages, and other document building blocks. When you create pictures, charts, or diagrams, they also coordinate with your current document look.
On the Insert tab, the galleries include items that are designed to coordinate with the overall look of your document. You can use these galleries to insert tables, headers, footers, lists, cover pages, and other document building blocks. When you create pictures, charts, or diagrams, they also coordinate with your current document look.

You can easily change the formatting of selected text in the document text by choosing a look for the selected text from the Quick Styles gallery on the Home tab. You can also format text directly by using the other controls on the Home tab. Most controls offer a choice of using the look from the current theme or using a format that you specify directly.

On the Insert tab, the galleries include items that are designed to coordinate with the overall look of your document. You can use these galleries to insert tables, headers, footers, lists, cover pages, and other document building blocks. When you create pictures, charts, or diagrams, they also coordinate with your current document look.

end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.4 applyBreakingRules (Use Legacy Ethiopic and Amharic Line Breaking Rules)

This element specifies whether applications shall use a legacy set of line breaking rules when determining line breaks for text consisting of Ethiopic and/or Amharic characters.

Typically, when line breaking this text, applications should allow line breaks to occur after a character between the UTF-16 (hexadecimal) values 0x1361 and 0x1368 when those characters appear in the document's content. This element, when present with a val attribute value of `true` (or equivalent), specifies that when a line break would occur after a character between the UTF-16 hexadecimal) values 0x1361 and 0x1368, the line break shall occur before all instances of these characters (i.e. no break opportunity shall be afforded after a character in this range).

[*Example*: Consider a WordprocessingML document with a series of Ethopic characters in this range. The default presentation would have any line breaks pushed before or after these characters, ensuring that the characters remain together on a single line.

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:applyBreakingRules />
</w:compat>
```

Then a line break opportunity shall be afforded at any point in a range of these characters, as needed. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>`

Attributes	Description
	The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.5 autofitToFirstFixedWidthCell (Allow Table Columns To Exceed Preferred Widths of Constituent Cells)

This element specifies that when performing an AutoFit on a table in a WordprocessingML document in order to display it, applications shall alter that logic slightly in order to mimic the behavior of a previous word processing application.

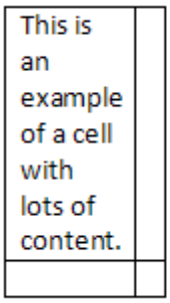
Normally, the AutoFit behavior of a table is as is described in the associated simple type. This element, when present with a val attribute value of true (or equivalent), specifies that this logic shall be changed as follows:

- If the width of a grid column in a table has been set by a preferred table cell width, then that column's width may be enlarged by the content of cells which themselves do not have a preferred width (in contrast, the normal logic never allows the content of cells to override a preferred width on a grid column).

[*Example*: Consider a WordprocessingML table with only one preferred cell width, a width of 720 points on the second cell in the first column, as follows:

```
<w:tbl>
  <w:tr>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
  </w:tr>
  <w:tr>
    <w:tc>
      <w:tcPr>
        <w:tcW w:w="720" w:type="dxa" />
      </w:tcPr>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
  </w:tr>
</w:tbl>
```

The default presentation would have the first column constrained to 720 points by the preferred width of the second cell in the first column:



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:autofitToFirstFixedWidthCell />
</w:compat>
```

Then the column would be resized proportionally based on the content (ignoring the preferred width in that row), resulting in the following output:

This is an example of a cell with lots of content.	

end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.6 autoSpaceLikeWord95 (Emulate Word 95 Full-Width Character Spacing)

This element specifies that applications shall emulate the behavior of a previously existing word processing application (Microsoft Word 95) when determining the spacing between full-width East Asian characters in a document's content.

[*Guidance*: To faithfully replicate this behavior, applications must imitate the behavior of that application, which involves many possible behaviors and cannot be faithfully placed into narrative for this Office Open XML Standard. If applications wish to match this behavior, they must utilize and duplicate the output of those applications. It is recommended that applications not intentionally replicate this behavior as it was deprecated due to issues with its output, and is maintained only for compatibility with existing documents from that application. *end guidance*]

Typically, applications shall not perform this compatibility. This element, when present with a val attribute value of `true` (or equivalent), specifies that applications shall attempt to mimic that existing word processing application in this regard.

[*Example*: Consider a WordprocessingML document with a series of full-width East Asian characters.

If this compatibility setting is turned on:

```
<w:compat>
  <w:autoSpaceLikeWord95 />
</w:compat>
```

Then applications should mimic the behavior of Microsoft Word 95 when determining the space between those characters, as needed. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.7 balanceSingleByteDoubleByteWidth (Balance Single Byte and Double Byte Characters)

This element specifies whether applications shall balance the width of Single Byte Character Set characters and Double Byte Character Set characters when rendering WordprocessingML documents. Specifically, this element

specifies to adjust the fixed pitch fonts' half-width space character and full-width space character to attain a 1 to 2 ratio.

[*Note*: This element is used with East Asian content. Layout and line breaking for East Asian text is dependent on the character width. Half width characters (or Hankaku characters) are one half of an em wide, and full width characters (or Zenkaku characters) are one em wide. Legacy encoding often used a single byte to encode half-width characters and two bytes to encode full width characters. *end note*]

Typically, no adjustment is done on any character when it is displayed as part of a WordprocessingML document. This element, when present with a val attribute value of `true` (or equivalent), specifies that character sizes shall be adjusted as needed to meet the 1:2 ratio described above.

[*Example*: Consider a WordprocessingML document with both SBCS and DBCS characters. The default presentation would have the text displayed as follows:

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:balanceSingleByteDoubleByteWidth />
</w:compat>
```

Then this character-level adjustment must be performed, resulting in the following output:

This adjustment is usually very minute in nature, therefore the result is better illustrated by showing how the characters after the English text were pushed out due to the width balancing of that text:

end example]

Parent Elements

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.8 cachedColBalance (Use Cached Paragraph Information for Column Balancing)

This element specifies whether applications shall incorrectly calculate the height of a paragraph for the purposes of column balancing when rendering WordprocessingML documents. Specifically, this element specifies that when a paragraph's lines have differing heights, an application shall treat this paragraph as though it had only one line equaling the full paragraph height, regardless of the actual number of lines in the paragraph.

[*Guidance*: It is recommended that applications not intentionally replicate this behavior as it was deprecated due to issues with its output, and is maintained only for compatibility with existing documents from a legacy application. *end guidance*]

Typically, lines are correctly measured for their height when balancing columns as part of a WordprocessingML document. This element, when present with a val attribute value of true (or equivalent), specifies that applications shall perform the incorrect calculation in the conditions described above.

[*Example*: Consider a WordprocessingML document with two columns of text which shall be balanced.

If this compatibility setting is turned on:

```
<w:compat>
  <w:cachedColBalance />
</w:compat>
```

Then applications should perform the calculation described above to balance the columns, as needed. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.9 compat (Compatibility Settings)

This element specifies a set of optional compatibility options for the current document.

All settings in this section are optional, but some are very commonly used by different languages, and those which are typically used are as follows:

For Thai, Lao, Khmer, Tibetan, and Armenian:

- The applyBreakingRules setting (§2.15.3.4)

For East Asian languages:

- The adjustLineHeightInTable setting (§2.15.3.1)

- The balanceSingleByteDoubleByteWidth setting (§2.15.3.7)
- The doNotExpandShiftReturn setting (§2.15.3.15)
- The doNotLeaveBackslashAlone setting (§2.15.3.16)
- The spaceForUL setting (§2.15.3.43)
- The ulTrailSpace setting (§2.15.3.55)

[*Example*: Consider a WordprocessingML document with a series of compatibility settings:

```
<w:settings>
  <w:compat>
    …
  </w:compat>
</w:settings>
```

The compat element specifies the set of compatibility settings for a document. *end example*]

Parent Elements
settings (§2.15.1.78)

Child Elements	Subclause
adjustLineHeightInTable (Add Document Grid Line Pitch To Lines in Table Cells)	§2.15.3.1
alignTablesRowByRow (Align Table Rows Independently)	§2.15.3.2
allowSpaceOfSameStyleInTable (Allow Contextual Spacing of Paragraphs in Tables)	§2.15.3.3
applyBreakingRules (Use Legacy Ethiopic and Amharic Line Breaking Rules)	§2.15.3.4
autofitToFirstFixedWidthCell (Allow Table Columns To Exceed Preferred Widths of Constituent Cells)	§2.15.3.5
autoSpaceLikeWord95 (Emulate Word 95 Full-Width Character Spacing)	§2.15.3.6
balanceSingleByteDoubleByteWidth (Balance Single Byte and Double Byte Characters)	§2.15.3.7
cachedColBalance (Use Cached Paragraph Information for Column Balancing)	§2.15.3.8
convMailMergeEsc (Treat Backslash Quotation Delimiter as Two Quotation Marks)	§2.15.3.10
displayHangulFixedWidth (Always Use Fixed Width for Hangul Characters)	§2.15.3.11
doNotAutofitConstrainedTables (Do Not AutoFit Tables To Fit Next To Wrapped Objects)	§2.15.3.12
doNotBreakConstrainedForcedTable (Don't Break Table Rows Around Floating Tables)	§2.15.3.13
doNotBreakWrappedTables (Do Not Allow Floating Tables To Break Across Pages)	§2.15.3.14
doNotExpandShiftReturn (Don't Justify Lines Ending in Soft Line Break)	§2.15.3.15
doNotLeaveBackslashAlone (Convert Backslash To Yen Sign When Entered)	§2.15.3.16
doNotSnapToGridInCell (Do Not Snap to Document Grid in Table Cells with Objects)	§2.15.3.17
doNotSuppressIndentation (Do Not Ignore Floating Objects When Calculating Paragraph Indentation)	§2.15.3.18

Child Elements	Subclause
doNotSuppressParagraphBorders (Do Not Suppress Paragraph Borders Next To Frames)	§2.15.3.19
doNotUseEastAsianBreakRules (Do Not Compress Compressible Characters When Using Document Grid)	§2.15.3.20
doNotUseHTMLParagraphAutoSpacing (Use Fixed Paragraph Spacing for HTML Auto Setting)	§2.15.3.21
doNotUseIndentAsNumberingTabStop (Ignore Hanging Indent When Creating Tab Stop After Numbering)	§2.15.3.22
doNotVertAlignCellWithSp (Don't Vertically Align Cells Containing Floating Objects)	§2.15.3.23
doNotVertAlignInTxbx (Ignore Vertical Alignment in Textboxes)	§2.15.3.24
doNotWrapTextWithPunct (Do Not Allow Hanging Punctuation With Character Grid)	§2.15.3.25
footnoteLayoutLikeWW8 (Emulate Word 6.x/95/97 Footnote Placement)	§2.15.3.26
forgetLastTabAlignment (Ignore Width of Last Tab Stop When Aligning Paragraph If It Is Not Left Aligned)	§2.15.3.27
growAutofit (Allow Tables to AutoFit Into Page Margins)	§2.15.3.28
layoutRawTableWidth (Ignore Space Before Table When Deciding If Table Should Wrap Floating Object)	§2.15.3.29
layoutTableRowsApart (Allow Table Rows to Wrap Inline Objects Independently)	§2.15.3.30
lineWrapLikeWord6 (Emulate Word 6.0 Line Wrapping for East Asian Text)	§2.15.3.31
mwSmallCaps (Emulate Word 5.x for the Macintosh Small Caps Formatting)	§2.15.3.32
noColumnBalance (Do Not Balance Text Columns within a Section)	§2.15.3.33
noExtraLineSpacing (Do Not Center Content on Lines With Exact Line Height)	§2.15.3.34
noLeading (Do Not Add Leading Between Lines of Text)	§2.15.3.35
noSpaceRaiseLower (Do Not Increase Line Height for Raised/Lowered Text)	§2.15.3.36
noTabHangInd (Do Not Create Custom Tab Stop for Hanging Indent)	§2.15.3.37
printBodyTextBeforeHeader (Print Body Text before Header/Footer Contents)	§2.15.3.38
printColBlack (Print Colors as Black And White without Dithering)	§2.15.3.39
selectFldWithFirstOrLastChar (Select Field When First or Last Character Is Selected)	§2.15.3.40
shapeLayoutLikeWW8 (Emulate Word 97 Text Wrapping Around Floating Objects)	§2.15.3.41
showBreaksInFrames (Display Page/Column Breaks Present in Frames)	§2.15.3.42
spaceForUL (Add Additional Space Below Baseline For Underlined East Asian Text)	§2.15.3.43
spacingInWholePoints (Only Expand/Condense Text By Whole Points)	§2.15.3.44
splitPgBreakAndParaMark (Always Move Paragraph Mark to Page after a Page Break)	§2.15.3.45
subFontBySize (Increase Priority Of Font Size During Font Substitution)	§2.15.3.46
suppressBottomSpacing (Ignore Exact Line Height for Last Line on Page)	§2.15.3.47
suppressSpacingAtTopOfPage (Ignore Minimum Line Height for First Line on Page)	§2.15.3.48
suppressSpBfAfterPgBrk (Do Not Use Space Before On First Line After a Page Break)	§2.15.3.49

Child Elements	Subclause
suppressTopSpacing (Ignore Minimum and Exact Line Height for First Line on Page)	§2.15.3.50
suppressTopSpacingWP (Emulate WordPerfect 5.x Line Spacing)	§2.15.3.51
swapBordersFacingPages (Swap Paragraph Borders on Odd Numbered Pages)	§2.15.3.52
truncateFontHeightsLikeWP6 (Emulate WordPerfect 6.x Font Height Calculation)	§2.15.3.53
ulTrailSpace (Underline All Trailing Spaces)	§2.15.3.55
underlineTabInNumList (Underline Following Character Following Numbering)	§2.15.3.56
useAltKinsokuLineBreakRules (Use Alternate Set of East Asian Line Breaking Rules)	§2.15.3.57
useAnsiKerningPairs (Use ANSI Kerning Pairs from Fonts)	§2.15.3.58
useFELayout (Do Not Bypass East Asian/Complex Script Layout Code)	§2.15.3.59
useNormalStyleForList (Do Not Automatically Apply List Paragraph Style To Bulleted/Numbered Text)	§2.15.3.60
usePrinterMetrics (Use Printer Metrics To Display Documents)	§2.15.3.61
useSingleBorderforContiguousCells (Use Simplified Rules For Table Border Conflicts)	§2.15.3.62
useWord2002TableStyleRules (Emulate Word 2002 Table Style Rules)	§2.15.3.63
useWord97LineBreakRules (Emulate Word 97 East Asian Line Breaking)	§2.15.3.64
wpJustification (Emulate WordPerfect 6.x Paragraph Justification)	§2.15.3.65
wpSpaceWidth (Space width)	§2.15.3.66
wrapTrailSpaces (Line Wrap Trailing Spaces)	§2.15.3.67

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Compat">
   <sequence>
      <element name="useSingleBorderforContiguousCells" type="CT_OnOff" minOccurs="0"/>
      <element name="wpJustification" type="CT_OnOff" minOccurs="0"/>
      <element name="noTabHangInd" type="CT_OnOff" minOccurs="0"/>
      <element name="noLeading" type="CT_OnOff" minOccurs="0"/>
      <element name="spaceForUL" type="CT_OnOff" minOccurs="0"/>
      <element name="noColumnBalance" type="CT_OnOff" minOccurs="0"/>
      <element name="balanceSingleByteDoubleByteWidth" type="CT_OnOff" minOccurs="0"/>
      <element name="noExtraLineSpacing" type="CT_OnOff" minOccurs="0"/>
      <element name="doNotLeaveBackslashAlone" type="CT_OnOff" minOccurs="0"/>
      <element name="ulTrailSpace" type="CT_OnOff" minOccurs="0"/>
      <element name="doNotExpandShiftReturn" type="CT_OnOff" minOccurs="0"/>
      <element name="spacingInWholePoints" type="CT_OnOff" minOccurs="0"/>
      <element name="lineWrapLikeWord6" type="CT_OnOff" minOccurs="0"/>
      <element name="printBodyTextBeforeHeader" type="CT_OnOff" minOccurs="0"/>
      <element name="printColBlack" type="CT_OnOff" minOccurs="0"/>
      <element name="wpSpaceWidth" type="CT_OnOff" minOccurs="0"/>
      <element name="showBreaksInFrames" type="CT_OnOff" minOccurs="0"/>
      <element name="subFontBySize" type="CT_OnOff" minOccurs="0"/>
      <element name="suppressBottomSpacing" type="CT_OnOff" minOccurs="0"/>
      <element name="suppressTopSpacing" type="CT_OnOff" minOccurs="0"/>
      <element name="suppressSpacingAtTopOfPage" type="CT_OnOff" minOccurs="0"/>
      <element name="suppressTopSpacingWP" type="CT_OnOff" minOccurs="0"/>
      <element name="suppressSpBfAfterPgBrk" type="CT_OnOff" minOccurs="0"/>
      <element name="swapBordersFacingPages" type="CT_OnOff" minOccurs="0"/>
      <element name="convMailMergeEsc" type="CT_OnOff" minOccurs="0"/>
      <element name="truncateFontHeightsLikeWP6" type="CT_OnOff" minOccurs="0"/>
      <element name="mwSmallCaps" type="CT_OnOff" minOccurs="0"/>
      <element name="usePrinterMetrics" type="CT_OnOff" minOccurs="0"/>
      <element name="doNotSuppressParagraphBorders" type="CT_OnOff" minOccurs="0"/>
      <element name="wrapTrailSpaces" type="CT_OnOff" minOccurs="0"/>
      <element name="footnoteLayoutLikeWW8" type="CT_OnOff" minOccurs="0"/>
      <element name="shapeLayoutLikeWW8" type="CT_OnOff" minOccurs="0"/>
      <element name="alignTablesRowByRow" type="CT_OnOff" minOccurs="0"/>
      <element name="forgetLastTabAlignment" type="CT_OnOff" minOccurs="0"/>
      <element name="adjustLineHeightInTable" type="CT_OnOff" minOccurs="0"/>
      <element name="autoSpaceLikeWord95" type="CT_OnOff" minOccurs="0"/>
      <element name="noSpaceRaiseLower" type="CT_OnOff" minOccurs="0"/>
      <element name="doNotUseHTMLParagraphAutoSpacing" type="CT_OnOff" minOccurs="0"/>
      <element name="layoutRawTableWidth" type="CT_OnOff" minOccurs="0"/>
      <element name="layoutTableRowsApart" type="CT_OnOff" minOccurs="0"/>
      <element name="useWord97LineBreakRules" type="CT_OnOff" minOccurs="0"/>
      <element name="doNotBreakWrappedTables" type="CT_OnOff" minOccurs="0"/>
      <element name="doNotSnapToGridInCell" type="CT_OnOff" minOccurs="0"/>
      <element name="selectFldWithFirstOrLastChar" type="CT_OnOff" minOccurs="0"/>
      <element name="applyBreakingRules" type="CT_OnOff" minOccurs="0"/>
      <element name="doNotWrapTextWithPunct" type="CT_OnOff" minOccurs="0"/>
      <element name="doNotUseEastAsianBreakRules" type="CT_OnOff" minOccurs="0"/>
      <element name="useWord2002TableStyleRules" type="CT_OnOff" minOccurs="0"/>
      <element name="growAutofit" type="CT_OnOff" minOccurs="0"/>
      <element name="useFELayout" type="CT_OnOff" minOccurs="0"/>
```

```
      <element name="useNormalStyleForList" type="CT_OnOff" minOccurs="0"/>
      <element name="doNotUseIndentAsNumberingTabStop" type="CT_OnOff" minOccurs="0"/>
      <element name="useAltKinsokuLineBreakRules" type="CT_OnOff" minOccurs="0"/>
      <element name="allowSpaceOfSameStyleInTable" type="CT_OnOff" minOccurs="0"/>
      <element name="doNotSuppressIndentation" type="CT_OnOff" minOccurs="0"/>
      <element name="doNotAutofitConstrainedTables" type="CT_OnOff" minOccurs="0"/>
      <element name="autofitToFirstFixedWidthCell" type="CT_OnOff" minOccurs="0"/>
      <element name="underlineTabInNumList" type="CT_OnOff" minOccurs="0"/>
      <element name="displayHangulFixedWidth" type="CT_OnOff" minOccurs="0"/>
      <element name="splitPgBreakAndParaMark" type="CT_OnOff" minOccurs="0"/>
      <element name="doNotVertAlignCellWithSp" type="CT_OnOff" minOccurs="0"/>
      <element name="doNotBreakConstrainedForcedTable" type="CT_OnOff" minOccurs="0"/>
      <element name="doNotVertAlignInTxbx" type="CT_OnOff" minOccurs="0"/>
      <element name="useAnsiKerningPairs" type="CT_OnOff" minOccurs="0"/>
      <element name="cachedColBalance" type="CT_OnOff" minOccurs="0"/>
   </sequence>
</complexType>
```

2.15.3.10 convMailMergeEsc (Treat Backslash Quotation Delimiter as Two Quotation Marks)

This element specifies whether applications should perform a conversion of the contents of a mail merge data source when reading those contents in order to perform a mail merge operation with their contents.

Typically, the contents of a mail merge data source are read in exactly as specified when performing a mail merge with the contents of a data source. This element, when present with a val attribute value of `true` (or equivalent), specifies that applications shall interpret delimiters composed of a backslash and quotation mark (\") as two quotation marks (""), within external data sources to be connected to via a mail merge.

[*Example*: Consider a WordprocessingML document with the following content in its data source:

```
This is a \"test\".
```

The default presentation would have the resulting merged data read in just as it appears:

```
This is a \"test\".
```

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:convMailMergeEsc />
</w:compat>
```

Then instances of a backslash and quotation mark would be converted, resulting in the following output:

```
This is a ""test"".
```

end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

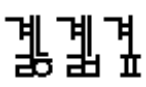
```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.11 displayHangulFixedWidth (Always Use Fixed Width for Hangul Characters)

This element specifies whether applications should assume that all characters in the Hangul Syllables Unicode sub range (character values between 0xAC00 and 0xD7FF) are of a single fixed width or shall use the characters widths defined by the font in use (typical for a proportional width font).

Typically, applications shall retrieve the character width for any character in a document from the associated font, allowing each character to be of its own width (a proportional width character). This element, when present with a val attribute value of true (or equivalent), specifies that applications shall instead assume a single fixed width for all characters in the Hangul Syllables sub range, by reading the width of Unicode character 0x4E00 from the associated font and using that width for all Hangul characters (or, if that character is not present, the next available character in the font).

[*Example*: Consider a WordprocessingML document with three Hangul characters:



The default presentation would have each of those characters using the widths defined by the font (the highlighting indicates that each character has its own width):

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:displayHangulFixedWidth />
</w:compat>
```

Then all three characters are forced to the fixed width of character 0x4E00 from the font (or, in this case, the next available character), resulting in the characters in the font being forced to that fixed width, which results in the following output:

Notice from the highlighting that the characters have been compressed to the width of the single character and displayed at that fixed width. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off.

Attributes	Description
	[*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

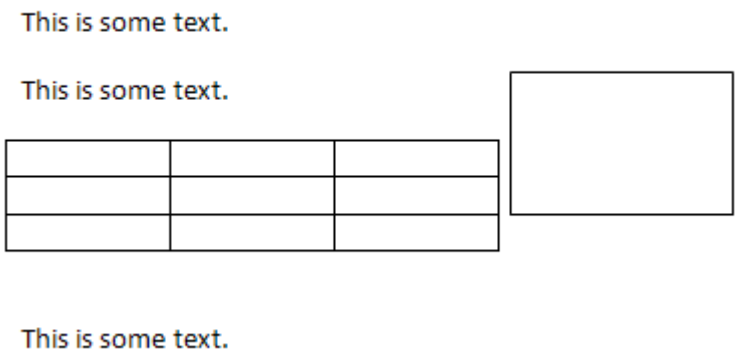
```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.12 doNotAutofitConstrainedTables (Do Not AutoFit Tables To Fit Next To Wrapped Objects)

This element specifies whether applications shall allow tables to be resized to the remaining available line width when they are using the AutoFit algorithm and part of that line is filled by a shape with a wrapping type with a value of `square` or `tight`.

Typically, a table which is AutoFit and has a preferred width shall have its width reduced in order to allow a floating shape to wrap around its contents within the document, as that shape simply reduces the width of the line and the AutoFit algorithm applies to the remaining line width. This element, when present with a val attribute value of `true` (or equivalent), specifies that tables shall never have any preferred width overridden to allow them to wrap around that floating object, and shall instead be pushed to the next full width line in the document to be displayed.

[*Example*: Consider a WordprocessingML document with a floating shape centered in the document, followed by a table with preferred cell widths of 2.22", as follows:

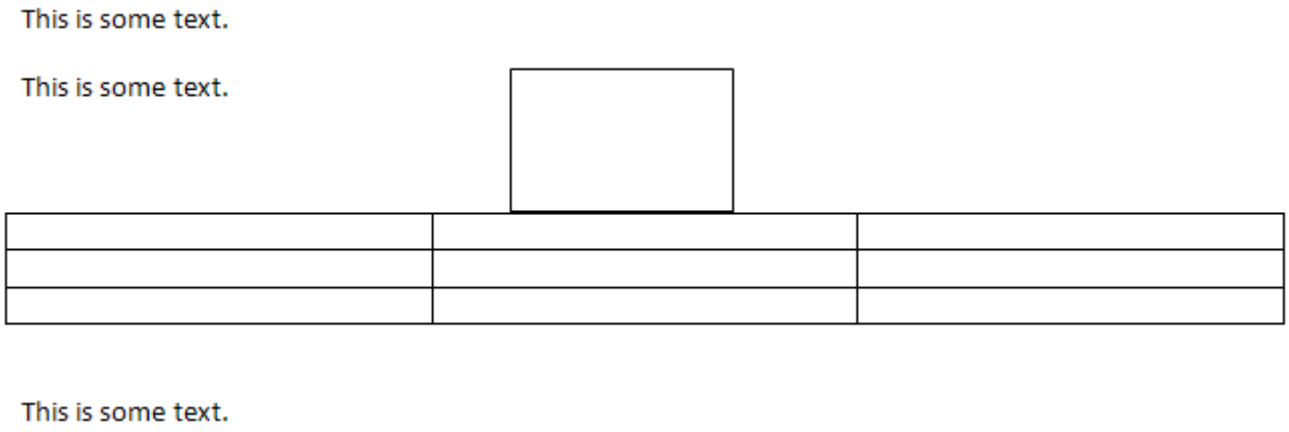


The default presentation of this document overrides the preferred cell widths to force the table to fit on the line next to the floating shape with tight wrapping.

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:dontAutofitConstrainedTables />
</w:compat>
```

Then that table is not resized, so it cannot fit and must be pushed to the next full width line, resulting in the following output:



end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

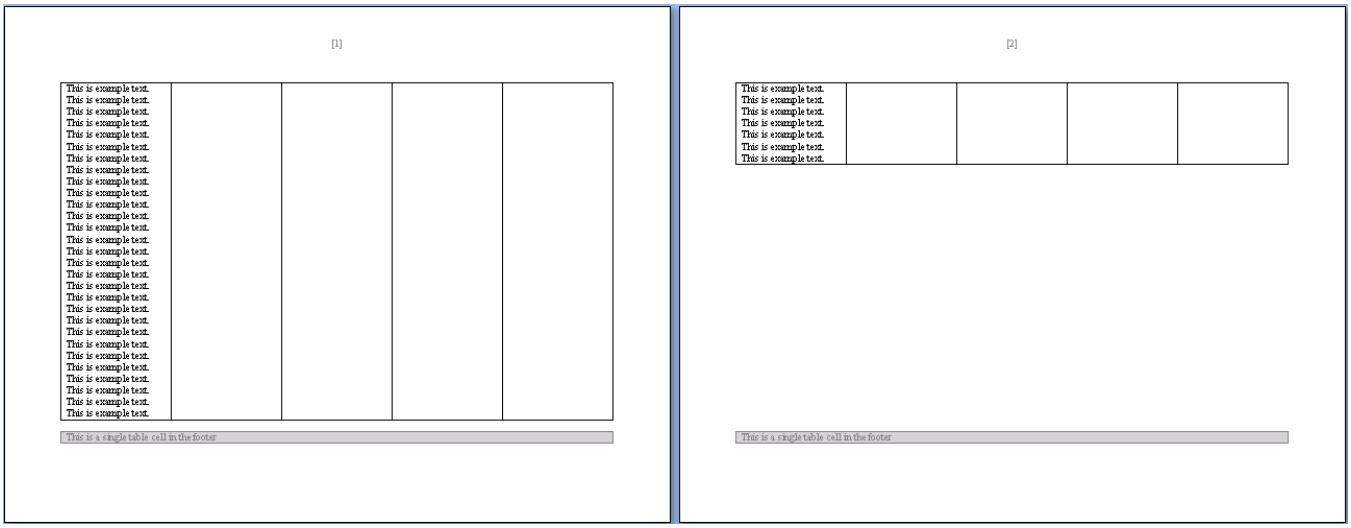
2.15.3.13 doNotBreakConstrainedForcedTable (Don't Break Table Rows Around Floating Tables)

This element specifies whether applications shall allow a table row to be split in two when its contents are displayed under the following circumstances:

- The table row exceeds one page in height (it must be split into two pages)
- The table row would need to be split in order to accommodate a floating table also on the page (tables which have been set to floating using the tblpPr element (§2.4.54))

Typically, assuming the cantSplit property (§2.4.6) is not set, a table row which cannot fit on one single page shall be split as needed around any floating table on a page, in order to allow its contents to be fully displayed across two or more pages. This element, when present with a val attribute value of `true` (or equivalent), specifies that table rows which exceed one page in height shall never be split around floating tables in the document, and shall instead be displayed on the first page below the floating table, even if that means that part of the table row is clipped by the edge of the page.

[*Example*: Consider a WordprocessingML document with a long single table row which must be split across two separate pages in the document, in order to accommodate a floating table anchored in the footer, as follows:

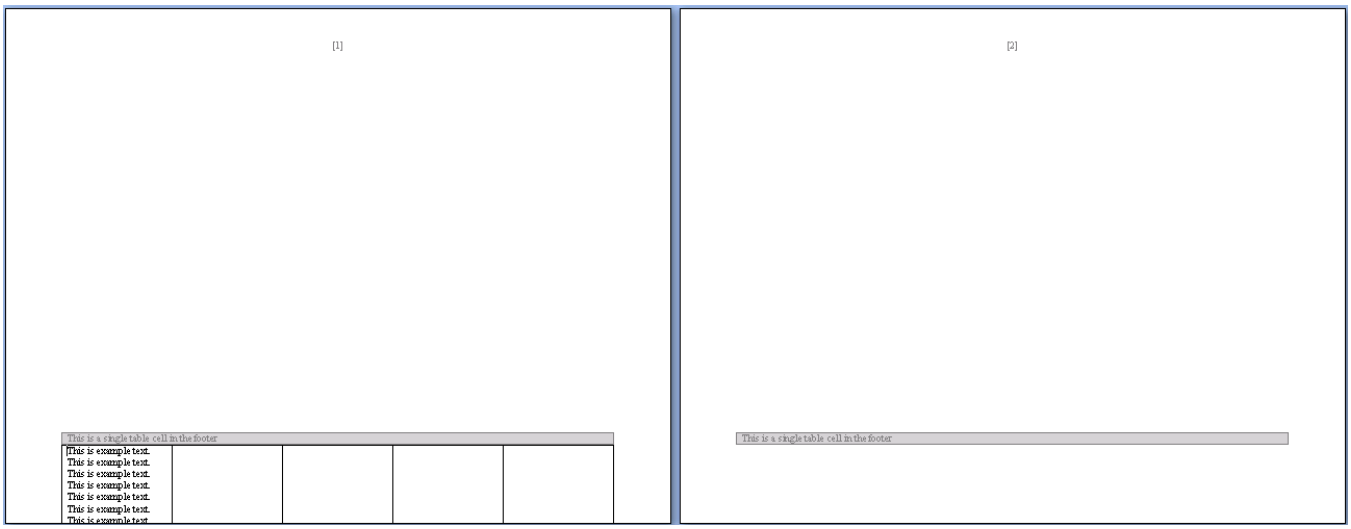


The default presentation of this document forces that row to be split as needed around that floating table.

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:dontBreakConstrainedForcedTable />
</w:compat>
```

Then that table row is never split around the floating table, so it is always placed below that floating table on the page, and allowed to flow off the page as needed, resulting in the following output:



This example, while extreme, shows how the row is placed below the floating table, rather than breaking around it. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

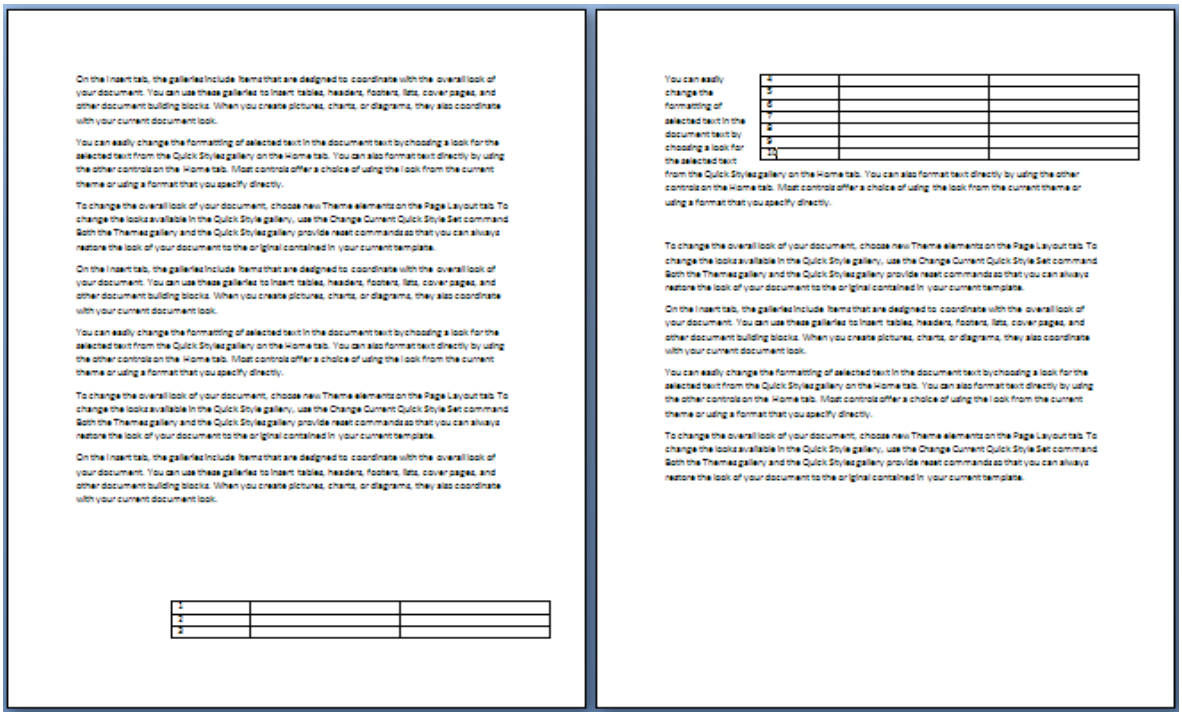
```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.14 doNotBreakWrappedTables (Do Not Allow Floating Tables To Break Across Pages)

This element specifies whether applications shall allow tables which have been set to floating using the tblpPr element (§2.4.54) shall be allowed to break across multiple pages when needed.

Typically, a table whose contents cannot all be displayed on one page is broken as needed across multiple pages in order to preserve the location of the table (just as a paragraph of multiple lines is broken across pages as needed). This element, when present with a val attribute value of `true` (or equivalent), specifies that floating tables shall never be broken across pages, and shall instead be put on the first page by adjusting the starting position of the table as needed to fit on that single page.

[*Example*: Consider a WordprocessingML document with a floating table positioned at the bottom of a page , as follows:

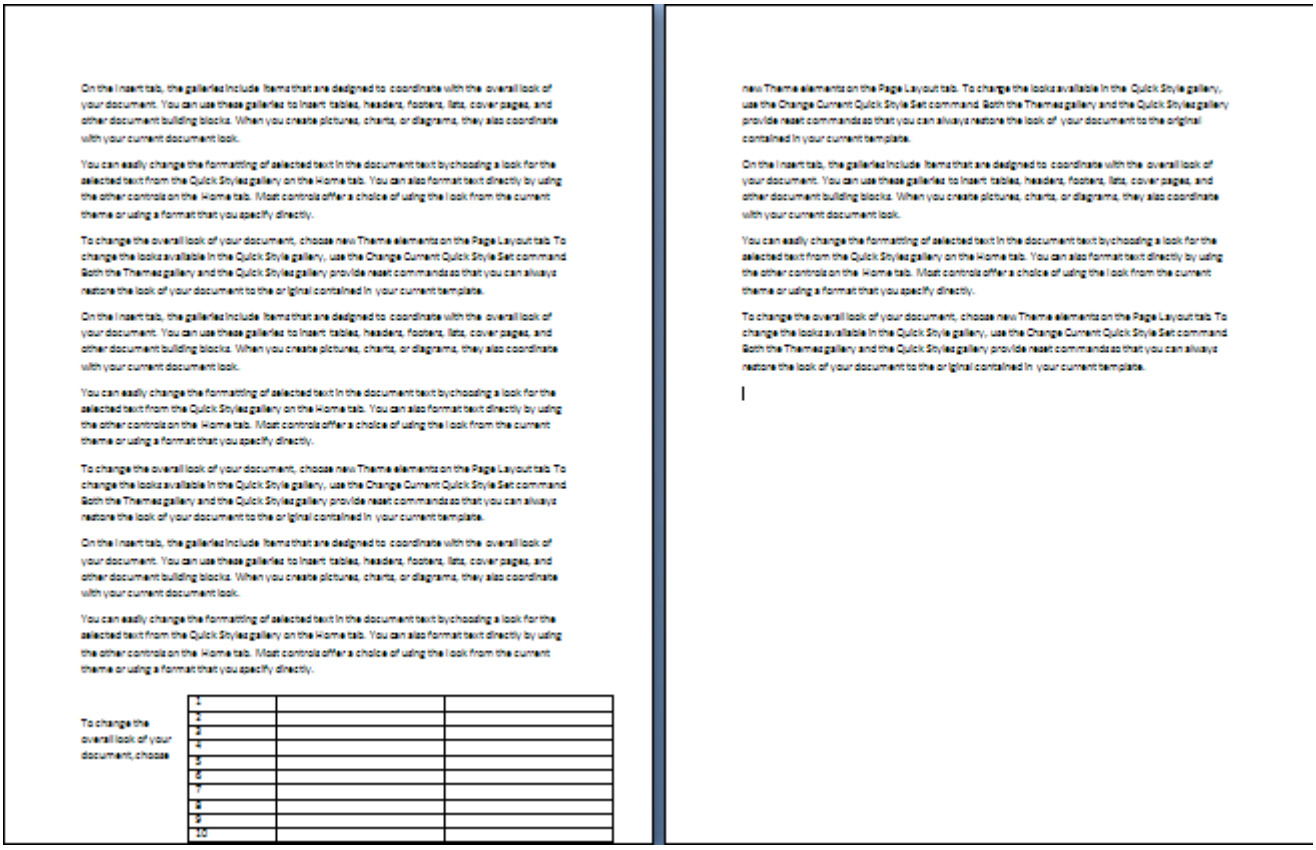


The default presentation of this document results in that table being broken across two pages of content.

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:dontBreakWrappedTables />
</w:compat>
```

Then that table is not broken across the page boundary, so it must be moved further up on the first page to accommodate its entire size, resulting in the following output:



Notice that the table now flows into the page margins in order to keep it on one page. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.15 doNotExpandShiftReturn (Don't Justify Lines Ending in Soft Line Break)

This element specifies whether applications should fully justify the contents of incomplete lines which end in a soft line break when the parent paragraph is fully justified using the jc element (§2.3.1.13).

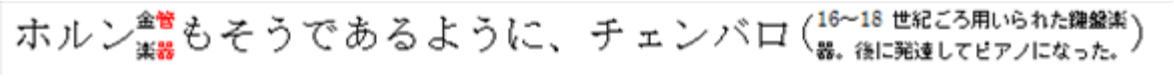
Typically, applications shall fully justify all lines in a paragraph when that setting is specified using the jc element except for the last line in the paragraph (the line ending with the paragraph mark). This element, when present with a val attribute value of `true` (or equivalent), specifies that any line which ends in a soft line break shall also not be fully justified when the paragraph specifies that setting.

[*Example*: Consider a WordprocessingML document with a paragraph whose first single line consists of East Asian characters followed by a soft paragraph mark. The default presentation would have the contents of that line fully justified:

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:doNotExpandShiftReturn />
</w:compat>
```

Then this line is not fully justified, as it ends with a soft line break, resulting in the following output:



end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off.

Attributes	Description
	[*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.16 doNotLeaveBackslashAlone (Convert Backslash To Yen Sign When Entered)

This element specifies whether applications should automatically convert the backslash character into the yen character when it is added through user keyboard input.

Typically, no automatic conversion of one character to another is done when characters are entered by the user. This element, when present with a val attribute value of true (or equivalent), specifies that all entries of the backslash (\) character shall automatically be converted to a yen symbol (¥) when the former is entered.

[*Example*: Consider a WordprocessingML document where the user types the following:

```
Hello \ world.
```

The default presentation would have exactly that:

```
Hello \ world.
```

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:doNotLeaveBackslashAlone />
</w:compat>
```

Then the backslash would be converted, resulting in the following output:

```
Hello ¥ world.
```

end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.17 doNotSnapToGridInCell (Do Not Snap to Document Grid in Table Cells with Objects)

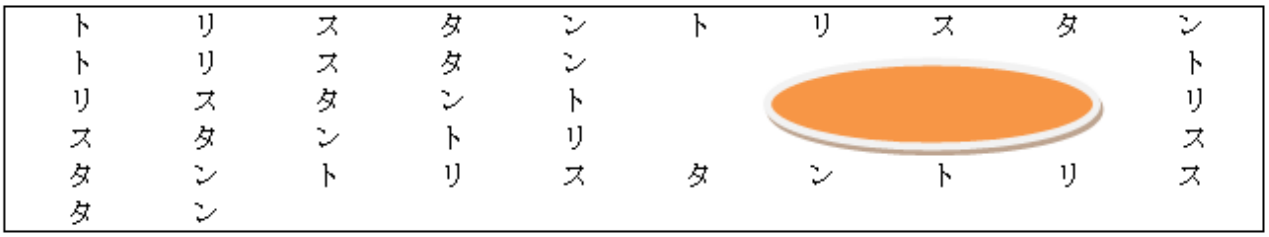
This element specifies whether a document grid defined using the docGrid element (§2.6.5) shall be applied to the contents of table cells in that section which also contain floating objects defined using the Vector Markup Language syntax. Note that the floating object must be part of the cell, and simply not displayed over the cell due to its anchoring relative to another part of the document.

Typically, if a floating object is present in a table cell, then that setting shall have no impact on whether East Asian text in that cell is snapped to the document grid (as text is always snapped to the grid). This element, when present with a val attribute value of true (or equivalent), specifies that whenever a floating object defined using VML is present in a table cell, that the cell's contents shall not be snapped to the document grid.

[*Example*: Consider a WordprocessingML document consisting of a single section, whose document grid settings specify that each page shall be exactly 10 characters wide, as follows:

```
<w:sectPr>
  <w:docGrid w:type="snapToChars" w:charSpace="146636" />
</w:sectPr>
```

If this document contains a table with a single cell, containing some text and a single shape defined using the Vector Markup Language syntax, the contents of the cell are still snapped to the 10 characters per line character grid, as follows:



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:doNotSnapToGridInCell />
</w:compat>
```

Then the presence of a floating object in each cell shall result in the document grid setting being ignored, resulting in the following output:

The additional character pitch was still added to each character on the line, but those characters are no longer snapped to the document grid. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>`

Attributes	Description
	The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.18 doNotSuppressIndentation (Do Not Ignore Floating Objects When Calculating Paragraph Indentation)

This element specifies whether applications should ignore the presence of floating objects when calculating the starting position of paragraphs which are wrapped around floating objects defined using the Vector Markup Language (VML) syntax.

Typically, the presence of a floating object on the same line or lines as a paragraph shall only affect the text when the floating object occurs where that text would normally be presented. [*Example*: Text at a 1" indentation would only be displaced by a floating object that appears at that position and not one that appears from 0" to 0.5" on the same line. *end example*].

This element, when present with a val attribute value of true (or equivalent), specifies that floating objects shall always impact paragraphs on the same line in two ways:

- If the paragraph is not numbered, then it shall tightly wrap any floating object which precedes it on the same line, ignoring its own indentation settings. [*Example*: A paragraph with a 1" left indent shall tightly wrap a floating object which appears at only 0.25" on the same line. *end example*]
- If the paragraph is numbered using the numPr element (§2.3.1.19), then it shall calculate and use its full indent relative to the edge of the floating object, not relative to the edge of the page. [*Example*: A numbered paragraph with a 1" left indent shall appear 1.5" into the page if it is preceded by a floating object which appears at 0.5" on the same line. *end example*]

[*Example*: Consider a WordprocessingML document with a narrow floating object at 0.5" on the page, surrounded by both numbered and unnumbered paragraphs.

The default presentation would have no impact on the paragraphs based on that floating object, since two two do not intersect:

One

Two

Three

Four

Five

1. One
2. Two
3. Three
4. Four
5. Five

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:doNotSuppressIndentation />
</w:compat>
```

Then the two alternate rules defined above would apply, resulting in the following output:

One

Two

Three

Four

Five

1. One
    2. Two
    3. Three
    4. Four
5. Five

end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

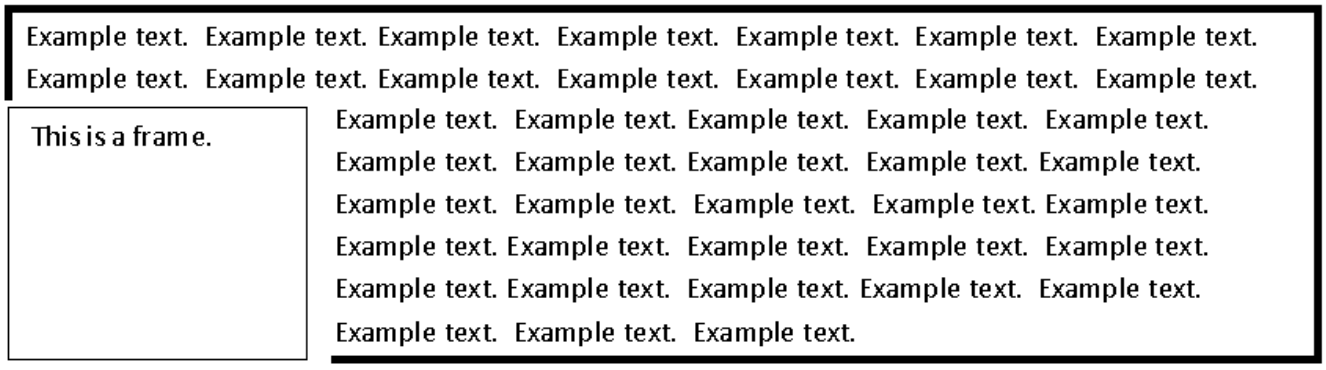
2.15.3.19 doNotSuppressParagraphBorders (Do Not Suppress Paragraph Borders Next To Frames)

This element specifies whether applications should suppress paragraph borders defined using the pBdr element (§2.3.1.24) when those borders would be displayed next to the contents of paragraphs which have been defined as frames using the framePr element (§2.3.1.11).

Typically, when a paragraph's borders appear next to a frame, those borders are suppressed to avoid having two borders in close proximity. This element, when present with a val attribute value of `true` (or equivalent), specifies that those borders shall not be suppressed.

[*Example*: Consider a WordprocessingML document with a paragraph with a paragraph border that is bounded on its bottom let side by a text frame.

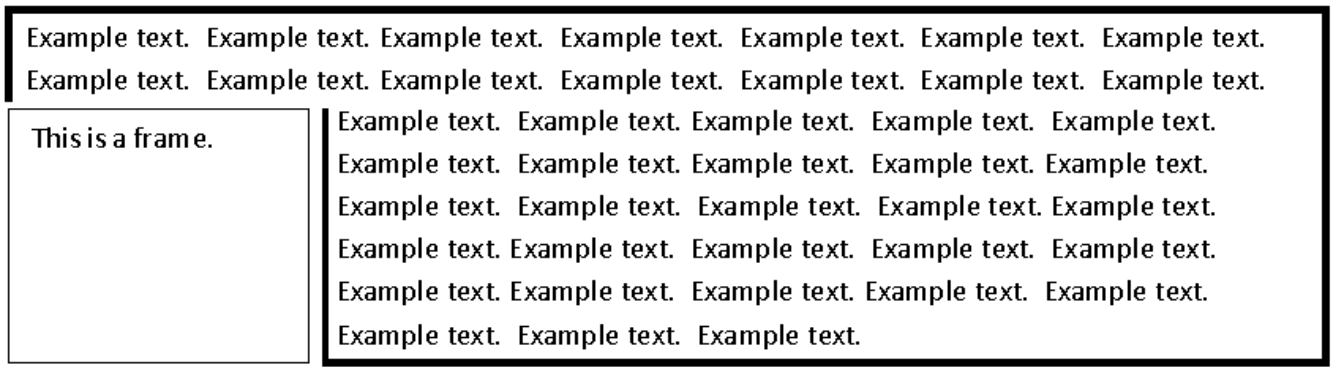
The default presentation would suppress the borders which intersect the frame (in this case, the right border of lines three through eight):



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:doNotSuppressParagraphBorders />
</w:compat>
```

Then no border suppression shall take place, resulting in the following output:



end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.20 doNotUseEastAsianBreakRules (Do Not Compress Compressible Characters When Using Document Grid)

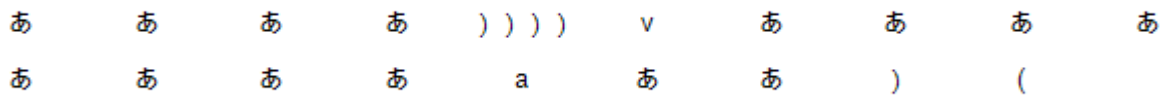
This element specifies whether applications should compress characters with identical compression rules when the document grid has been defined using the docGrid element (§2.6.5). *Compression rules* refer to the additional bearing on the left and/or right side of a typical character, which can be compressed as needed without modifying the actual width of the character (its breadth).

Typically, punctuation characters with an identical set of compression rules are compressed when the contents of a document are displayed. This element, when present with a val attribute value of true (or equivalent), specifies that if a document grid is defined for the current section, compression shall never be performed on any character - all compressible characters shall be individually snapped to the document grid.

[*Example*: Consider a WordprocessingML document with a document grid set to allow 10 characters per line:

```
<w:sectPr>
  <w:docGrid w:type="snapToChars" w:charSpace="146636" … />
</w:sectPr>
```

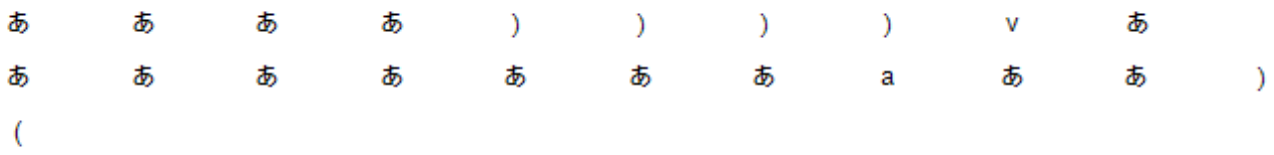
The default presentation would allow characters with identical compression rules to compress and utilize a single slot on the document grid (notice that the four parenthesis on the first line are combined since they can be compressed identically, while the two parenthesis with different compression on line two are not):



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:useEastAsianBreakRules />
</w:compat>
```

Then no character with compression is compressed and instead are snapped to the grid individually, resulting in the following output:



end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.21 doNotUseHTMLParagraphAutoSpacing (Use Fixed Paragraph Spacing for HTML Auto Setting)

This element specifies whether applications should use a fixed definition when interpreting automatic paragraph spacing defined by a value of `true` (or equivalent) on the beforeAutospacing and/or afterAutospacing attributes on the spacing element (§2.3.1.33).

Typically, applications shall interpret these settings to match the behavior of most HTML user agents, mimicking the default spacing above and below an HTML p element without additional spacing information. This element, when present with a val attribute value of `true` (or equivalent), specifies that those two attributes shall result in the following settings for each value:

- beforeAutospacing = 5 points of spacing before
- afterAutospacing = 10 points of spacing after

[*Example*: Consider a WordprocessingML document with a three paragraphs using HTML autospacing, as follows:

```
<w:p>
  <w:pPr>
    <w:spacing w:beforeAutospacing="true" w:afterAutospacing="true" />
  </w:pPr>
  <w:r>
    <w:t>Paragraph One</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:spacing w:beforeAutospacing="true" w:afterAutospacing="true" />
  </w:pPr>
  <w:r>
    <w:t>Paragraph Two</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:spacing w:beforeAutospacing="true" w:afterAutospacing="true" />
  </w:pPr>
  <w:r>
    <w:t>Paragraph Three</w:t>
  </w:r>
</w:p>
```

The default presentation would result in output designed to match that of all common HTML user agents:

Paragraph One.

Paragraph Two.

Paragraph Three.

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:doNotUseHTMLParagraphAutoSpacing />
</w:compat>
```

Then the paragraphs will have exact spacing of 5 points before and 10 points after, resulting in the following output:

Paragraph One.

Paragraph Two.

Paragraph Three.

Notice that the paragraphs are more condensed in the second example. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.22 doNotUseIndentAsNumberingTabStop (Ignore Hanging Indent When Creating Tab Stop After Numbering)

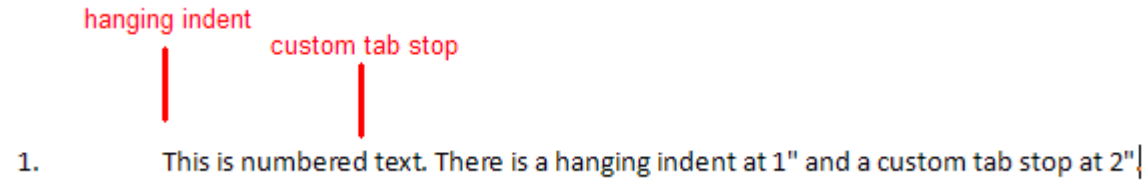
This element specifies whether applications shall use the custom tab stop generated by the hanging indent (if any) when advancing the text after the numbering for a numbered paragraph.

Typically, a hanging indent on a paragraph creates a virtual custom tab stop at that location, and therefore a tab added after the numbering on a numbered paragraph by the suff element (§2.9.30) shall advance to that tab stop, so that the text of the numbered paragraph begins at that location. This element, when present with a val attribute value of `true` (or equivalent), specifies that a tab stop added as the suffix to the numbering of a numbered paragraph shall ignore that virtual custom tab stop and shall instead advance to the next real tab stop (custom or automatic) on the current line.

[*Example*: Consider a WordprocessingML document with numbering, whose first level of numbering specifies a tab stop suffix, a hanging indent at 1", and a custom tab stop at 2":

```
<w:abstractNum w:numId="0">
  …
  <w:lvl w:ilvl="0">
    <w:suff w:val="tab" />
    <w:pPr>
      <w:ind w:left="1440" w:hanging="1440" />
      <w:tabs>
        <w:tab w:val="2880" />
      </w:tabs>
    </w:pPr>
  </w:lvl>
</w:abstractNum>
```

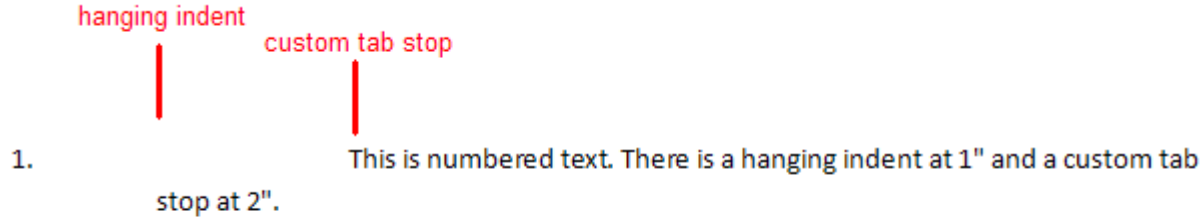
The default presentation of this document results in the tab stop generated by the numbering advancing to the virtual tab stop generated by the hanging indent at 1", as follows:



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:dontUseIndentAsNumberingTabStop />
</w:compat>
```

Then that tab suffix ignores the virtual tab stop of the hanging indent, so it must advance to the next custom tab stop on the line (at 2"), resulting in the following output:



end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

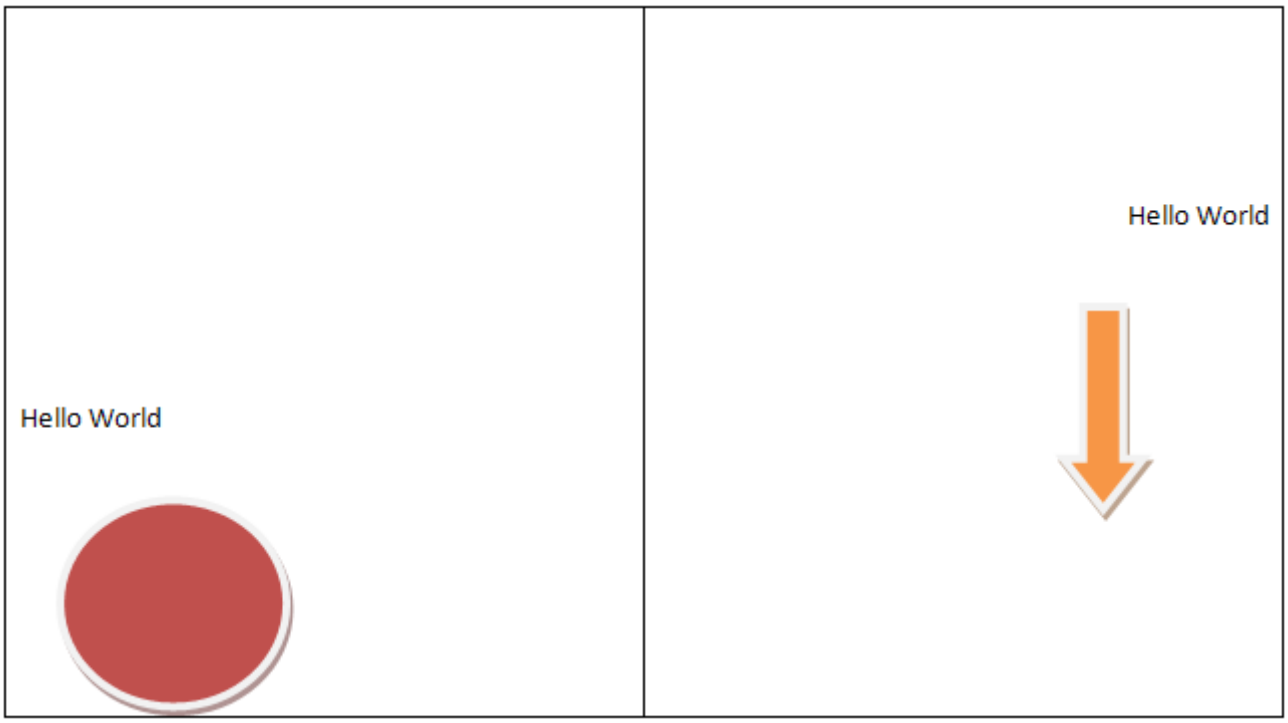
2.15.3.23 doNotVertAlignCellWithSp (Don't Vertically Align Cells Containing Floating Objects)

This element specifies whether applications shall vertically align the contents of a table cell, even when the contents of that table cell include one or more floating objects defined using the Vector Markup Language syntax. Note that the floating object must be part of the cell, and simply not displayed over the cell due to its anchoring relative to another part of the document.

Typically, if the alignment of a table cell in a WordprocessingML document is specified, then the entire contents of that cell are aligned as specified [*Example*: The entire contents of the cell are centered vertically and moved right-aligned horizontally at that point. *end example*]. This element, when present with a val attribute value of true (or equivalent), specifies that whenever a floating object defined using VML is present in a table cell, that no vertical alignment shall be applied to the contents of that cell, and the contents of the cell shall instead always be top aligned to the cell's contents.

[*Example*: Consider a WordprocessingML table with two cells, each containing some text and a single shape defined using the Vector Markup Language syntax. The first cell is vertically aligned to the bottom of the cell, and the second cell is vertically aligned to the center of the cell.

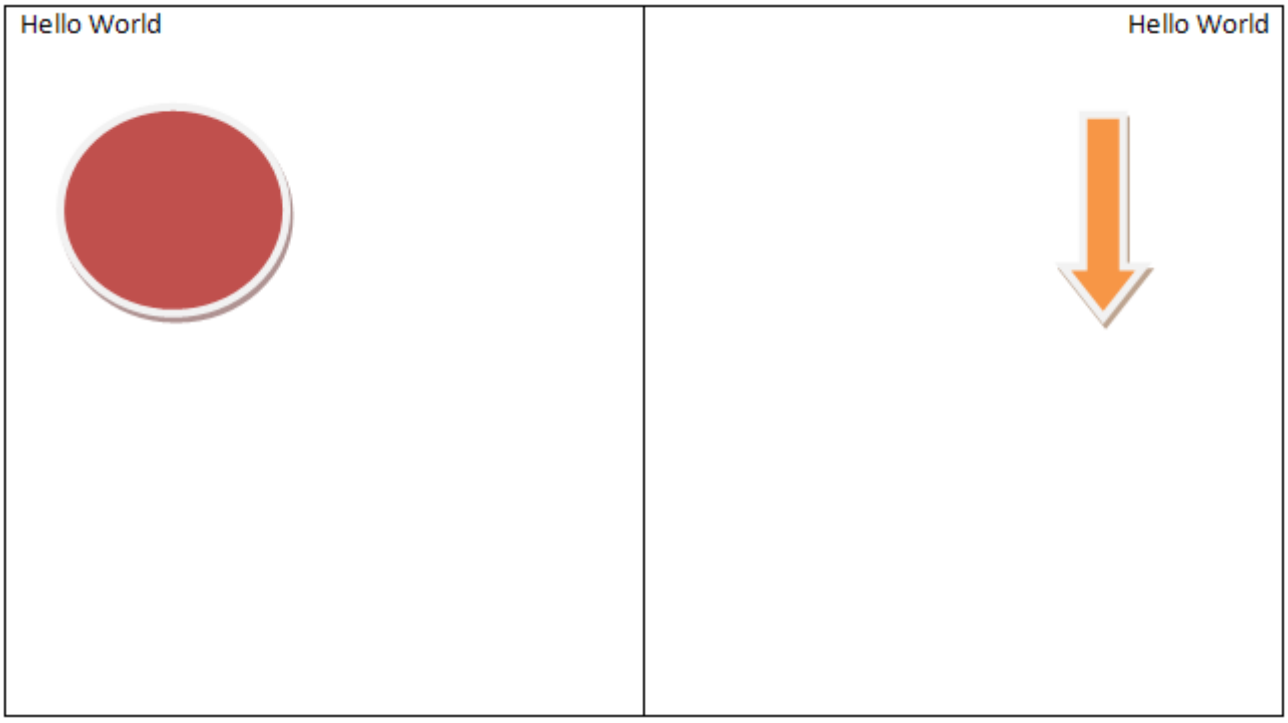
The default presentation of this document results in each cell (including the extents of the floating objects) being vertically aligned as specified, as follows:



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:dontVertAlignCellWithSp />
</w:compat>
```

Then the presence of a floating object in each cell shall result in the vertical alignment setting being ignored (each vertical alignment shall be top-aligned relative to the cell), resulting in the following output:



end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.24 doNotVertAlignInTxbx (Ignore Vertical Alignment in Textboxes)

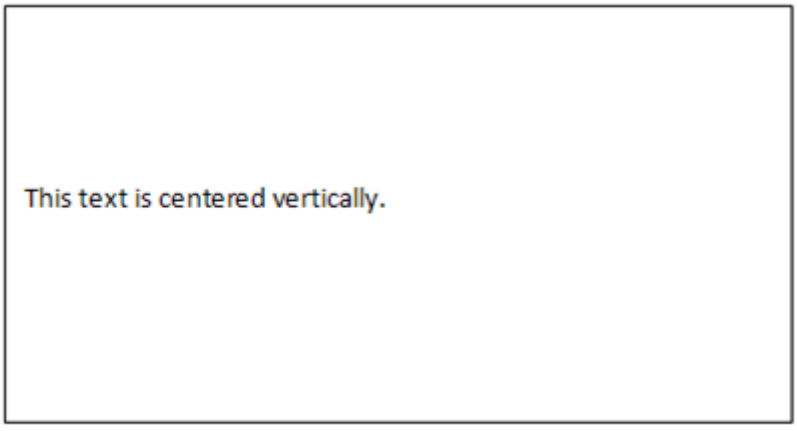
This element specifies whether applications shall allow text within text boxes to be vertically aligned when the `v-text-anchor` property is set within the parent VML shape.

Typically, if when the `v-text-anchor` property is set within the parent VML shape, then based on the value of that property, the text is top, center, or bottom aligned appropriately. This element, when present with a val attribute value of `true` (or equivalent), specifies that the property shall be ignored, and instead the contents of the table shall always be top-aligned.

[*Example*: Consider a WordprocessingML table with a single center-aligned text box:

```
<v:shape id="_x0000_s1026" type="#_x0000_t202" style="v-text-anchor:middle">
  <v:textbox>
    <w:txbxContent>
      <w:p>
        <w:r>
          <w:t>This text is centered vertically.</w:t>
        </w:r>
      </w:p>
    </w:txbxContent>
  </v:textbox>
</v:shape>
```

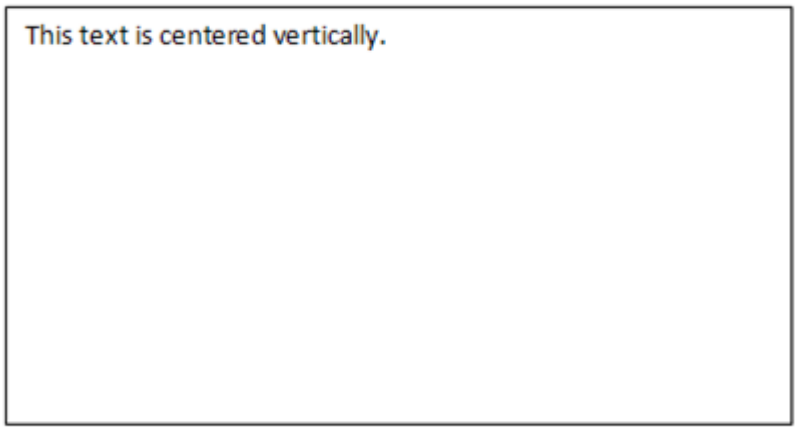
The default presentation of this document results in the contents of the text box being center aligned, as follows:



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:doNotVertAlignInTxbx />
</w:compat>
```

Then the text shall always be top aligned, regardless of the `-text-anchor` property, resulting in the following output:



end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.25 doNotWrapTextWithPunct (Do Not Allow Hanging Punctuation With Character Grid)

This element specifies whether applications shall allowing hanging punctuation when:

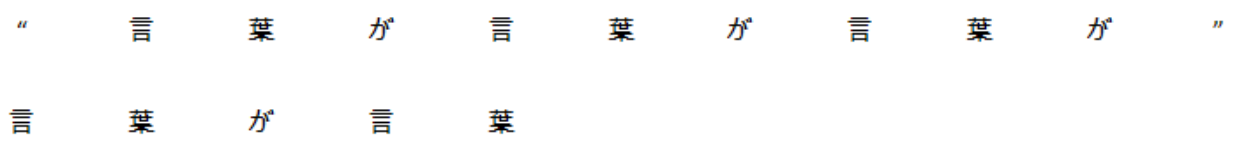
- The overflowPunct element (§2.3.1.21) is turned on for a paragraph
- A document grid is defined using the docGrid element (§2.6.5) which defines the number of characters per line

Typically, paragraphs which allow hanging punctuation shall allow the number of characters on a line as specified by the document grid to be exceeded by one in order to allow for hanging punctuation. This element, when present with a val attribute value of `true` (or equivalent), specifies that the document grid shall never be exceeded for hanging punctuation.

[*Example*: Consider a WordprocessingML document with a document grid set to allow 10 characters per line:

```
<w:sectPr>
  <w:docGrid w:type="snapToChars" w:charSpace="146636" … />
</w:sectPr>
```

If the eleventh character on the line was a punctuation characters, the default presentation would allow that character to behave as hanging punctuation on the first line:



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:wrapTextWithPunct />
</w:compat>
```

Then the character grid cannot be exceeded even for the hanging punctuation, resulting in the following output:

“ 言 葉 が 言 葉 が 言 葉

が ” 言 葉 が 言 葉

The hanging punctuation was disallowed, moving it (and the character before it, since that character cannot begin a line) to the following line. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.26 footnoteLayoutLikeWW8 (Emulate Word 6.x/95/97 Footnote Placement)

This element specifies that applications shall emulate the behavior of a previously existing word processing application (Microsoft Word 6.x/95/97) when determining the placement of the contents of footnotes relative to the page on which the footnote reference occurs. This emulation typically involves some and/or all of the footnote being inappropriately placed on the page following the footnote reference.

[*Guidance*: To faithfully replicate this behavior, applications must imitate the behavior of that application, which involves many possible behaviors and cannot be faithfully placed into narrative for this Office Open XML Standard. If applications wish to match this behavior, they must utilize and duplicate the output of those applications. It is recommended that applications not intentionally replicate this behavior as it was deprecated

due to issues with its output, and is maintained only for compatibility with existing documents from that application. *end guidance*]

Typically, applications shall not perform this compatibility. This element, when present with a val attribute value of true (or equivalent), specifies that applications shall attempt to mimic that existing word processing application in this regard.

[*Example*: Consider a WordprocessingML document with a series of footnotes.

If this compatibility setting is turned on:

```
<w:compat>
  <w:footnoteLayoutLikeWW8 />
</w:compat>
```

Then applications should mimic the behavior of Microsoft Word 6.x/95/97 when determining the placement of those footnotes on the displayed page, as needed. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.27 forgetLastTabAlignment (Ignore Width of Last Tab Stop When Aligning Paragraph If It Is Not Left Aligned)

This element specifies how applications should handle the final tab stop on a line when aligning the contents of a paragraph as specified by the jc element (§2.3.1.13) in the paragraph's properties.

Typically, aligning the contents of a paragraph involves the following:

- Determining the layout of that line before the alignment (including all tab stops)
- Aligning the resulting contents of the line

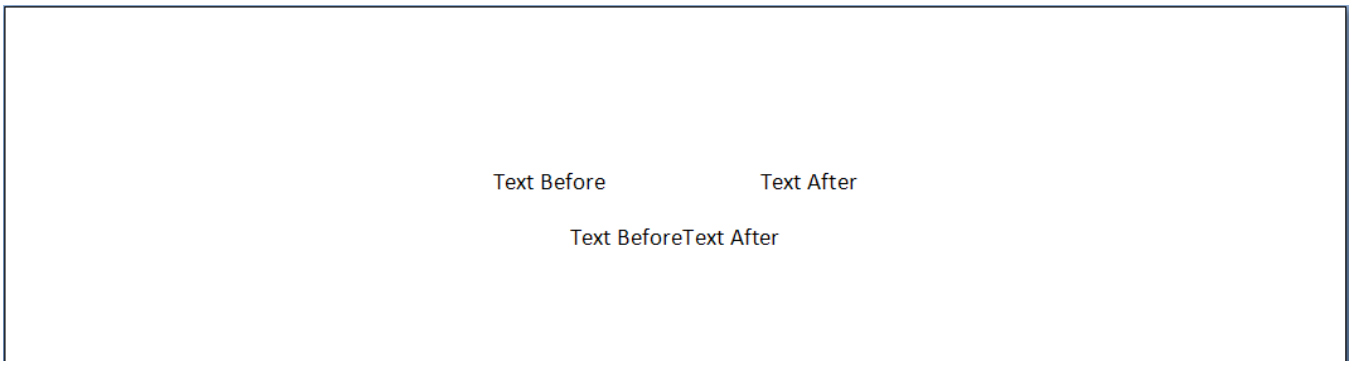
This is done to ensure that tab stops on a line do not change when the contents of the paragraph are aligned (i.e. the tab stops should not have to take into account the paragraph alignment).

This element, when present with a val attribute value of `true` (or equivalent), specifies that applications shall ignore the additional line width generated by the last tab stop (and only the last tab stop) when the alignment of the tab stop as defined by the val attribute on the tab element (§2.3.1.37) is not `left` (or `bar`, which as defined by this Office Open XML Standard, is not a tab stop per se) when determining the width of the line. The resulting full line shall then be aligned at the position where the line would have been aligned without that tab stop.

[*Example*: Consider a WordprocessingML document with two center aligned paragraphs of text - the first also containing a centered tab stop positioned at 2":

```
<w:p>
  <w:pPr>
    <w:tabs>
      <w:tab w:val="center" w:pos="2880" />
    </w:tabs>
    <w:jc w:val="center" />
  </w:pPr>
  <w:r>
    <w:t>Text Before</w:t>
    <w:tab/>
    <w:t>Text After</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:jc w:val="center" />
  </w:pPr>
  <w:r>
    <w:t>Text BeforeText After</w:t>
  </w:r>
</w:p>
```

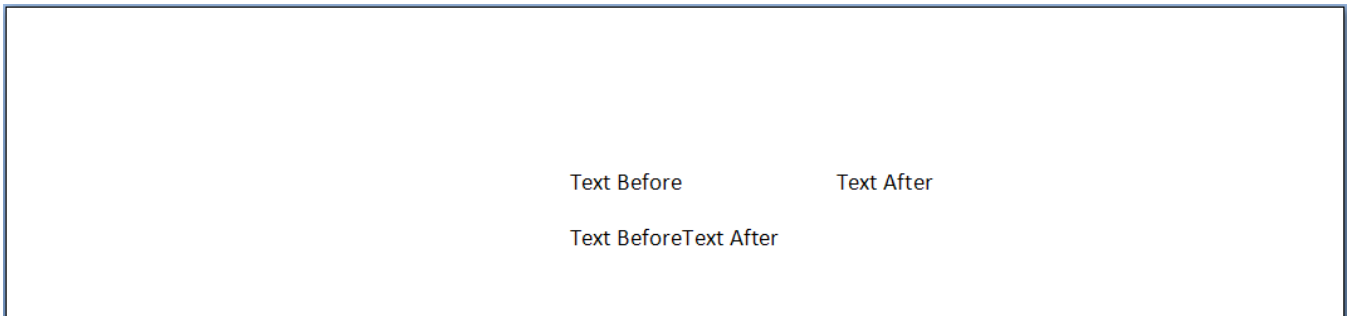
The default presentation would determine the full width of each line including the tab stops, finally aligning the resulting text to the `center` position as requested by the jc element:



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:forgetLastTabAlignment />
</w:compat>
```

Then the width added to the line by the last tab is ignored when centering the paragraph because that tab is a `center` aligned tab stop, resulting in the following output:



In the resulting output, the starting location of both lines is at the same place on the page, as the resulting width of both lines is identical when the tab stop is removed from the line width calculation. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted.

Attributes	Description
	A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

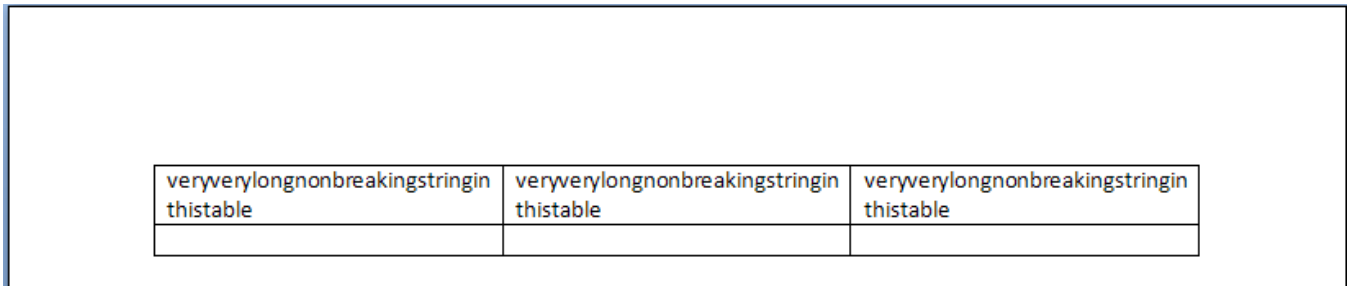
2.15.3.28 growAutofit (Allow Tables to AutoFit Into Page Margins)

This element specifies whether applications shall allow a table which is using the AutoFit table layout algorithm to extend beyond the margins of the page if the minimum width of each table cell would result in an overall table width which is wider than those page margins.

Typically, if a table is using the AutoFit layout algorithm, then based on the definition of that logic, each column in the table shall be increased to the minimum width of its contents (e.g. the longest non-breaking run of text contained within it and/or the width of an inline image contained in one of its cells) until the overall width of the table reaches that of the text extents on the page, at which point text shall be broken and images shall be clipped as needed to maintain the width of the table at the page width (i.e. the page width is an immutable maximum width for the table). This element, when present with a val attribute value of `true` (or equivalent), specifies that the minimum width of the cells shall not be constrained by the page width, and instead the table shall be allowed to extend into the page margins as needed in order to meet the minimum widths of each of its cells.

[*Example*: Consider a WordprocessingML table with three cells in each row. If the contents of each cell in that first row each contain a long non-breaking string (such that the minimum widths of each cell's contents exceed the page width), then the rules for table AutoFit specify that each cell must be broken proportionally when the overall width of the table reaches the page width.

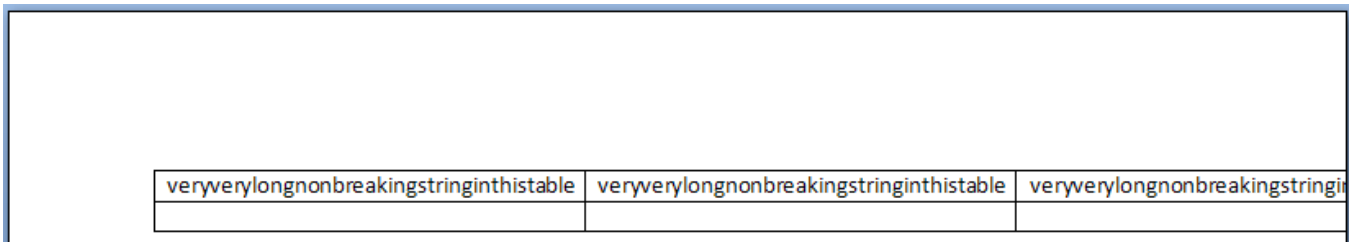
The default presentation of this document results in each cell being broken as needed to maintain the table width, as follows:



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:growAutofit />
</w:compat>
```

Then the presence of those long non-breaking strings (and the resulting large minimum widths for each table cell) shall result in a table width which is then allowed to override the page margins, resulting in the following output:



The resulting table is clipped by the edge of the page on its right side, but the minimum widths of each cell are maintained as defined by the long non-breaking string contents of each. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>`

Attributes	Description
	The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.29 layoutRawTableWidth (Ignore Space Before Table When Deciding If Table Should Wrap Floating Object)

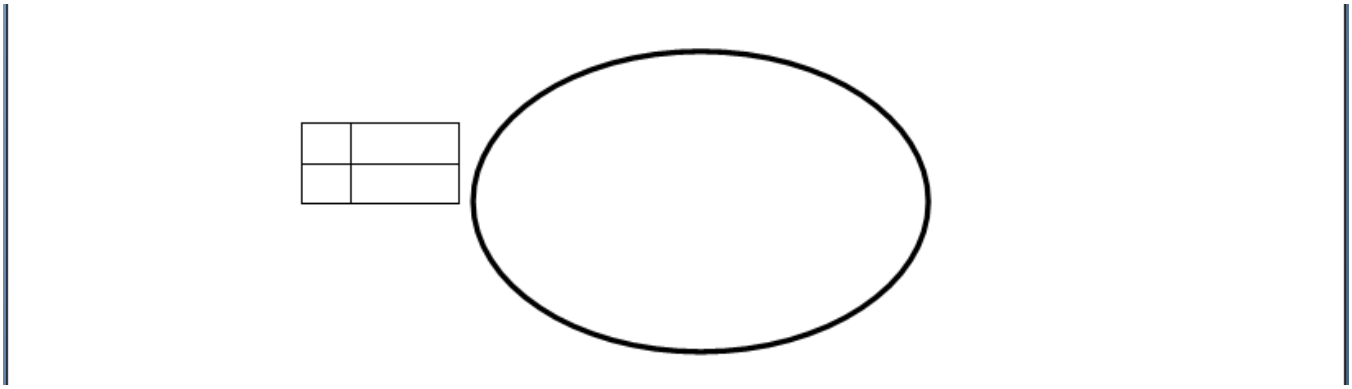
This element specifies how tables which have been indented from the margin using the tblInd element (§2.4.48) shall be wrapped around floating objects defined using the Vector Markup Language (VML) syntax.

Typically, when a table is positioned next to a floating object, the table shall only remain next to the object if it can fit in the remaining space on the line when considering the full width needed for the table: the space before the table, plus the width of the table. This element, when present with a val attribute value of true (or equivalent), specifies that the calculation determining whether the table shall fit next to the object shall not include the space before the table, even if that means that the table is actually clipped by the object.

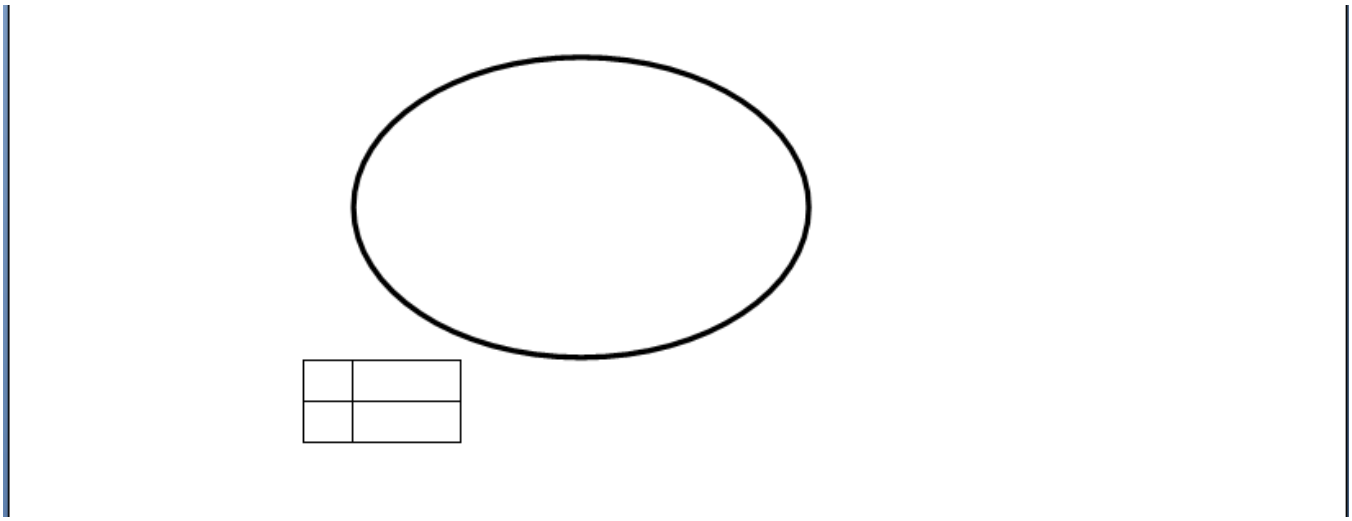
[*Example*: Consider a WordprocessingML document with a floating VML shape using square wrapping, next to a table which has been indented one inch from the left margin:

```
<w:tbl>
  <w:tblPr>
    <w:tblInd w:w="1440" w:type="dxa" />
  </w:tblPr>
  …
</w:tbl>
```

The resulting presentation would place the table next to the object:



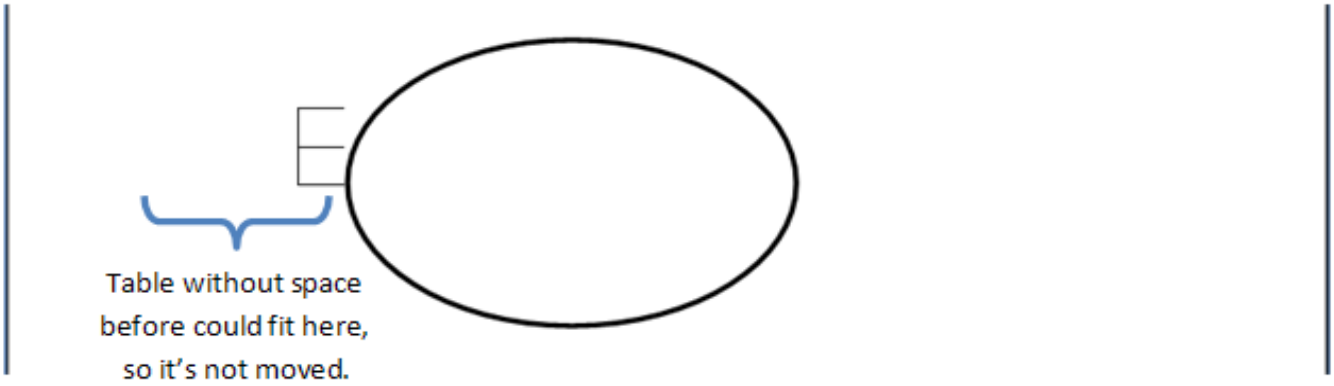
If this object is then moved to the left, such that it would clip the table, the default presentation would have the entire table moved below the shape, since it does not fit in the remaining space on the line:



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:layoutRawTableWidth />
</w:compat>
```

Then the determination to move the table is done ignoring the spaced needed before the table, resulting in the following output:



The resulting table is clipped behind the object, as the fit calculation ignores the space needed before the table. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

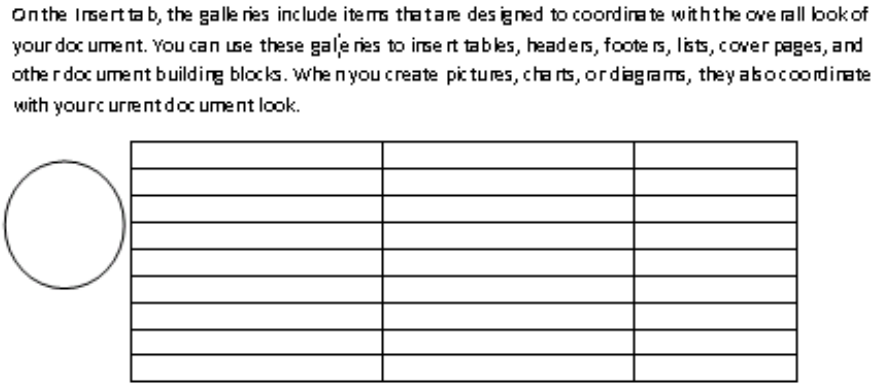
2.15.3.30 layoutTableRowsApart (Allow Table Rows to Wrap Inline Objects Independently)

This element specifies whether tables which are wrapping around floating objects defined using the Vector Markup Language (VML) syntax shall wrap around the object as a whole, or if each table row shall individually wrap the object as needed (causing a more stuttered, yet tighter, wrapping of the object).

Typically, when a table wraps around a floating object, the table must wrap the object as a unit (i.e. the whole table square wraps the object). This element, when present with a val attribute value of true (or equivalent), specifies that wrapping is applied to each row in the table one by one, even if its means that each row has a different resulting position with respect to the table.

[*Example*: Consider a WordprocessingML document with a floating VML shape using square wrapping.

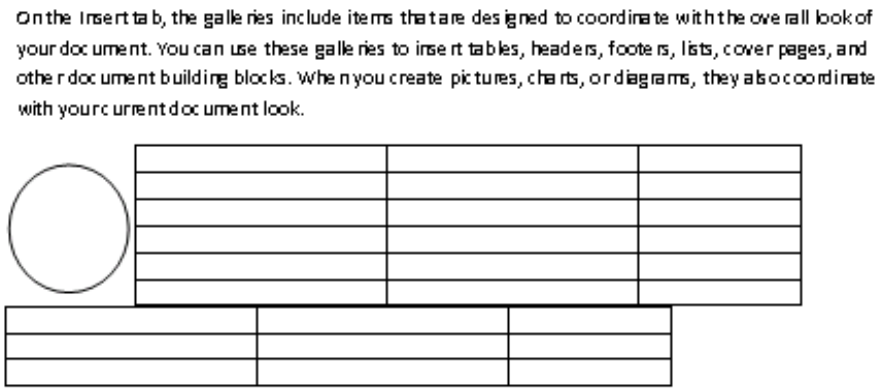
The default presentation would have the entire table wrapping around that shape:



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:layoutTableRowsApart />
</w:compat>
```

Then each row would wrap around the shape one by one, resulting in the following output:



end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but

Attributes	Description
	this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.31 lineWrapLikeWord6 (Emulate Word 6.0 Line Wrapping for East Asian Text)

This element specifies that applications shall emulate the behavior of a previously existing word processing application (Microsoft Word 6.0) when determining the whitespace compression of the final character on each line in the document. This emulation typically results in characters ending a line that may be compressed on the right being compressed on the right irrespective to whether the compression will allow another character to be included on the given line or not.

[*Guidance*: To faithfully replicate this behavior, applications must imitate the behavior of that application, which involves many possible behaviors and cannot be faithfully placed into narrative for this Office Open XML Standard. If applications wish to match this behavior, they must utilize and duplicate the output of those applications. It is recommended that applications not intentionally replicate this behavior as it was deprecated due to issues with its output, and is maintained only for compatibility with existing documents from that application. *end guidance*]

Typically, applications shall not perform this compatibility. This element, when present with a val attribute value of `true` (or equivalent), specifies that applications shall attempt to mimic that existing word processing application in this regard.

[*Example*: Consider a WordprocessingML document with East Asian text.

If this compatibility setting is turned on:

```
<w:compat>
  <w:lineWrapLikeWord6 />
</w:compat>
```

Then applications should mimic the behavior of Microsoft Word 6.0 when determining the character compression of those characters at the end of each line on the displayed page, as needed. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.32 mwSmallCaps (Emulate Word 5.x for the Macintosh Small Caps Formatting)

This element specifies that applications shall emulate the behavior of a previously existing word processing application (Microsoft Word 5.x for the Macintosh) when determining the resulting formatting when the smallCaps element (§2.3.2.31) is applied to runs of text within this WordprocessingML document. This emulation typically results in small caps which are smaller than typical small caps at most font sizes.

[*Guidance*: To faithfully replicate this behavior, applications must imitate the behavior of that application, which involves many possible behaviors and cannot be faithfully placed into narrative for this Office Open XML Standard. If applications wish to match this behavior, they must utilize and duplicate the output of those applications. It is recommended that applications not intentionally replicate this behavior as it was deprecated due to issues with its output, and is maintained only for compatibility with existing documents from that application. *end guidance*]

Typically, applications shall not perform this compatibility. This element, when present with a val attribute value of `true` (or equivalent), specifies that applications shall attempt to mimic that existing word processing application in this regard.

[*Example*: Consider a WordprocessingML document with small caps on its text contents.

If this compatibility setting is turned on:

```
<w:compat>
  <w:mwSmallCaps />
</w:compat>
```

Then applications should mimic the behavior of Microsoft Word 5.x for the Macintosh when determining the formatting for small caps of characters which specify this formatting, as needed.

As an example of the typical differences, the output of a normal small caps implementation (in black) and one intended to replicate Word 5.x for the Macintosh (in red) are displayed below for several font sizes:

end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element.

Attributes	Description
	A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.33 noColumnBalance (Do Not Balance Text Columns within a Section)

This element specifies whether the contents of sections with multiple columns defined using the cols element (§2.6.4) should automatically be balanced. In terms of column layout, *balancing* is the act of attempting to ensure that the number of lines in each column is equivalent (rather than completely filling one column before populating the next).

Typically, column balancing is automatically performed on the contents of sections with multiple columns. This element, when present with a val attribute value of true (or equivalent), specifies that column balancing shall not occur, and each column shall be filled individually until the end of the current page, until all text has been displayed, even if this means one or more columns are unused.

[*Example*: Consider a WordprocessingML document with an initial section with three columns, defined by the following section properties:

```
<w:sectPr>
  <w:cols w:num="3" w:space="720" />
</w:sectPr>
```

The default presentation would have the text in that section balanced between those three columns:

This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns.

This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns.

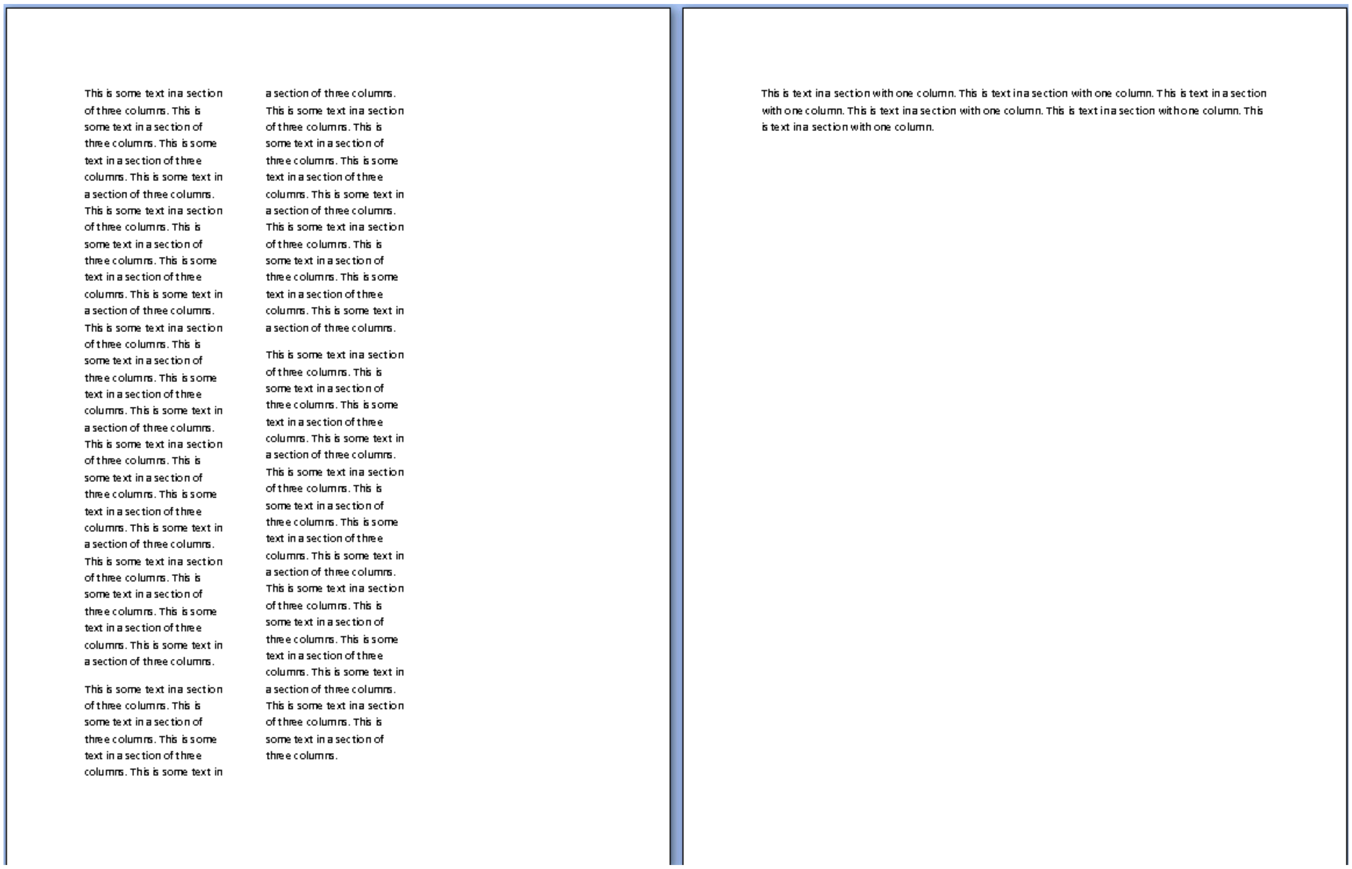
This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns. This is some text in a section of three columns.

This is text in a section with one column. This is text in a section with one column. This is text in a section with one column. This is text in a section with one column. This is text in a section with one column. This is text in a section with one column.

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:noColumnBalance />
</w:compat>
```

Then the columns are not balanced, and the contents of the section are used to fill each column to the bottom of the current page in succession, resulting in the following output:



The next section is now forced to begin on the next page, as the columns on page one extend to the bottom of that page. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.34 noExtraLineSpacing (Do Not Center Content on Lines With Exact Line Height)

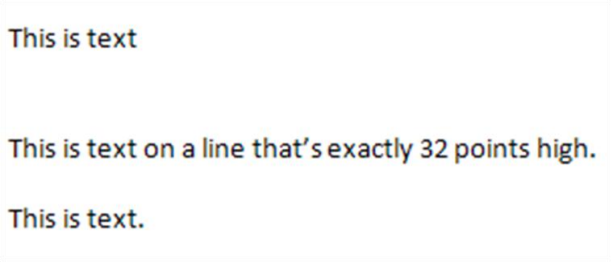
This element specifies whether an exact line height using the spacing element (§2.3.1.33) in the paragraph's properties, each line shall not be automatically centered within the given amount of line spacing.

Typically, if the exact amount of spacing allotted to a line via the paragraph properties exceeds the amount of space required by that line, then the line of text shall be automatically centered when the text of the document is displayed. This element, when present with a val attribute value of true (or equivalent), specifies thatall additional spacing shall instead be placed below the normal layout of the line of text.

[*Example*: Consider a WordprocessingML document with a line with an exact height of 32 points:

```
<w:p>
  <w:pPr>
    <w:spacing w:line="640" w:lineRule="exact" />
  </w:pPr>
  <w:r>
    <w:t>This is text on a line that's exactly 32 points high.</w:t>
  </w:r>
</w:p>
```

The default presentation would have the resulting text centered on that line:



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:noExtraLineSpacing />
</w:compat>
```

Then all line spacing is added after the text, resulting in the following output:

This is text

This is text on a line that's exactly 32 points high.

This is text.

end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.35 noLeading (Do Not Add Leading Between Lines of Text)

This element specifies whether the additional leading specified by the current font face shall be added between each line of text when that text is displayed. *Leading* refers to the additional spacing requested by a particular font in order to ensure that letters on subsequent lines do not display in a fashion where they are positioned too closely together.

Typically, leading should be added as specified by the associated font. This element, when present with a val attribute value of true (or equivalent), specifies that the additional leading specified by the font shall never be output when the text is displayed.

[*Example*: Consider a WordprocessingML document with three lines of text. The default presentation would have the text displayed as follows:

EXAMPLE TEXT

Some text.
Some text.
Some text.

However, if this compatibility setting is turned on:

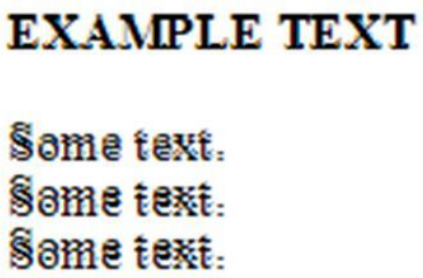
```
<w:compat>
  <w:noLeading />
</w:compat>
```

Then no leading is added between lines, resulting in the following output:

EXAMPLE TEXT

Some text.
Some text.
Some text.

This adjustment is usually very minute in nature; therefore the result is better illustrated by showing how the characters were pushed out due to the leading added to that text:



end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element.

Attributes	Description
	A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.36 noSpaceRaiseLower (Do Not Increase Line Height for Raised/Lowered Text)

This element specifies whether the height which is allotted to any given line of text when the contents of this document are displayed shall include additional spacing in order to ensure that all raised and/or lowered text can be fully displayed.

Typically, any extra space needed is added to the line to prevent raised and lowered text from being truncated or hidden. This element, when present with a val attribute value of true (or equivalent), specifies that the height of the line shall be determined solely by the spacing settings on the parent paragraph, and any raised/lowered text shall just be clipped if it exceeds that space.

[*Example*: Consider a WordprocessingML document with both raised and lowered text. The default presentation would have that text visible:

This is text.

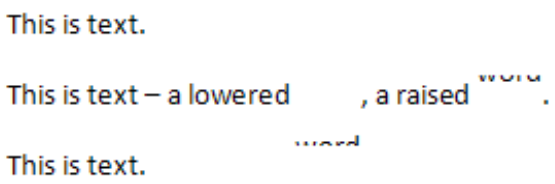
This is text – a lowered word, a raised word.

This is text.

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:noSpaceRaiseLower />
</w:compat>
```

Then no additional space should be added to the line height, resulting in the following output:



end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.37 noTabHangInd (Do Not Create Custom Tab Stop for Hanging Indent)

This element specifies whether applications should always create a hanging indent as a custom tab stop when handling tabs within the contents of a WordprocessingML paragraph. The dontUseIndentAsNumberingTabStop element (§2.15.3.22) specifies if this tab stop shall be used in the case of a tab added as the suffix to numbering in a numbered paragraph, while this element handles the same

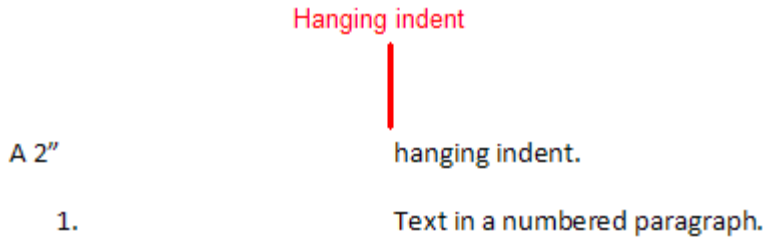
functionality in the generic case (i.e. this element, when set, renders that setting irrelevant as the tab stop is never used).

Typically, the hanging indent on a paragraph shall be treated as a custom tab stop location within that paragraph, allowing the first tab on the first line in the paragraph to advance to the location of the hanging indent. This element, when present with a val attribute value of `true` (or equivalent), specifies that no custom tab stop shall be created for a hanging indent on a line under any circumstances.

[*Example*: Consider a WordprocessingML document with two paragraphs (the second numbered, the first not), each with a 2" hanging indent defined as follows (assume the numbering suffix - not shown - is a tab character):

```
<w:p>
  <w:pPr>
    <w:ind w:left="2880" w:hanging="2880" />
  </w:pPr>
  <w:r>
    <w:t>A 2"</w:t>
    <w:tab/>
    <w:t>hanging indent</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="0" />
      <w:numId w:val="1" />
    </w:numPr>
    <w:ind w:left="2880" w:hanging="2880" />
  </w:pPr>
  <w:t>Text in a numbered paragraph.</w:t>
</w:p>
```

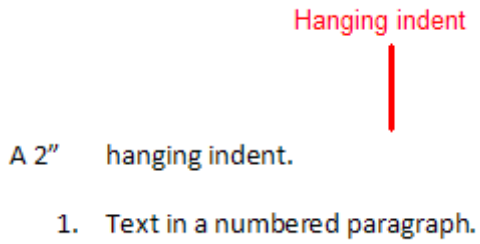
The default presentation would have both the numbering and the tab in the regular paragraph advancing to the 2" custom tab stop generated by the hanging indent:



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:noTabHangInd />
</w:compat>
```

Then no tab stop exists at 2", and therefore the tab stops must advance to the location of the next automatic tab stop for this document (which is set to occur every 0.5"), resulting in the following output:



end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.38 printBodyTextBeforeHeader (Print Body Text before Header/Footer Contents)

This element specifies the order in which the contents of the main document story and any headers and/or footers shall be sent to the printer.

Typically, the contents of a document are sent to the printer as follows:

- First, the contents of headers/footers are sent to the printer
- Finally, the contents of the main document story are sent to the printer

This element, when present with a val attribute value of `true` (or equivalent), specifies that this order shall be reversed, and that the body text shall be sent to the printer before any header/footer text. This reversal allows for the processing of PostScript codes in the text layer in the same order as afforded by some legacy word processing applications.

[*Example*: Consider a WordprocessingML document which is printed. The default resulting print order is the headers and footers for each page, followed by the page contents.

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:printBodyTextBeforeHeader />
</w:compat>
```

Then this order shall be reversed, and the page contents shall be printed before the corresponding header and/or footer for each page. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.39 printColBlack (Print Colors as Black And White without Dithering)

This element specifies the way in which colored text and/or objects shall be handled when printed to a printer whose printer settings indicate that it can only handle black and white text.

Typically, the contents of a colored document are sent to a black and white printer using grayscale (different shades of gray) to represent each of the possible colors. This element, when present with a val attribute value of `true` (or equivalent), specifies that colors will not be printed as mapped shades of grey, but rather exclusively in solid black and white. This setting prevents the fuzzy look that may occur when gray or blue content is dithered. *Dithering* is the process by which colors are simulated using various patterns of black dots on a white background

[*Example*: Consider a WordprocessingML document which is printed to a black and white printer. The default resulting printed content is typically dithered to appear in the appropriate shade of grayscale text.

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:printColBlack />
</w:compat>
```

Then the page contents shall be printed as exclusively black or exclusively white text as needed, and no grayscale output shall occur. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property:

Attributes	Description
	<w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

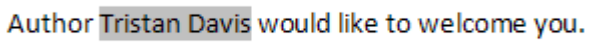
```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.40 selectFldWithFirstOrLastChar (Select Field When First or Last Character Is Selected)

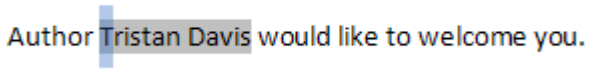
This element specifies whether applications should automatically select the entire contents of a field in a WordprocessingML document when the first or last character is selected.

Typically, users can select any character individually within the result of a field in the document. This element, when present with a val attribute value of true (or equivalent), specifies that selecting the first or last character of that field result shall automatically result in the selection of the entire field.

[*Example*: Consider a WordprocessingML document which contains the following (with a field marked in gray shading):



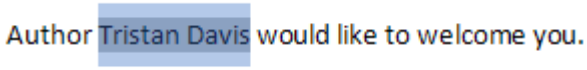
The default presentation would allow the first character of that field to be selected:



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:selectFldWithFirstOrLastChar />
</w:compat>
```

Then that selection would automatically result in the entire field being selected, resulting in the following:



end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.41 shapeLayoutLikeWW8 (Emulate Word 97 Text Wrapping Around Floating Objects)

This element specifies that applications shall emulate the behavior of a previously existing word processing application (Microsoft Word 97) when determining how to wrap text around floating objects using topAndBottom wrapping defined using the Vector Markup Language (VML) syntax. This emulation typically results in the wrapping above and below the object allowing the text to wrap more tightly around the object (text wraps more tightly around the object than it normally would).

[*Guidance*: To faithfully replicate this behavior, applications must imitate the behavior of that application, which involves many possible behaviors and cannot be faithfully placed into narrative for this Office Open XML Standard. If applications wish to match this behavior, they must utilize and duplicate the output of those applications. It is recommended that applications not intentionally replicate this behavior as it was deprecated due to issues with its output, and is maintained only for compatibility with existing documents from that application. *end guidance*]

Typically, applications shall not perform this compatibility. This element, when present with a val attribute value of true (or equivalent), specifies that applications shall attempt to mimic that existing word processing application in this regard.

[*Example*: Consider a WordprocessingML document with a floating shape using top and bottom wrapping.

If this compatibility setting is turned on:

```
<w:compat>
  <w:shapeLayoutLikeWW8 />
</w:compat>
```

Then applications should mimic the behavior of Microsoft Word 97 when determining the way in which text wraps around that object, as needed. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

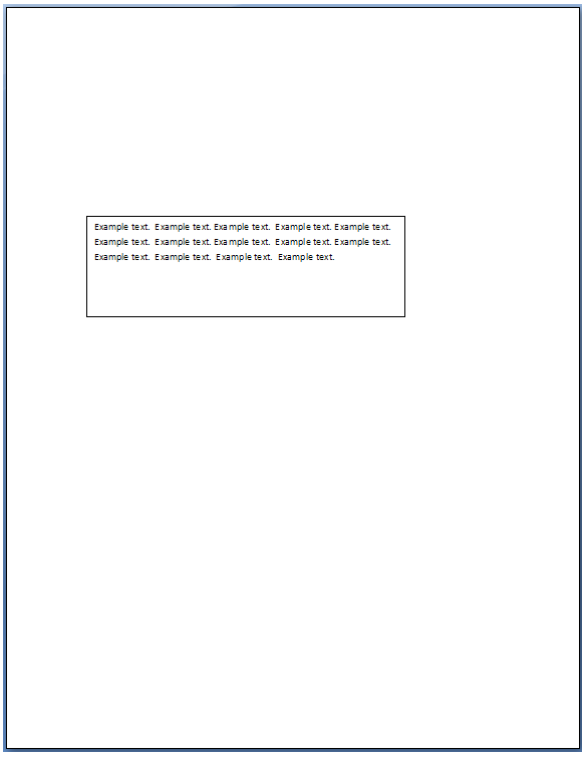
```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.42 showBreaksInFrames (Display Page/Column Breaks Present in Frames)

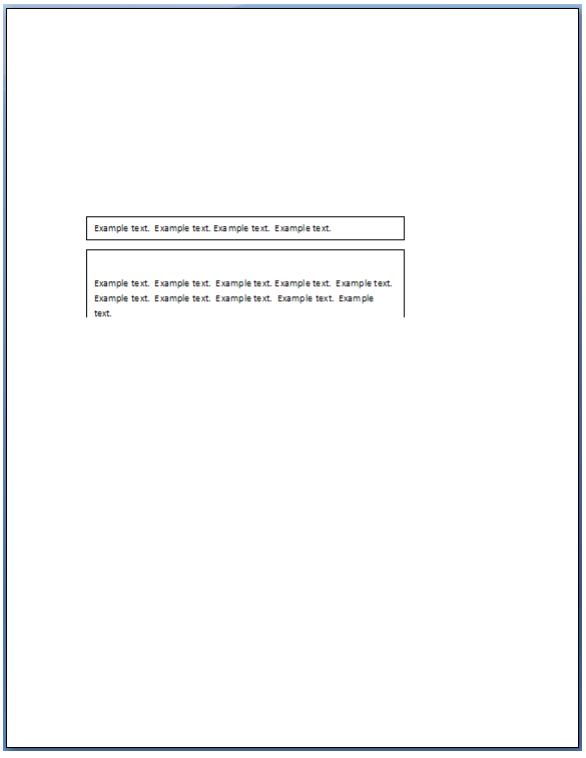
This element specifies whether applications should honor the presence of page and/or column breaks which are present within the contents of paragraphs which have been defined as frames using the framePr element (§2.3.1.11).

Typically, breaks within frames shall be ignored and shall have no effect on the display of the paragraph in which they are contained. This element, when present with a val attribute value of `true` (or equivalent), specifies that rather than completely ignoring these breaks, applications should display the break and move the remaining frame content, and all subsequent text, to the next page and/or column, as needed.

[*Example*: Consider a WordprocessingML document with a paragraph contained within a text frame:



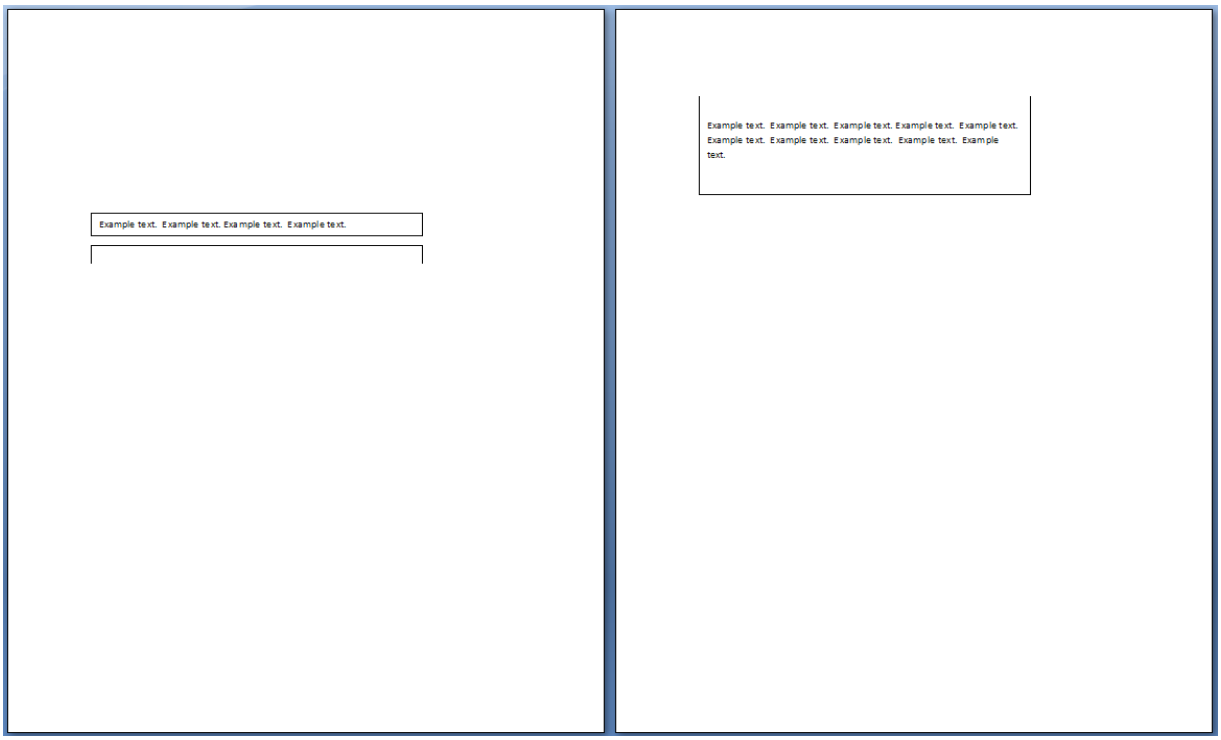
The default presentation would display the page break inline in the frame (breaking the frame into two) but would not actually break the page:



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:showBreaksInFrames />
</w:compat>
```

Then the page breaks will be used even though they are present in the frame, breaking the page and resulting in the following output:



end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property:

Attributes	Description
	<w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.43 spaceForUL (Add Additional Space Below Baseline For Underlined East Asian Text)

This element specifies whether East Asian content in a WordprocessingML document which has been underlined using the u element shall have additional descent added to the properties of the font in order to ensure that there is adequate spacing between the characters in the font and the underlining applied to the text.

Typically, no adjustments are made to the contents of text runs containing East Asian text which have been underlined. This element, when present with a val attribute value of true (or equivalent), specifies that whenever the following conditions are met:

- The text run contains East Asian characters
- The text run is not using baseline font alignment as defined by the textAlignment property

That the larger of the following two values will be added to the descent property of that font in order to provide additional padding between the text characters and the underline:

- 3 percent of the font size
- 40 twentieths of a point (31 twentieths of a point for Japanese text)

[*Example*: Consider a WordprocessingML document consisting of a single run of underlined Japanese text, as follows:

```
<w:p>
  <w:r>
    <w:rPr>
      <w:u w:type="double" />
    </w:rPr>
    <w:t>クリスタ</w:t>
  </w:r>
</w:p>
```

If this document is displayed, then the text is laid out along with the underline, as follows:

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:spaceForUL />
</w:compat>
```

Then the additional descent specified using the logic above is added to the text, resulting in the following output:

end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.44 spacingInWholePoints (Only Expand/Condense Text By Whole Points)

This element specifies how applications should apply text expansion/compression defined using the spacing element (§2.3.2.33) within a set of run properties.

Typically, as defined in the spacing element, text within runs in a WordprocessingML document may be expanded or compressed in increments of twentieths of a point. This element, when present with a val attribute value of true (or equivalent), specifies that the expansion and compression of text shall only be performed in increments of points. Any value which is not equal to an expansion or compression of a whole point shall be rounded down to the nearest whole point when the text is expanded/compressed within the WordprocessingML document.

[*Example*: Consider a WordprocessingML document with three paragraphs of text, each expanded by a varying amount, as follows:

```
<w:p>
  …
  <w:r>
    <w:t>This is text.</w:t>
  </w:r>
</w:p>
<w:p>
  …
  <w:r>
    <w:rPr>
      <w:spacing w:val="20" />
    </w:rPr>
    <w:t>This is text.</w:t>
  </w:r>
</w:p>
<w:p>
  …
  <w:r>
    <w:rPr>
      <w:spacing w:val="36" />
    </w:rPr>
    <w:t>This is text.</w:t>
  </w:r>
</w:p>
```

The default presentation would have each run of text expanded exactly as requested:

Regular Text: This is text.

Text expanded by 1 point: This is text.

Text expanded by 1.8 points: This is text.

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:spacingInWholePoints />
</w:compat>
```

Then the third line - with an expansion of 1.8 points - would instead be rounded down to the nearest whole number of points when expanded, resulting in the following output:

Regular Text: This is text.

Text expanded by 1 point: This is text.

Text expanded by 1.8 points: This is text.

In the resulting output, the second and third lines are identical, as the third line has a next expansion of exactly one point. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.45 splitPgBreakAndParaMark (Always Move Paragraph Mark to Page after a Page Break)

This element specifies whether a page break shall automatically complete the line on which it appears, moving the end of the paragraph to a new line on the next page, or if it shall behave as true run-level content within its current paragraph.

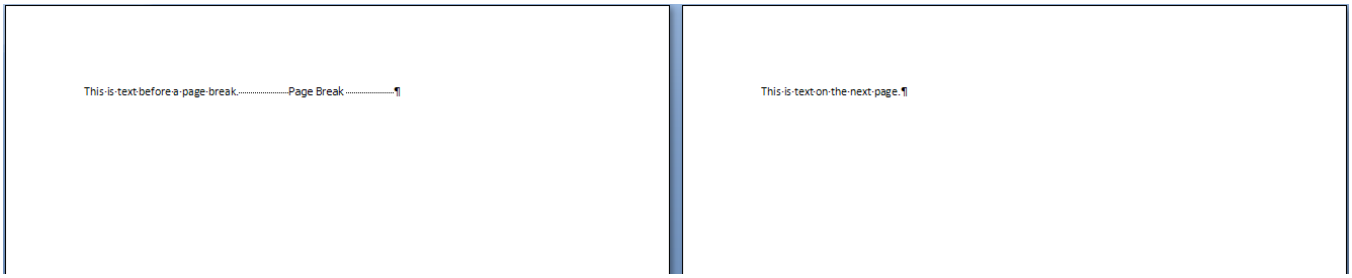
Typically, a page break defined using the br element (§2.3.3.1) is treated as run-level content, which means that although it delimits the end of the page, if there is no content after it within the current paragraph, that the paragraph shall also end on that page. This element, when present with a val attribute value of `true` (or equivalent), specifies that a page break shall always immediately end the current page, moving the paragraph mark which delimits the end of its parent paragraph to a new line on the next page.

Note that this setting only affects the case where there is no run-level content after the page break within the paragraph - if any further run content appears in the paragraph it shall appear on subsequent lines on the next page.

[*Example*: Consider a WordprocessingML document with two paragraphs of content - the first ending with a page break:

```
<w:p>
  <w:r>
    <w:t>This is text before a page break.</w:t>
    <w:br w:type="page" />
  </w:r>
</w:p>
<w:p>
  <w:r>
    <w:t>This is text on the next page.</w:t>
  </w:r>
</w:p>
```

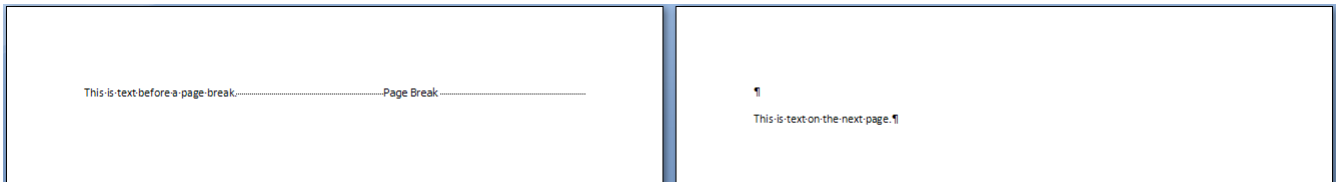
The default presentation would have the text content `This is text on the next page.` as the first line of the second page, as there is no run content after the page break in paragraph one, and therefore no need for a new line on page two (in this image, a graphical illustration of the pilcrow and the page break have been added for clarity):



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:splitPgBreakAndParaMark />
</w:compat>
```

Then even though it is followed by no additional content, the page break shall immediately end the first page, pushing the end of the first paragraph onto the first line of the second page, resulting in the following output:



end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.46 subFontBySize (Increase Priority Of Font Size During Font Substitution)

This element specifies whether applications shall increase the priority of font size when performing font substitution in a WordprocessingML document. *Font substitution* is the process by which an application determines which font to use in place of a font that is referenced by a document, but is not available to the application trying to display the document.

Typically, applications may perform font substitution using any mechanism available. This element, when present with a val attribute value of `true` (or equivalent), specifies that finding a font with a similar font size shall have increased precedence when doing font substitution for this document.

[*Example*: Consider a WordprocessingML document with a series of characters in an unavailable font. The default presentation would use any method used by the application to perform that font substitution.

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:subFontBySize />
</w:compat>
```

Then font size shall take precedence when performing font substitution for this document, as needed. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.47 suppressBottomSpacing (Ignore Exact Line Height for Last Line on Page)

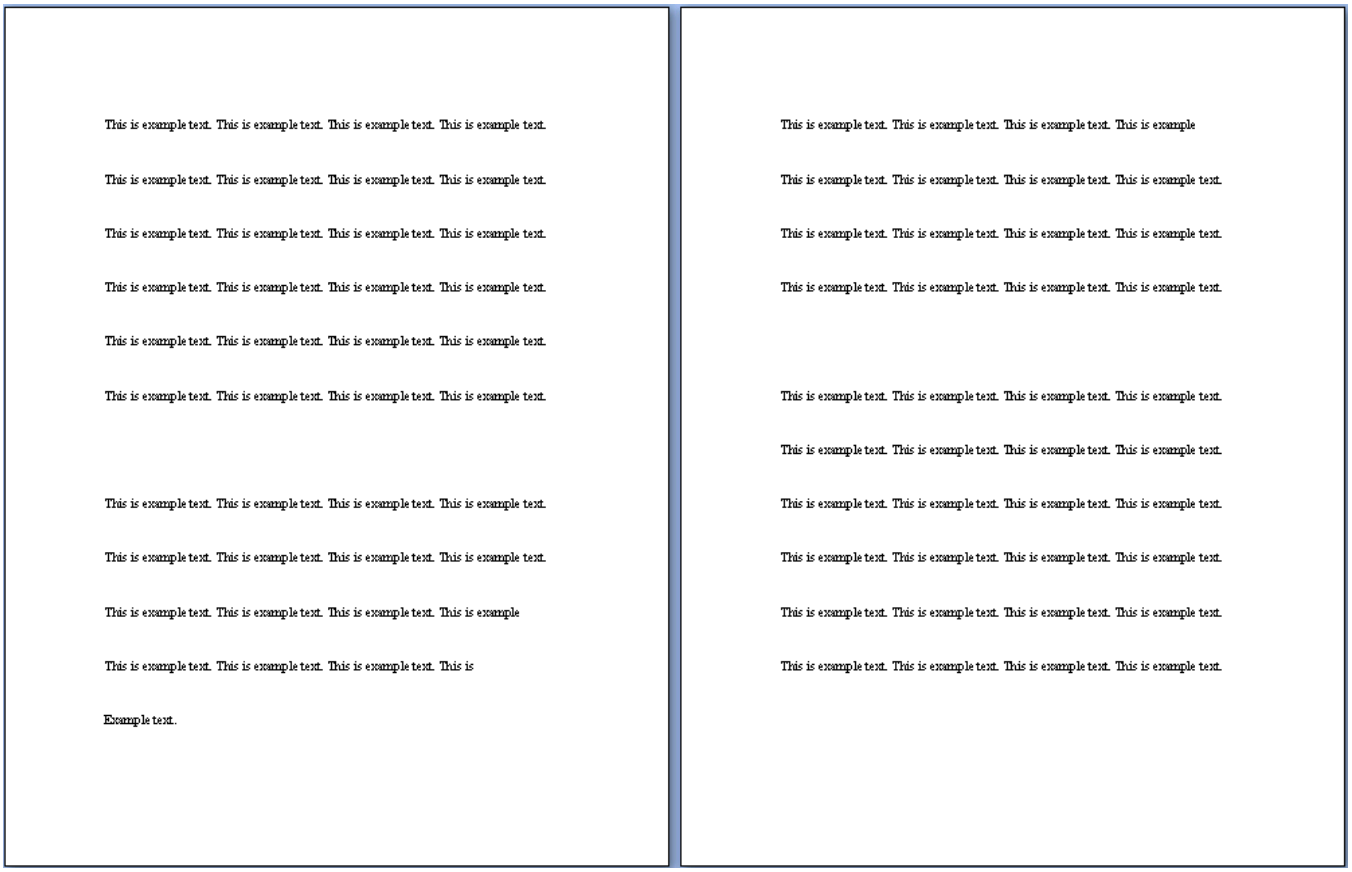
This element specifies whether an exact line height specified using the spacing element (§2.3.1.33) with a lineRule attribute value of exact shall be ignored for the last line on each page.

Typically, if an exact line height has been specified using the spacing element, then all lines within that paragraph have the necessary line spacing added to them in order to meet this constraint. This element, when present with a val attribute value of true (or equivalent), specifies that no additional spacing shall be added below the last line on each page as a result of these line spacing requirements - a line shall be placed on the bottom of the page if its characters fit on that page ignoring the necessary space after.

[*Example*: Consider a WordprocessingML document whose first paragraph has a line spacing setting requiring exactly 48 points of space per line:

```
<w:p>
  <w:pPr>
    <w:spacing w:line="960 w:lineRule="exact" />
  </w:pPr>
  …
</w:p>
```

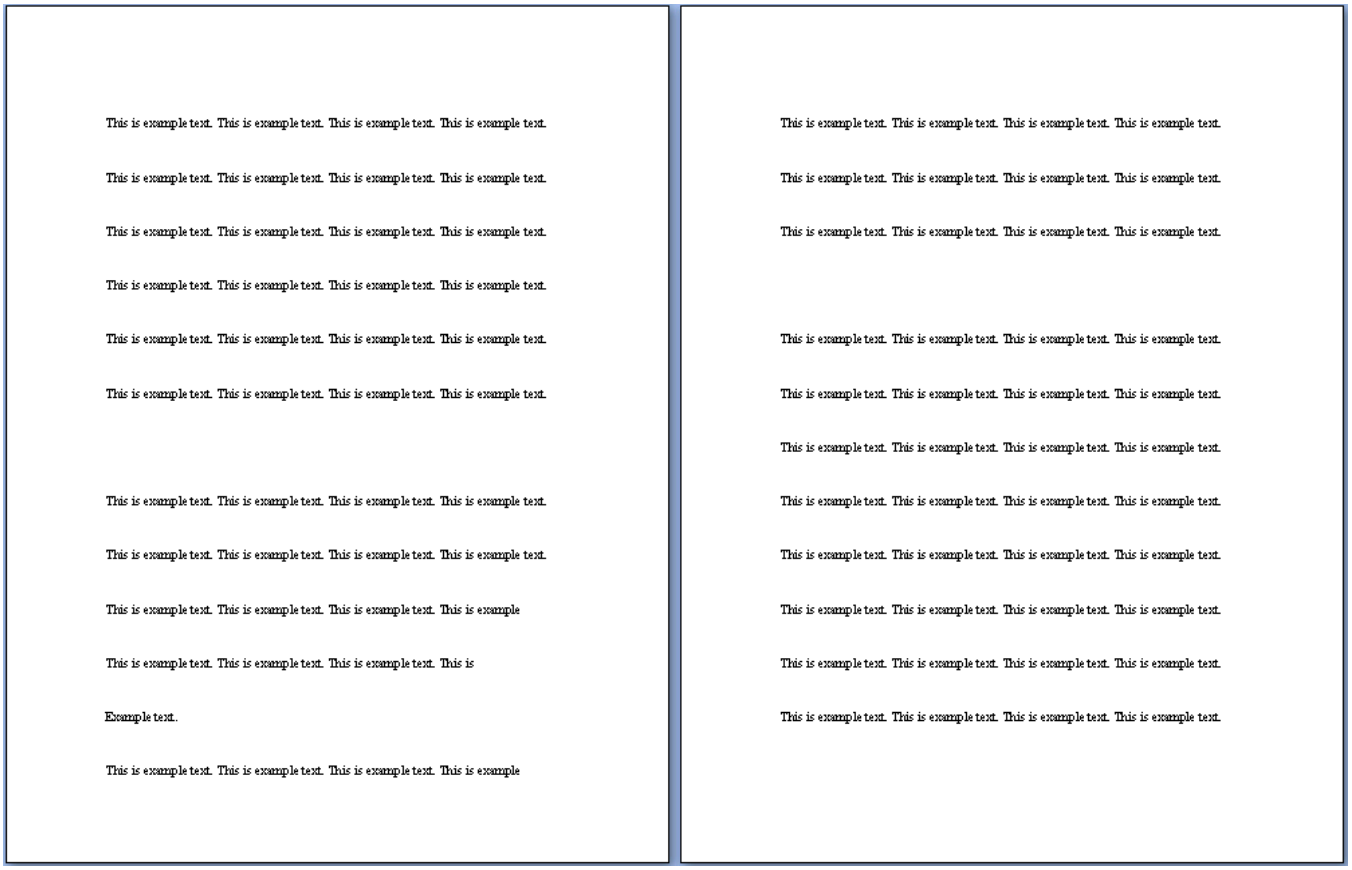
The default presentation would have the necessary amount of space added between each line such that all lines in the paragraph are centered within 48 points of spacing:



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:suppressBottomSpacing />
</w:compat>
```

Then that constraint shall be lifted for the last line on the page (although all other lines are unaffected), resulting in the following output:



The first line from the following page was moved on the first page, as without being subjected to the line height constraint, it is possible to fit it at the bottom of the first page. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.48 suppressSpacingAtTopOfPage (Ignore Minimum Line Height for First Line on Page)

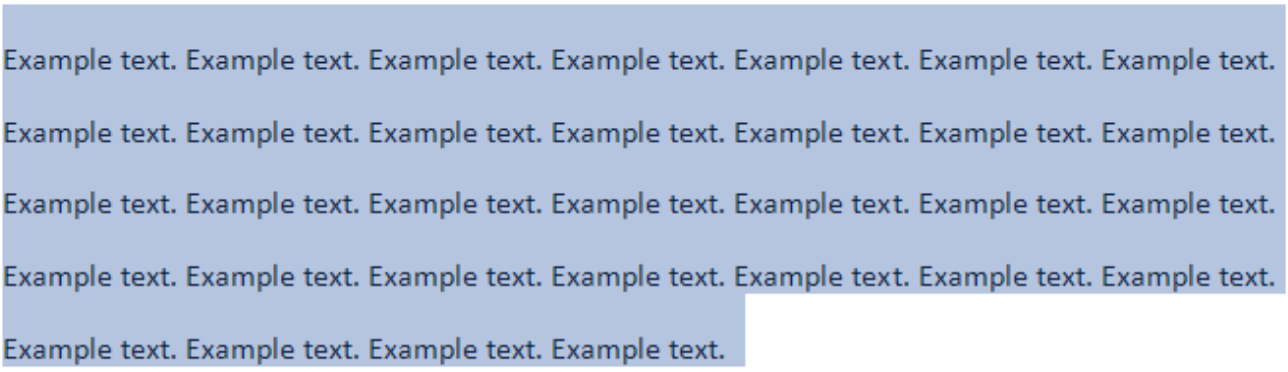
This element specifies whether the minimum line height specified using the spacing element (§2.3.1.33) with a lineRule attribute value of `atLeast` shall be ignored for the first line on each page.

Typically, if a minimum line height has been specified using the spacing element, then all lines within that paragraph have the necessary line spacing added to them in order to meet this constraint. This element, when present with a val attribute value of `true` (or equivalent), specifies that no additional spacing shall be added above the first line on each page as a result of this line spacing requirements - the top of the text characters on the first line shall be at the top edge of the page.

[*Example*: Consider a WordprocessingML document whose first paragraph has a line spacing setting requiring at least 25 points of space per line:

```
<w:p>
  <w:pPr>
    <w:spacing w:line="500" w:lineRule="atLeast" />
  </w:pPr>
  …
</w:p>
```

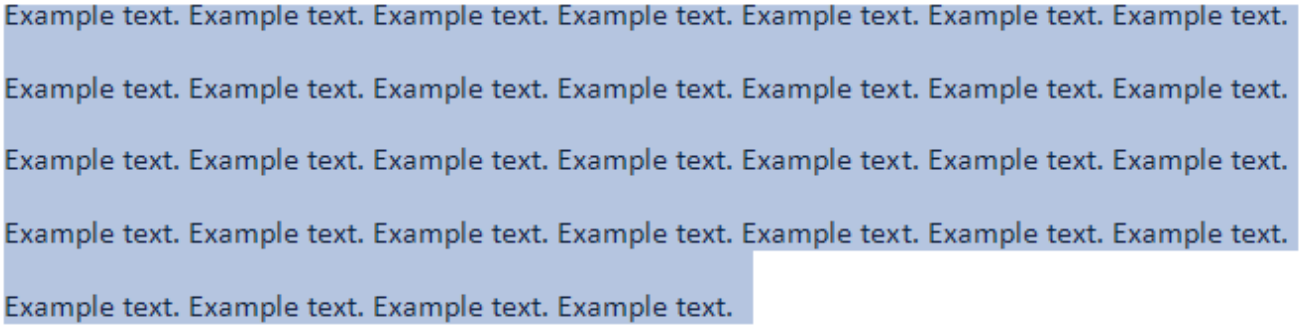
The default presentation would have the necessary amount of space added between each line such that all lines in the paragraph are centered within 25 points of spacing (highlighting has been added to the image below in order to illustrate the additional spacing above the first line):



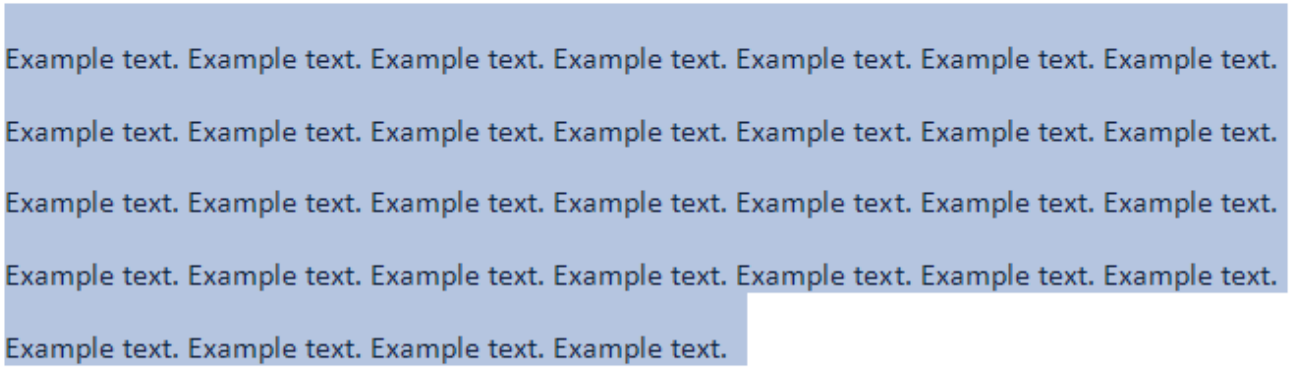
However, if this compatibility setting is turned on:

```
<w:compat>
  <w:suppressSpacingAtTopOfPage />
</w:compat>
```

Then no additional line spacing shall be added above the first line on the page (although all other lines are unaffected), resulting in the following output:



However, if this line spacing constraint was exactly 25 points, then this setting would have no effect:



end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted.

Attributes	Description
	A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.49 suppressSpBfAfterPgBrk (Do Not Use Space Before On First Line After a Page Break)

This element specifies that applications should not postpone any before paragraph spacing to the first line containing content after a page break.

Typically, a page break defined using the br element (§2.3.3.1) is treated as run-level content, which means that although it delimits the end of the page, if there is no content after it within the current paragraph, that the paragraph shall also end on that page. However, in the case where there is additional run-level content within the same paragraph, that content, although part of the same paragraph as the page break, is displayed on the following page.

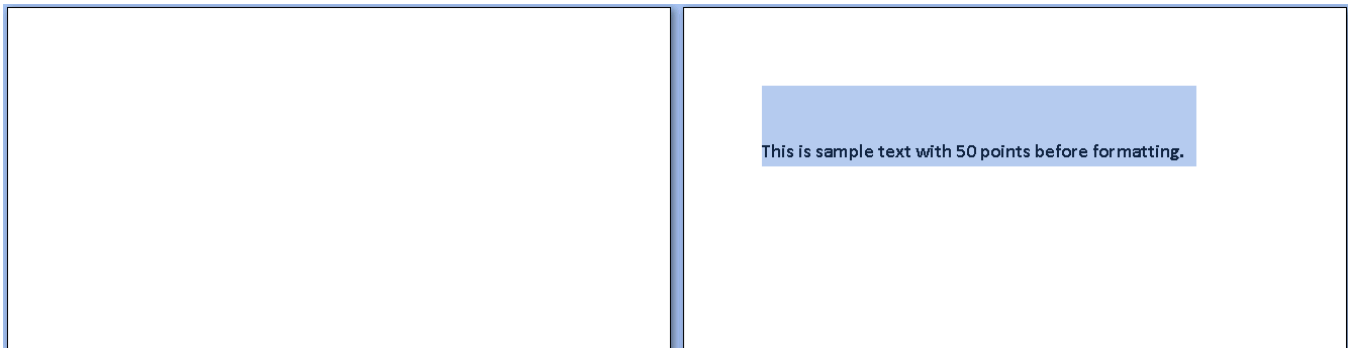
This leads to a situation where the only run content on the page with the page break is the break itself, with all subsequent content on the following page. In this case, applications shall apply the value specified by the spacing element's before attribute to the first line on the new page (since it is ostensibly the only page with content in that paragraph).

This element, when present with a val attribute value of `true` (or equivalent), specifies the paragraph before spacing shall not be 'postponed' in this way - if the line with the page break has no content, then the spacing element's before attribute is simply ignored.

[*Example*: Consider a WordprocessingML document whose first paragraph specifies that it shall be preceded by 50 points of additional spacing:

```
<w:p>
  <w:pPr>
    <w:spacing w:before="1000" />
  </w:pPr>
  <w:r>
    <w:br w:type="page" />
    <w:t>This is sample text with 50 points before formatting.</w:t>
  </w:r>
</w:p>
```

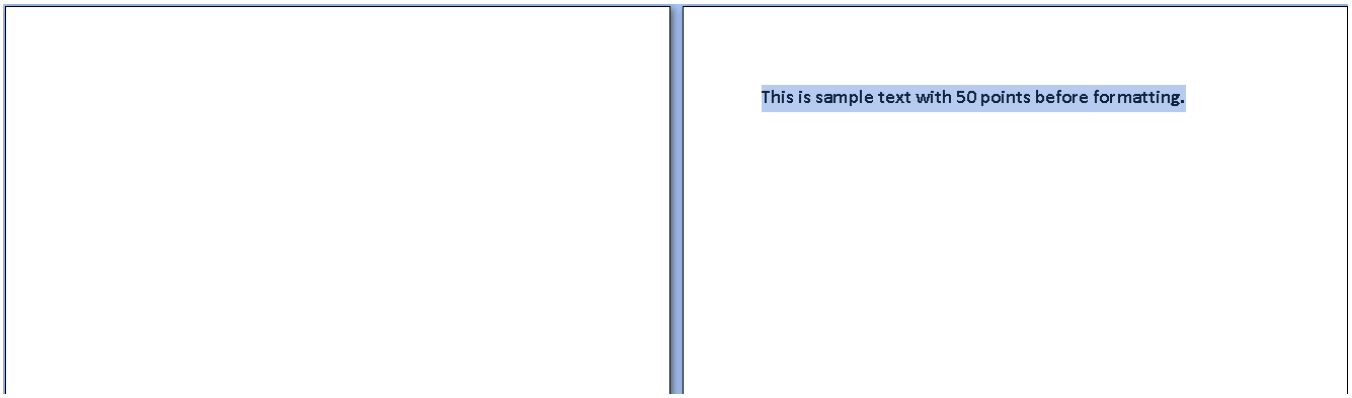
The default presentation would have the necessary amount of space added to the first line on the second page, as the page break was not preceded by any run content (highlighting has been added to the image below in order to illustrate the additional spacing above the first line):



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:suppressSpBfAfterPgBrk />
</w:compat>
```

Then the spacing shall not be added above the first line on the page (it is essentially ignored), resulting in the following output:



end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.50 suppressTopSpacing (Ignore Minimum and Exact Line Height for First Line on Page)

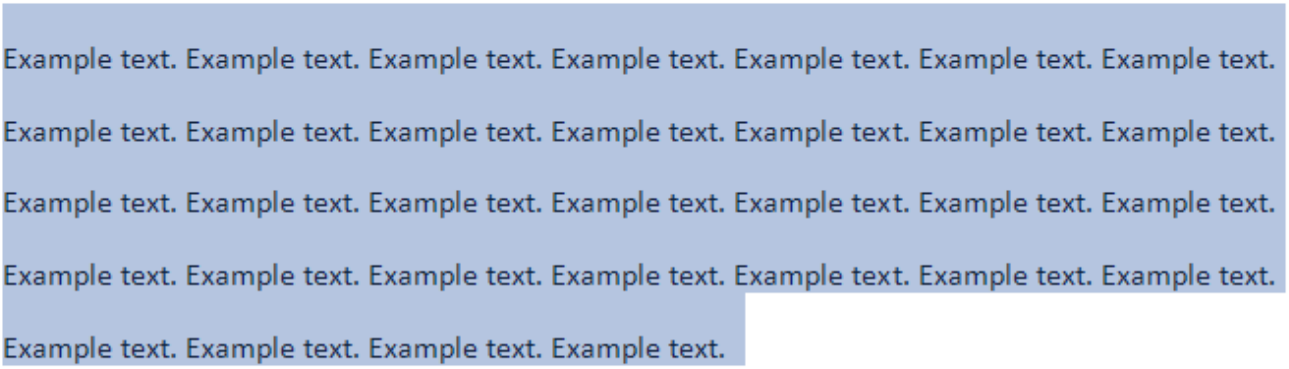
This element specifies whether the minimum line height specified using the spacing element (§2.3.1.33) with a lineRule attribute value of atLeast or exact shall be ignored for the first line on each page.

Typically, if a minimum or exact line height has been specified using the spacing element, then all lines within that paragraph have the necessary line spacing added to them in order to meet this constraint. This element, when present with a val attribute value of true (or equivalent), specifies that no additional spacing shall be added above the first line on each page as a result of these line spacing requirements - the top of the text characters on the first line shall be at the top edge of the page.

[*Example*: Consider a WordprocessingML document whose first paragraph has a line spacing setting requiring exactly 25 points of space per line:

```
<w:p>
  <w:pPr>
    <w:spacing w:line="500" w:lineRule="exact" />
  </w:pPr>
  …
</w:p>
```

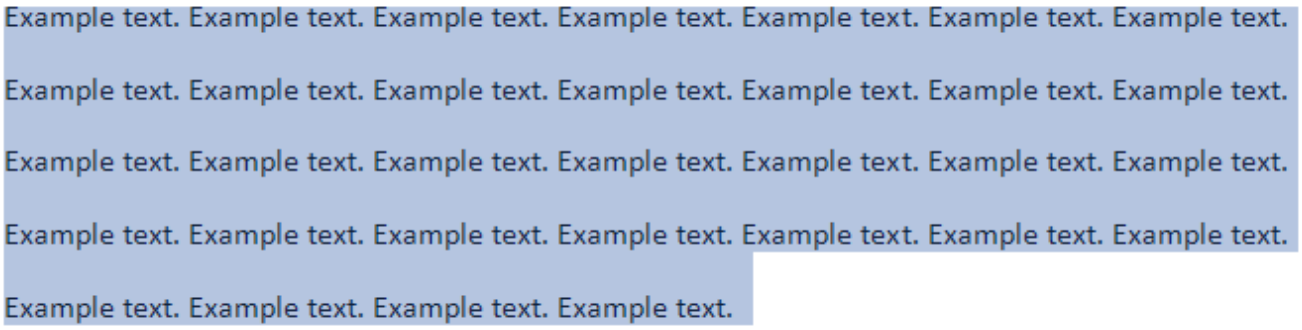
The default presentation would have the necessary amount of space added between each line such that all lines in the paragraph are centered within 25 points of spacing (highlighting has been added to the image below in order to illustrate the additional spacing above the first line):



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:suppressTopSpacing />
</w:compat>
```

Then no additional line spacing shall be added above the first line on the page (although all other lines are unaffected), resulting in the following output:



end example]

Parent Elements

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.51 suppressTopSpacingWP (Emulate WordPerfect 5.x Line Spacing)

This element specifies that applications shall emulate the behavior of a previously existing word processing application (WordPerfect 5.x) when determining the resulting spacing between lines in a paragraph using the spacing element (§2.3.1.33). This emulation typically results in line spacing which is reduced from its normal size.

[*Guidance*: To faithfully replicate this behavior, applications must imitate the behavior of that application, which involves many possible behaviors and cannot be faithfully placed into narrative for this Office Open XML Standard. If applications wish to match this behavior, they must utilize and duplicate the output of those applications. It is recommended that applications not intentionally replicate this behavior as it was deprecated due to issues with its output, and is maintained only for compatibility with existing documents from that application. *end guidance*]

Typically, applications shall not perform this compatibility. This element, when present with a val attribute value of true (or equivalent), specifies that applications shall attempt to mimic that existing word processing application in this regard.

[*Example*: Consider a WordprocessingML document with triple line spacing defined using the spacing element:

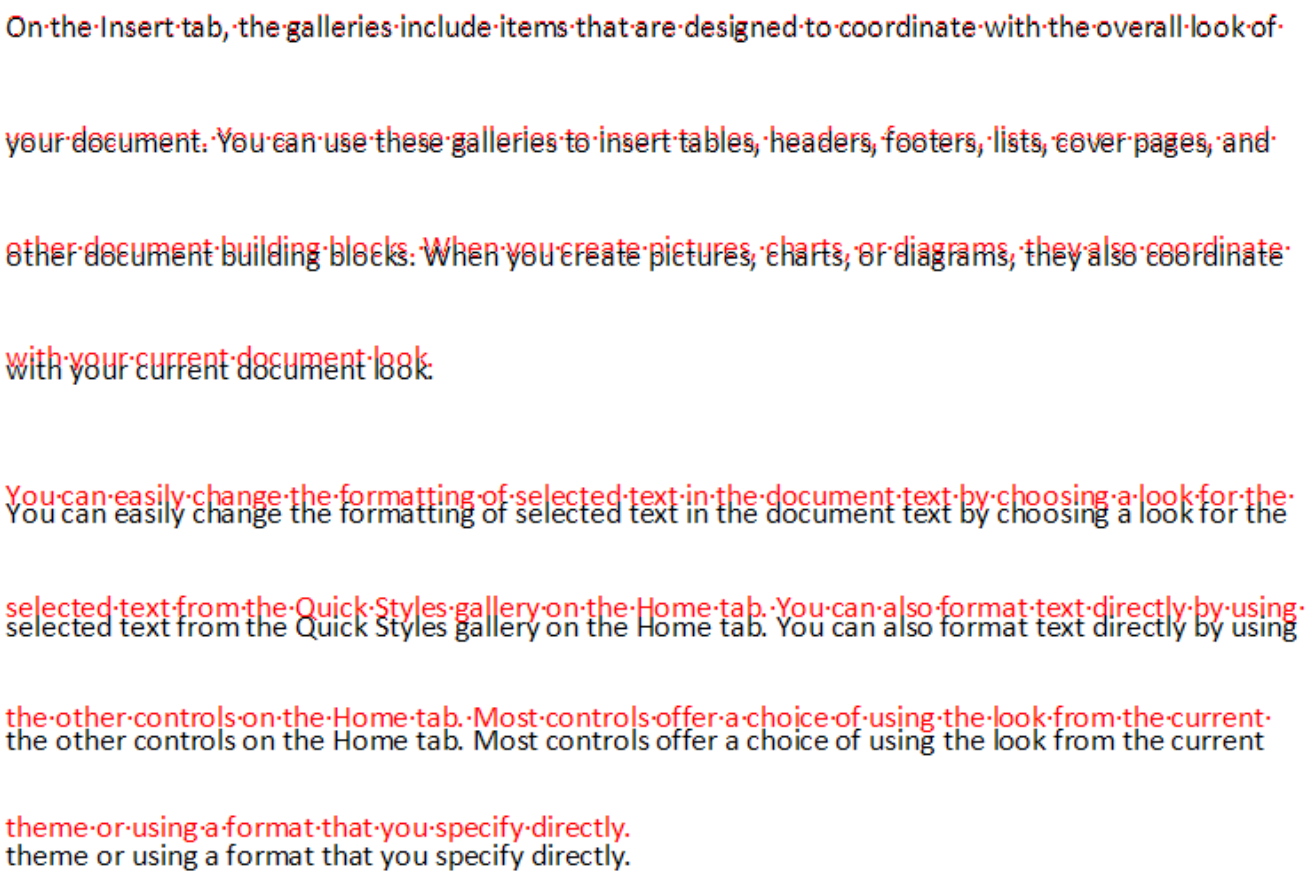
```
<w:pPr>
  <w:spacing w:line="720" w:lineRule="auto" />
</w:pPr>
```

If this compatibility setting is turned on:

```
<w:compat>
  <w:suppressTopSpacingWP />
</w:compat>
```

Then applications should mimic the behavior of WordPerfect 5.x when determining the resulting spacing between each line with additional line spacing, as needed.

As an example of the difference, the output of a normal pairing of triple spaced paragraphs (in black) and one intended to replicate WordPerfect 5.x (in red) is displayed below:



end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.52 swapBordersFacingPages (Swap Paragraph Borders on Odd Numbered Pages)

This element specifies whether left and right paragraph borders defined under the pBdr element (§2.3.1.24) shall be swapped under conditions where it is possible that the those pages are intended to be used to create a book-like publication.

Typically, no changes shall be made to the positions of paragraph borders defined under the pBdr element - a right border is always on the right, and a left border is always on the left. This element, when present with a val attribute value of true (or equivalent), specifies that under the two following conditions:

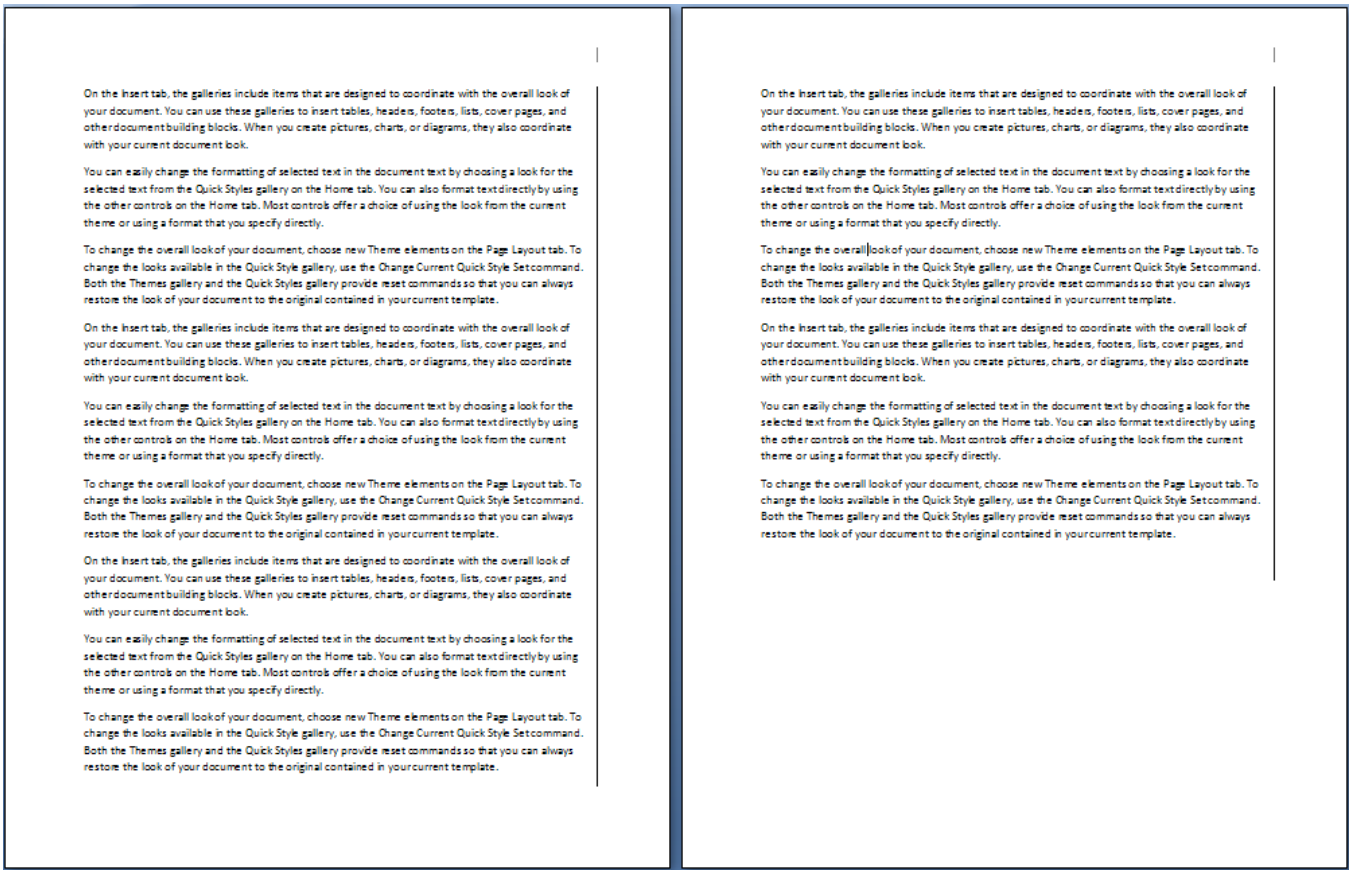
- The margins in this document are mirrored using the mirrorMargins element (§2.15.1.57)
- The header/footers in this document are different on even and odd numbered pages using the evenAndOddHeaders element (§2.10.1)

That paragraph borders on odd-numbered pages will be swapped - that is, left borders shall be displayed on the right and right borders shall be displayed on the left.

[*Example*: Consider a WordprocessingML document for which the mirrorMargins element is present, and whose default paragraph style includes a paragraph border to be displayed on the right side of each paragraph:

```
<w:style w:type="paragraph" w:default="1" w:styleId="Normal" >
  …
  <w:pPr>
    <w:pBdr>
      <w:right w:val="single" w:color="auto" />
    </w:pBdr>
    …
  </w:pPr>
</w:style>
```

If a two-page document is created using this default paragraph style, then all paragraphs will have a border on the right side, as follows:



On the Insert tab, the galleries include items that are designed to coordinate with the overall look of your document. You can use these galleries to insert tables, headers, footers, lists, cover pages, and other document building blocks. When you create pictures, charts, or diagrams, they also coordinate with your current document look.

You can easily change the formatting of selected text in the document text by choosing a look for the selected text from the Quick Styles gallery on the Home tab. You can also format text directly by using the other controls on the Home tab. Most controls offer a choice of using the look from the current theme or using a format that you specify directly.

To change the overall look of your document, choose new Theme elements on the Page Layout tab. To change the looks available in the Quick Style gallery, use the Change Current Quick Style Set command. Both the Themes gallery and the Quick Styles gallery provide reset commands so that you can always restore the look of your document to the original contained in your current template.

On the Insert tab, the galleries include items that are designed to coordinate with the overall look of your document. You can use these galleries to insert tables, headers, footers, lists, cover pages, and other document building blocks. When you create pictures, charts, or diagrams, they also coordinate with your current document look.

You can easily change the formatting of selected text in the document text by choosing a look for the selected text from the Quick Styles gallery on the Home tab. You can also format text directly by using the other controls on the Home tab. Most controls offer a choice of using the look from the current theme or using a format that you specify directly.

To change the overall look of your document, choose new Theme elements on the Page Layout tab. To change the looks available in the Quick Style gallery, use the Change Current Quick Style Set command. Both the Themes gallery and the Quick Styles gallery provide reset commands so that you can always restore the look of your document to the original contained in your current template.

On the Insert tab, the galleries include items that are designed to coordinate with the overall look of your document. You can use these galleries to insert tables, headers, footers, lists, cover pages, and other document building blocks. When you create pictures, charts, or diagrams, they also coordinate with your current document look.

You can easily change the formatting of selected text in the document text by choosing a look for the selected text from the Quick Styles gallery on the Home tab. You can also format text directly by using the other controls on the Home tab. Most controls offer a choice of using the look from the current theme or using a format that you specify directly.

To change the overall look of your document, choose new Theme elements on the Page Layout tab. To change the looks available in the Quick Style gallery, use the Change Current Quick Style Set command. Both the Themes gallery and the Quick Styles gallery provide reset commands so that you can always restore the look of your document to the original contained in your current template.

On the Insert tab, the galleries include items that are designed to coordinate with the overall look of your document. You can use these galleries to insert tables, headers, footers, lists, cover pages, and other document building blocks. When you create pictures, charts, or diagrams, they also coordinate with your current document look.

You can easily change the formatting of selected text in the document text by choosing a look for the selected text from the Quick Styles gallery on the Home tab. You can also format text directly by using the other controls on the Home tab. Most controls offer a choice of using the look from the current theme or using a format that you specify directly.

To change the overall look of your document, choose new Theme elements on the Page Layout tab. To change the looks available in the Quick Style gallery, use the Change Current Quick Style Set command. Both the Themes gallery and the Quick Styles gallery provide reset commands so that you can always restore the look of your document to the original contained in your current template.

On the Insert tab, the galleries include items that are designed to coordinate with the overall look of your document. You can use these galleries to insert tables, headers, footers, lists, cover pages, and other document building blocks. When you create pictures, charts, or diagrams, they also coordinate with your current document look.

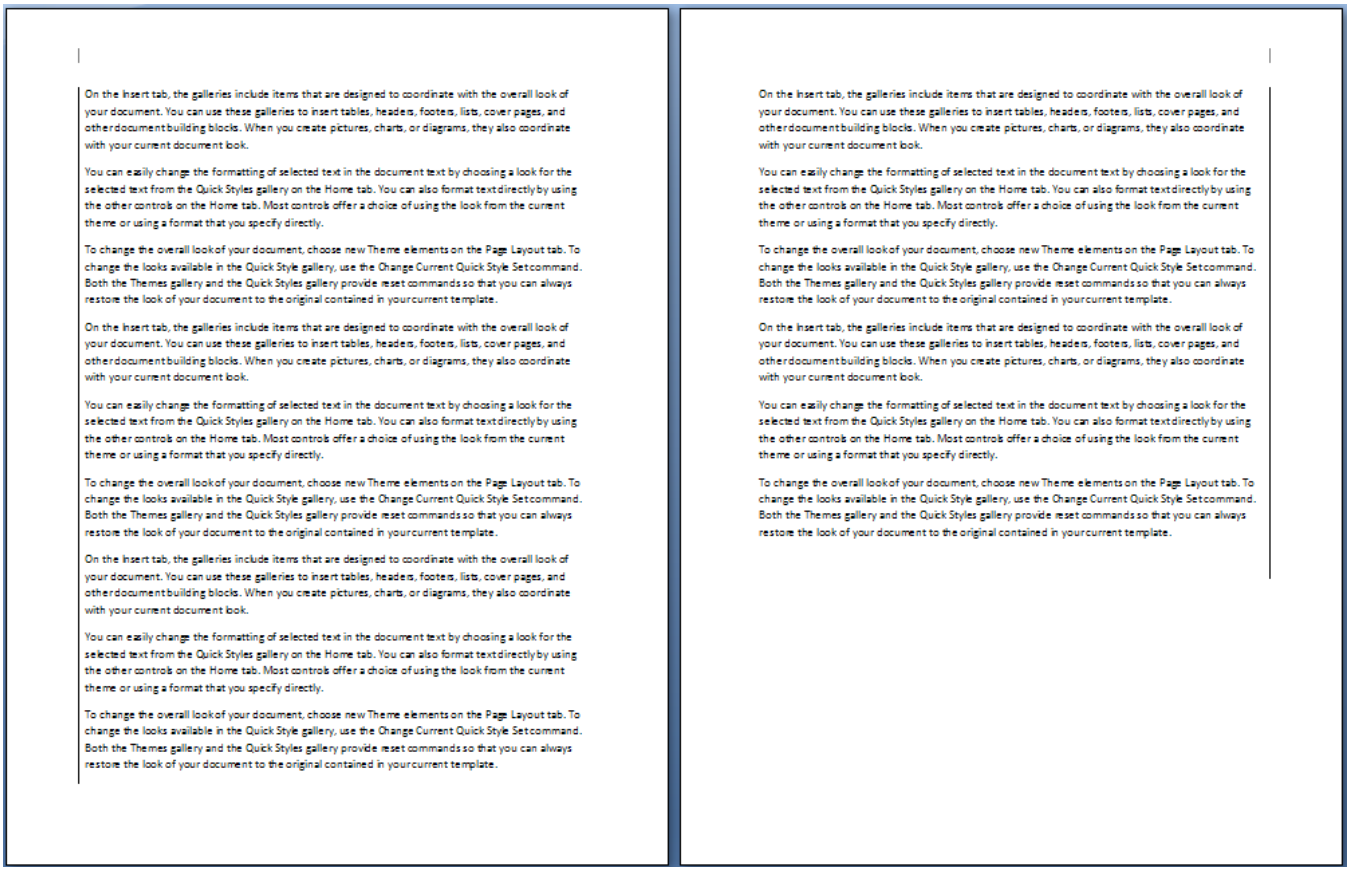
You can easily change the formatting of selected text in the document text by choosing a look for the selected text from the Quick Styles gallery on the Home tab. You can also format text directly by using the other controls on the Home tab. Most controls offer a choice of using the look from the current theme or using a format that you specify directly.

To change the overall look of your document, choose new Theme elements on the Page Layout tab. To change the looks available in the Quick Style gallery, use the Change Current Quick Style Set command. Both the Themes gallery and the Quick Styles gallery provide reset commands so that you can always restore the look of your document to the original contained in your current template.

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:swapBordersFacingPages />
</w:compat>
```

Then the borders on the first page (being an odd-numbered page) shall be swapped, resulting in the following output:



On the Insert tab, the galleries include items that are designed to coordinate with the overall look of your document. You can use these galleries to insert tables, headers, footers, lists, cover pages, and other document building blocks. When you create pictures, charts, or diagrams, they also coordinate with your current document look.

You can easily change the formatting of selected text in the document text by choosing a look for the selected text from the Quick Styles gallery on the Home tab. You can also format text directly by using the other controls on the Home tab. Most controls offer a choice of using the look from the current theme or using a format that you specify directly.

To change the overall look of your document, choose new Theme elements on the Page Layout tab. To change the looks available in the Quick Style gallery, use the Change Current Quick Style Set command. Both the Themes gallery and the Quick Styles gallery provide reset commands so that you can always restore the look of your document to the original contained in your current template.

On the Insert tab, the galleries include items that are designed to coordinate with the overall look of your document. You can use these galleries to insert tables, headers, footers, lists, cover pages, and other document building blocks. When you create pictures, charts, or diagrams, they also coordinate with your current document look.

You can easily change the formatting of selected text in the document text by choosing a look for the selected text from the Quick Styles gallery on the Home tab. You can also format text directly by using the other controls on the Home tab. Most controls offer a choice of using the look from the current theme or using a format that you specify directly.

To change the overall look of your document, choose new Theme elements on the Page Layout tab. To change the looks available in the Quick Style gallery, use the Change Current Quick Style Set command. Both the Themes gallery and the Quick Styles gallery provide reset commands so that you can always restore the look of your document to the original contained in your current template.

On the Insert tab, the galleries include items that are designed to coordinate with the overall look of your document. You can use these galleries to insert tables, headers, footers, lists, cover pages, and other document building blocks. When you create pictures, charts, or diagrams, they also coordinate with your current document look.

You can easily change the formatting of selected text in the document text by choosing a look for the selected text from the Quick Styles gallery on the Home tab. You can also format text directly by using the other controls on the Home tab. Most controls offer a choice of using the look from the current theme or using a format that you specify directly.

To change the overall look of your document, choose new Theme elements on the Page Layout tab. To change the looks available in the Quick Style gallery, use the Change Current Quick Style Set command. Both the Themes gallery and the Quick Styles gallery provide reset commands so that you can always restore the look of your document to the original contained in your current template.

On the Insert tab, the galleries include items that are designed to coordinate with the overall look of your document. You can use these galleries to insert tables, headers, footers, lists, cover pages, and other document building blocks. When you create pictures, charts, or diagrams, they also coordinate with your current document look.

You can easily change the formatting of selected text in the document text by choosing a look for the selected text from the Quick Styles gallery on the Home tab. You can also format text directly by using the other controls on the Home tab. Most controls offer a choice of using the look from the current theme or using a format that you specify directly.

To change the overall look of your document, choose new Theme elements on the Page Layout tab. To change the looks available in the Quick Style gallery, use the Change Current Quick Style Set command. Both the Themes gallery and the Quick Styles gallery provide reset commands so that you can always restore the look of your document to the original contained in your current template.

On the Insert tab, the galleries include items that are designed to coordinate with the overall look of your document. You can use these galleries to insert tables, headers, footers, lists, cover pages, and other document building blocks. When you create pictures, charts, or diagrams, they also coordinate with your current document look.

You can easily change the formatting of selected text in the document text by choosing a look for the selected text from the Quick Styles gallery on the Home tab. You can also format text directly by using the other controls on the Home tab. Most controls offer a choice of using the look from the current theme or using a format that you specify directly.

To change the overall look of your document, choose new Theme elements on the Page Layout tab. To change the looks available in the Quick Style gallery, use the Change Current Quick Style Set command. Both the Themes gallery and the Quick Styles gallery provide reset commands so that you can always restore the look of your document to the original contained in your current template.

end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.53 truncateFontHeightsLikeWP6 (Emulate WordPerfect 6.x Font Height Calculation)

This element specifies that applications shall emulate the behavior of a previously existing word processing application (WordPerfect 6.x) when determining the character height for characters in a font. This emulation typically results slightly truncated character heights.

[*Guidance*: To faithfully replicate this behavior, applications must imitate the behavior of that application, which involves many possible behaviors and cannot be faithfully placed into narrative for this Office Open XML Standard. If applications wish to match this behavior, they must utilize and duplicate the output of those applications. It is recommended that applications not intentionally replicate this behavior as it was deprecated due to issues with its output, and is maintained only for compatibility with existing documents from that application. *end guidance*]

Typically, applications shall not perform this compatibility. This element, when present with a val attribute value of `true` (or equivalent), specifies that applications shall attempt to mimic that existing word processing application in this regard.

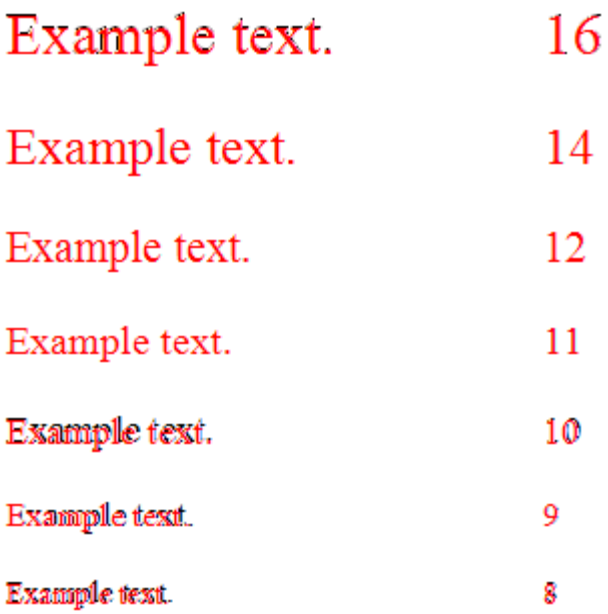
[*Example*: Consider a WordprocessingML document.

If this compatibility setting is turned on:

```
<w:compat>
  <w:truncateFontHeightsLikeWP6 />
</w:compat>
```

Then applications should mimic the behavior of WordPerfect 6.x when determining the height of characters, as needed.

As an example of the difference, the output of a normal pairing of triple spaced paragraphs (in black) and one intended to replicate WordPerfect 6.x (in red) is displayed below:



end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.54 uiCompat97To2003 (Disable Features Incompatible With Earlier Word Processing Formats)

Disable UI functionality that is not compatible with Word97-2003

Parent Elements
settings (§2.15.1.78)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.55 ulTrailSpace (Underline All Trailing Spaces)

This element specifies whether applications shall display underlining beneath all trailing spaces in the contents of a line when those contents are underlined. *Trailing spaces* are all space characters which are not followed by non-space characters on the same line.

Typically, applications do not display underlining on all trailing spaces which have the underline property applied to them. This element, when present with a val attribute value of true (or equivalent), specifies that all characters with underline applied, including trailing spaces, shall display underlining if it is applied to that content.

[*Example*: Consider a WordprocessingML document with the following line of Latin alphabetical character and punctuation, trailed by a series of spaces:

```
<w:r>
  <w:rPr>
    <w:u w:val="single"/>
  </w:rPr>
  <w:t>Example text. Example text. Example text. Example text. Example text.
</w:t>
</w:r>
```

The default presentation would have no underlining on those trailing spaces:

Example text. Example text. Example text. Example text. Example text.

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:ulTrailSpace />
</w:compat>
```

Then all trailing spaces would be underlined, resulting in the following output:

Example text. Example text. Example text. Example text. Example text.

end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.56 underlineTabInNumList (Underline Following Character Following Numbering)

This element specifies whether applications shall underline the character following the numbering defined using the suff element (§2.9.30) when both the numbering itself and the first letter of the corresponding numbered paragraph is underlined.

Typically, the tab or space character generated between numbering and the corresponding paragraph of text is never formatted, since it is automatically generated by the suff element. This element, when present with a val attribute value of `true` (or equivalent), specifies that the tab or space shall tab or space shall be underlined the same way as the numbering symbol itself in the following conditions:

- The numbering is underlined
- The first character of the paragraph is underlined

[*Example*: Consider a WordprocessingML document with two numbered paragraphs: one with underlined text and the other without. The default presentation would have the tab characters free of underlining in both cases:

1. Example Text

2. Example Text

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:underlineTabInNumList />
</w:compat>
```

Then the second paragraph meets the criteria defined above for having the suffix character underlined, resulting in the following output:

1. Example Text

2. Example Text

end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.57 useAltKinsokuLineBreakRules (Use Alternate Set of East Asian Line Breaking Rules)

This element specifies an alternate set of characters which may be used to determine which characters can begin and/or end a line when kinsoku line breaking rules are enabled using the kinsoku element (§2.3.1.16).

Typically, the characters used to determine which characters shall not end a line are those listed by the kinsoku element in the paragraph properties subclause of this document. This element, when present with a val attribute value of true (or equivalent), specifies that the following settings shall be used instead (for brevity, only those settings which are different are listed below):

Chinese (Simplified)

- Cannot start a line:
  !),.:;?]}¨·ˇ¯–‖’”…:、。〃々〉》」』】〕〗！＂＇），．：；？］｀｜｝～¢
- Cannot end a line:
  ([{·‘“〈《「『【〔〖（．［｛£¥

Chinese (Traditional)

- Cannot start a line:
  !),.:;?]}¢·——’”•…‧'-、。〉》」』】〕〞：‖｛ ‿⁀⁔▂﹀﹀﹂﹄﹏‧::?!)}}！），．：；？｜｝、

Korean

- Cannot end a line:
  ([\{£¥“ 〈 《 「 『 【 〔 $ ([{ ₩

[*Example*: Consider a line of text in a WordprocessingML document within a paragraph marked as Chinese (Simplified) which begins with a % symbol, as follows:

```
%...
```

Typically, the kinsoku settings for Chinese (Simplified) do not allow this character to begin a line, so the character before that symbol would be moved down onto this line:

```
〖%...
```

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:useAltKinsokuLineBreakRules />
</w:compat>
```

Then the alternate kinsoku rules are in place, which do not prevent the % character from beginning the new line, resulting in the following output:

```
%...
```

end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.58 useAnsiKerningPairs (Use ANSI Kerning Pairs from Fonts)

This element specifies whether applications shall use the ANSI or Unicode kerning pair information from fonts stored in the document when displaying those characters within the document's contents.

Typically, applications shall use the Unicode kerning pair information in order to determine all possible kerning pairs in the fonts in use. This element, when present with a val attribute value of `true` (or equivalent), specifies that the ANSI kerning information shall be used instead.

[*Example*: Consider a WordprocessingML document with text that contains one or more kerning pairs.

If this compatibility setting is turned on:

```
<w:compat>
  <w:useAnsiKerningPairs />
</w:compat>
```

Then the ANSI kerning pairs are used in place of the Unicode kerning pairs, potentially resulting in different line breaks.

end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property:

Attributes	Description
	`<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.59 useFELayout (Do Not Bypass East Asian/Complex Script Layout Code)

This element specifies that applications shall not bypass code relating to the layout of East Asian and/or Complex Script characters when presenting this document.

[*Guidance*: Previous word processing applications relied on this flag to determine whether to perform functions which allow for the correct layout of East Asian and Complex Script text. Although current applications no longer rely on this flag (as they should correctly use the Unicode subranges and code pages of the text in use), this flag should be output in order to ensure that files with this content can be viewed correctly in previous word processors. *end guidance*]

[*Example*: Consider a WordprocessingML document with East Asian text.

If this compatibility setting is turned on:

```
<w:compat>
  <w:useFELayout />
</w:compat>
```

Then the flag is set telling previous applications that East Asian content is present, and they should display the document accordingly. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off.

Attributes	Description
	[*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.60 useNormalStyleForList (Do Not Automatically Apply List Paragraph Style To Bulleted/Numbered Text)

This element specifies whether applications shall automatically apply the paragraph style with the styleId attribute `ListParagraph` when numbering is applied to a paragraph currently formatted using the default paragraph style.

Typically, when a paragraph is formatted using the default paragraph style, and numbering is subsequently applied, the paragraph style with the styleId attribute `ListParagraph` when numbering is applied to ensure that paragraph properties are appropriate for a numbered paragraph. This element, when present with a val attribute value of `true` (or equivalent), specifies that no alternate paragraph style shall ever be applied

[*Example*: Consider a WordprocessingML document with five unnumbered paragraphs:

Example text.

Example text.

Example text.

Example text.

Example text.

If numbering is applied to the three center paragraphs, the default presentation would have the `ListParagraph` style applied as well:

Example text.

- Example text.
- Example text.
- Example text.

Example text.

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:useNormalStyleForList />
</w:compat>
```

Then the new paragraph style shall not be applied, resulting in the following output:

Example text.

- Example text.
- Example text.
- Example text.

Example text.

end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/>

Attributes	Description
	The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.61 usePrinterMetrics (Use Printer Metrics To Display Documents)

This element specifies whether applications shall use the printer metrics of the currently active printer when determining how to display the contents of a WordprocessingML document. *Printer metrics* are printer-specific settings which can be queried to tell an application how and where text shall be displayed on a printed page.

Typically, applications display the content of a document in a device independent manner - the application is therefore not changing the layout of a document based on the currently attached printer, and instead shall dictate to the printer where characters shall be presented on the page when printed. This element, when present with a val attribute value of true (or equivalent), specifies that the metrics of the current printer shall be used to display the document instead.

Specifically, when this setting is enabled, the printer metrics are used to determine the number of pixels per logical inch along the screen width and height. This should then be used to compute the pixel height of the fonts requested when displaying the document, as well as to scale between any logical units within the document (e.g. drawing object sizes) to the appropriate device units. Those units would then need to be scaled back into screen units for final display to a screen, but not scaled again when displayed to a printer.

[*Note*: On the Windows platform, you can use the GetDeviceCaps function to retrieve device-specific information for the specified printer. For this specific setting, you can use GetDeviceCaps(hdc, LOGPIXELSX) and GetDeviceCaps(hdc, LOGPIXELSY) with a printer DC to retrieve the number of pixels per logical inch along the screen width and height. With this, you can then use those DPI metrics to compute a pixel value for the font request in the LOGFONT structure (the LOGFONT structure defines the attributes of a font). A common formula to do this is $S_{px} = S_{pts} * \frac{LOGPIXELSY}{72}$. *end note*]

[*Example*: Consider a WordprocessingML document. The default shall use device-independent layout to present the contents of the page.

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:usePrinterMetrics />
</w:compat>
```

Then the printer metrics of the current active printer shall be used to determine the display of the contents of the document instead, as needed. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

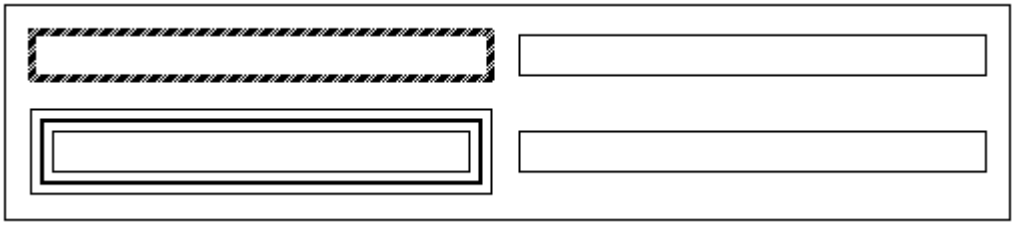
2.15.3.62 useSingleBorderforContiguousCells (Use Simplified Rules For Table Border Conflicts)

This element specifies whether applications should use an alternate simplified algorithm when handling conflicts between adjacent table borders within a table.

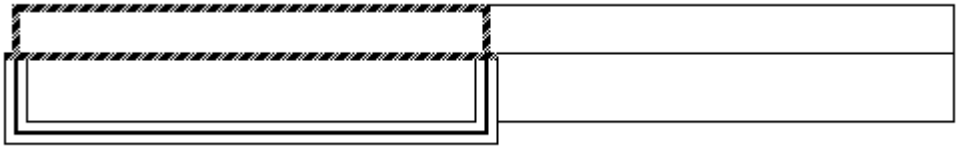
Typically, the conflicts between two adjacent table borders are handled using the conflict resolution algorithm defined in §2.4.38 of this Office Open XML Standard. This element, when present with a val attribute value of true (or equivalent), specifies that rather than using that algorithm to determine the outcome of the conflict to two adjacent borders, that the following logic shall be used instead:

- Cell borders shall supersede table borders
- Cell borders to the right shall supersede cell borders to the left (i.e. the rightmost border wins in conflicts between vertical borders)
- Cell borders below shall supersede cell borders above (i.e. the bottommost border wins in conflicts between horizontal borders)

[*Example*: Consider a WordprocessingML document with cell and table borders defined as follows. In the image below, 0.1" of padding has been added between each cell temporarily to clearly illustrate the borders on each cell and on the table:



The default presentation would have the border conflicts resolved using the algorithm defined by this Office Open XML Standard, resulting in the following table:



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:useSingleBorderForContiguousCells />
</w:compat>
```

Then the simplified table algorithm above shall be used instead (bottom and right cell borders always win), resulting in the following output:

end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property shall be explicitly applied. This is the

Attributes	Description
	default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.63 useWord2002TableStyleRules (Emulate Word 2002 Table Style Rules)

This element specifies that applications shall emulate the behavior of a previously existing word processing application (Microsoft Word 2002) when determining the formatting resulting from table styles applied to tables within a WordprocessingML document.

[*Guidance*: To faithfully replicate this behavior, applications must imitate the behavior of that application, which involves many possible behaviors and cannot be faithfully placed into narrative for this Office Open XML Standard. If applications wish to match this behavior, they must utilize and duplicate the output of those applications. It is recommended that applications not intentionally replicate this behavior as it was deprecated due to issues with its output, and is maintained only for compatibility with existing documents from that application. *end guidance*]

Typically, applications shall not perform this compatibility. This element, when present with a val attribute value of `true` (or equivalent), specifies that applications shall attempt to mimic that existing word processing application in this regard.

[*Example*: Consider a WordprocessingML document with a series of tables with table styles applied.

If this compatibility setting is turned on:

```
<w:compat>
  <w:useWord2002TableStyleRules />
</w:compat>
```

Then applications should mimic the behavior of Microsoft Word 2002 when determining the formatting resulting from the use of table styles. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.64 useWord97LineBreakRules (Emulate Word 97 East Asian Line Breaking)

This element specifies that applications shall emulate the behavior of a previously existing word processing application (Microsoft Word 97) when determining the line breaking rules for East Asian text within a WordprocessingML document.

[*Guidance*: To faithfully replicate this behavior, applications must imitate the behavior of that application, which involves many possible behaviors and cannot be faithfully placed into narrative for this Office Open XML Standard. If applications wish to match this behavior, they must utilize and duplicate the output of those applications. It is recommended that applications not intentionally replicate this behavior as it was deprecated due to issues with its output, and is maintained only for compatibility with existing documents from that application. *end guidance*]

Typically, applications shall not perform this compatibility. This element, when present with a val attribute value of true (or equivalent), specifies that applications shall attempt to mimic that existing word processing application in this regard.

[*Example*: Consider a WordprocessingML document with East Asian run content.

If this compatibility setting is turned on:

```
<w:compat>
  <w:useWord97LineBreakingRules />
</w:compat>
```

Then applications should mimic the behavior of Microsoft Word 97 when determining the line breaking from the use of the East Asian characters. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.65 wpJustification (Emulate WordPerfect 6.x Paragraph Justification)

This element specifies that applications shall emulate the behavior of a previously existing word processing application (WordPerfect 6.x) when performing full paragraph justification using a val attribute value of both on the jc element (§2.3.1.13). This alternate justification method involves biasing towards compressing rather than expanding spaces when needed to justify a line.

[*Guidance*: To faithfully replicate this behavior, applications must imitate the behavior of that application, which involves many possible behaviors and cannot be faithfully placed into narrative for this Office Open XML Standard. If applications wish to match this behavior, they must utilize and duplicate the output of those applications. It is recommended that applications not intentionally replicate this behavior as it was deprecated due to issues with its output, and is maintained only for compatibility with existing documents from that application. *end guidance*]

Typically, applications shall not perform this compatibility. This element, when present with a val attribute value of true (or equivalent), specifies that applications shall attempt to mimic that existing word processing application in this regard.

[*Example*: Consider a WordprocessingML document with one or more paragraphs using full paragraph justification:

```
<w:p>
  <w:pPr>
    <w:jc w:val="both" />
  </w:pPr>
  …
</w:p>
```

If this compatibility setting is turned on:

```
<w:compat>
  <w:wpJustification />
</w:compat>
```

Then applications should mimic the behavior of WordPerfect 6.x when performing full justification on text in these paragraphs. *end example*]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.66 wpSpaceWidth (Space width)

Set the width of a space like WordPerfect 5.x.

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.15.3.67 wrapTrailSpaces (Line Wrap Trailing Spaces)

This element specifies whether applications shall perform line wrapping on trailing spaces in the contents of a line when displaying in it a paragraph. *Trailing spaces* are all space characters which are not followed by non-space characters on the same line.

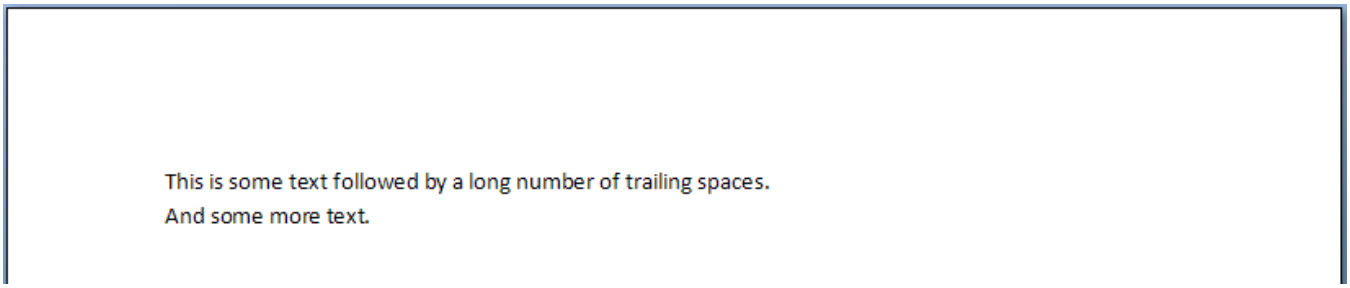
Typically, applications do not line wrap trailing spaces, instead allowing an unbounded number of trailing spaces on a line, with the next non-space character starting at the first character position on the next line. This element, when present with a val attribute value of true (or equivalent), specifies that all characters, including trailing spaces, shall be line wrapped normally.

[*Example*: Consider a WordprocessingML document with the following paragraph of text, including a long interstitial of spaces which become trailing spaces when the paragraph is displayed:

```
<w:r>
  <w:t> This is some text followed by a long number of trailing spaces.


                And some more text.</w:t>
</w:r>
```

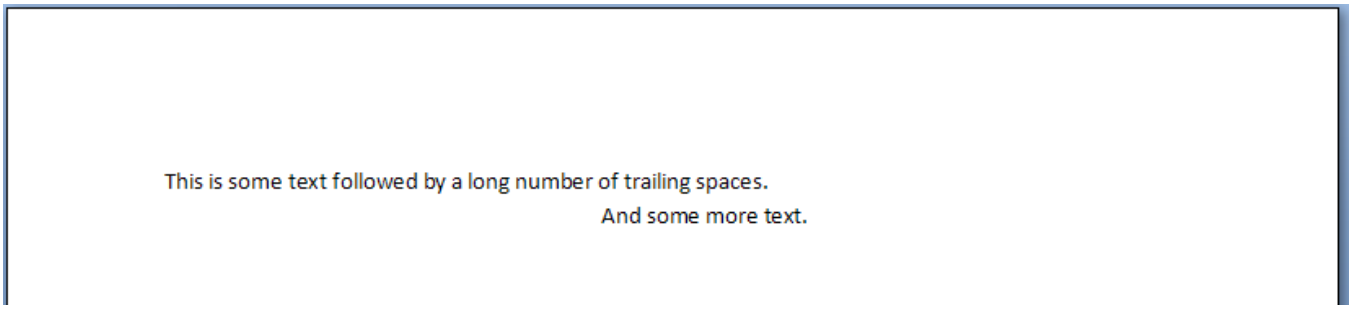
The default presentation would not wrap those trailing spaces, so the text at the end of the run would begin at the first character position on the second line:



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:wrapTrailSpaces />
</w:compat>
```

Then all trailing spaces would be handled as regular characters when line wrapping, resulting in the following output:



end example]

Parent Elements
compat (§2.15.3.9)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.16 Fields & Hyperlinks

Most text in a word processing document is static; that is, unless it is directly changed as the result of editing, its contents remain the same, no matter how the rest of the document might change. However, certain useful pieces of information can change value over the life of a document. Consider the case of a reference to a page number, as in "For more information on this topic, see page 56." Clearly, hard coding the page number as 56 means that that number will need to be manually replaced as the document's size or layout is changed. Even a simple change to any margin, line spacing, or font size can invalid such references.

Fields provide a mechanism for placeholders, such as page reference numbers, that can be added to a document such that those placeholders are replaced by their corresponding values when the document is rendered for display or print. Other applications for fields include, but are not limited to, automatic numbering of tables and figures, document creation and current date and time, document author information, and the computation of totals for a table column.

A *field* is a set of codes that instructs a WordprocessingML consumer to insert text, graphics, page numbers, and other material into a document automatically. [*Example*: The DATE field causes the current date to be inserted. *end example*] The text or graphics inserted into a document when a consumer carries out a field's codes is referred to as the *field result* for that field. The act of carrying out a field's codes is referred to as a *field update*. As to how or when any field is updated is outside the scope of this Office Open XML Standard.

2.16.1 Syntax

The general syntax of a field is as follows:

field:
field-type [*instruction*]

field-type:
date-and-time
document-automation
document-information
equations-and-formulas
index-and-tables
links-and-references
mail-merge
numbering
user-information
form-field

date-and-time:
CREATEDATE | DATE | EDITTIME | PRINTDATE | SAVEDATE | TIME

document-automation:
COMPARE | DOCVARIABLE | GOTOBUTTON | IF | MACROBUTTON | PRINT

document-information:
AUTHOR | COMMENTS | DOCPROPERTY | FILENAME | FILESIZE | INFO
| KEYWORDS | LASTSAVEDBY | NUMCHARS | NUMPAGES | NUMWORDS | SUBJECT
| TEMPLATE | TITLE

equations-and-formulas:
= *formula* | ADVANCE | EQ | SYMBOL

index-and-tables:
INDEX | RD | TA | TC | TOA | TOC | XE

links-and-references:
AUTOTEXT | AUTOTEXTLIST | BIBLIOGRAPHY | CITATION | HYPERLINK |
INCLUDEPICTURE | INCLUDETEXT
| LINK | NOTEREF | PAGEREF | QUOTE | REF | STYLEREF

mail-merge:
ADDRESSBLOCK | ASK | COMPARE | DATABASE | FILLIN | GREETINGLINE | IF
| MERGEFIELD | MERGEREC | MERGESEQ | NEXT | NEXTIF | SET | SKIPIF

numbering:
AUTONUM | AUTONUMLGL | AUTONUMOUT | BARCODE | LISTNUM | PAGE | REVNUM
| SECTION | SECTIONPAGES | SEQ

user-information:
USERADDRESS | USERINITIALS | USERNAME

form-field:
FORMCHECKBOX | FORMDROPDOWN | FORMTEXT

instruction:
field
field-argument
switches
field-argument switches
switches field-argument

field-argument:
["] *text* ["]

switches:
switch
switch switches

switch:
formatting-switch
field-specific-switch

formatting-switch:
date-and-time-formatting-switch
numeric-formatting-switch
general-formatting-switch

field-specific-switch:
field-switch-character [*field-argument*]

field-switch-character:
!
one or two Latin letters

formula is discussed in §2.16.3, and *formatting-switch*es are discussed in §2.16.4.

If the *text* in a *field-argument* contains white space, the delimiting double-quote characters shall be present; otherwise, they are optional. To include a double-quote character in *text*, it shall be preceded with a backslash (\). [*Example*: The field argument `"\"name\""` results in the argument's actually being `"name"`. *end example*] To include a backslash character in *text*, it shall be preceded with another backslash (\). [*Example*: File system pathnames on some systems use a backslash as a directory separator, as in the field

```
INCLUDETEXT "E:\\ReadMe.txt"
```

in which case, each such separator needs to be preceded with a backslash, as shown above. *end example*]

Arbitrary amount of white space can occur before the first token, after the last token, and between successive tokens, including no white space at all.

[*Example*: Here are examples of some fields:

```
DATE
DATE \@ "dddd, MMMM dd, yyyy"
DATE \@ "dddd, MMMM dd, yyyy" \h
```

The field result of all three is today's date: The first field uses some implementation-defined format and the Gregorian calendar; the second field uses the specified format and the Gregorian calendar; and the third field uses the specified format and the Hijri lunar calendar. When rendered in a US-English context on December 31, 2005, the results of these fields were as follows:

```
12/31/2005
Saturday, December 31, 2005
AsSabt, Thoul Ki'dah 30, 1426
```

end example]

Except for = *formula*, the terminals of *field-type* are alphabetic tokens [*Example*: Some field-type names are ASK, COMMENTS, NEXT, and SET. *end example*]. These tokens are called *field-type names*. Field-type names are case-insensitive. [*Example*: The field-type names DATE, Date, dAtE, and date are equivalent. *end example*]

*field-switch-character*s are case-insensitive. [*Example*: \b and \B are equivalent. *end example*]

There is no ordering of *switch* entries in *switches*.

2.16.2 XML representation

Fields shall be implemented in XML using either of two approaches:

- As a *simple field implementation*, using the fldSimple element, or
- As a *complex field implementation*, using a set of runs involving the fldChar and instrText elements.

For a simple field implementation, only one element, fldSimple, shall be used, in which case, its instr attribute shall contain a *field*, and the body of the element shall contain the most recently updated field result. [*Example*: Here is the corresponding XML for a simple field implementation of DATE:

```
<w:r>
  <w:fldSimple w:instr="DATE"> 12/31/2005 </w:fldSimple>
</w:r>
```

end example]

For a complex field implementation, a set of runs shall be used with each run containing, in sequence, the following elements:

- fldChar with attribute fldCharType value begin,
- One or more instrText elements, which, collectively, contain a complete *field*,
- Optionally,
    - fldChar with attribute fldCharType value separate, which separates the field from its field result,
    - Any number of runs and paragraphs that contains the most recently updated field result, and
- fldChar with attribute fldCharType value end.

[*Note*: Fields that are for display purposes only have no need to, and do not, store a field result. *end note*][*Example*: Here is the corresponding XML for a complex field implementation of DATE:

```
<w:r>
  <w:fldChar w:fldCharType="begin"/>
</w:r>
```

```
<w:r>
  <w:instrText xml:space="preserve"> DATE </w:instrText>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="separate"/>
</w:r>
<w:r>
  <w:t>12/31/2005</w:t>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="end"/>
</w:r>
```

end example]

[*Note*: Every simple field implementation for a given field has a corresponding complex field implementation. However, not every complex field implementation has a corresponding simple field implementation. If some characters in a *field* have different run properties than others, that field must be implemented using multiple runs, and that requires that complex field implementation be used. For an example, see §2.16.4.3, where the first letter of a DATE field is made bold, underlined, and red, while the other letters have none of these properties. *end note*]

As shown in §2.16.1, the *instruction* of one *field* can be another *field*, allowing fields to nest. In such cases, the XML run sequence for the inner field is defined at the point of reference for that inner field, inside the outer field's XML run sequence. [*Example*: Consider the following sentence:

It's `IF DATE \@ "M-d"<>"1-1" "not "` new year's day.

The `IF` field contains the nested field `DATE \@ "M-d"`. When updated, on January 1 of any year, the result sentence is "It's new year's day." On all other days of the year, the resulting sentence is "It's not new year's day."

Here is one way of writing the corresponding XML:

```
<w:r>
  <w:t xml:space="preserve">It’s </w:t>
</w:r>
<w:r …>
  <w:fldChar w:fldCharType="begin"/>
</w:r>
<w:r>
  <w:instrText xml:space="preserve">IF </w:instrText>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="begin"/>
</w:r>
```

```
<w:r>
  <w:instrText xml:space="preserve"> DATE  \@ "M-d"  </w:instrText>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="separate"/>
</w:r>
<w:r …>
  <w:instrText>1-4</w:instrText>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="end"/>
</w:r>
<w:r>
  <w:instrText>&lt;&gt;"1-1" "not "</w:instrText>
</w:r>
<w:r …>
  <w:fldChar w:fldCharType="separate"/>
</w:r>
<w:r …>
  <w:t xml:space="preserve">not </w:t>
</w:r>
<w:r …>
  <w:fldChar w:fldCharType="end"/>
</w:r>
<w:r>
  <w:t>new year’s day!</w:t>
</w:r>
```

end example]

2.16.3 Formulas and expressions

A field instruction can involve a calculation via a *formula*:

formula:
expression

where *expression* can be an arbitrary complex arithmetic expression involving constants (§2.16.3.1), bookmarks that refer to *expression*s (§2.16.3.2), arithmetic and logical operators (§2.16.3.3), functions (§2.16.3.4), values of cells in a table (§2.16.3.5), and *field*s that result in a single value. *expression* can contain grouping parentheses to document the default precedence or to override it.

All arithmetic terms in an *expression* are real numbers. Infinities and NaN (Not-a-Number) are not supported. [*Example*: In the expression `1/3`, although the operands appear to be integers, they are, in fact real numbers, and the result is `0.33`. *end example*]

2.16.3.1 Constants

A constant has the following form:

constant:
number

number:
whole-number-part [.]
. *fractional-part*
whole-number-part . *fractional-part*

whole-number-part:
series of one or more decimal digits

fractional-part:
series of one or more decimal digits

[*Example*: Here are some constants: `1234`, `1234.560`, `1234.`, and `.1234`. Exponents are not supported. *end example*]

2.16.3.2 Bookmarks

Any arbitrary piece of text and/or graphics in a WordprocessingML document can be assigned a name, called a *bookmark*. If a bookmark references text that represents an *expression*, that bookmark's name can be used as an operand in another *expression*. If a whole field is bookmarked, its bookmark name can also be used as an operand in an *expression*. [*Example*: Given that X is a bookmark for the text 4, Y is a bookmark for the text 2, and `Result` is a bookmark for the following field:

```
=X + Y
```

the field

```
=Result * 10
```

has the result `60`. *end example*]

2.16.3.3 Operators

The *operator*s permitted in *expression* are:

Operators		
Operator	Description	Precedence
-	Unary minus	highest
^	Powers and roots	
*	Multiplication	
/	Division	
%	Percentage	
+	Addition	

Operators		
-	Subtraction	
=	Equal to	
<>	Not equal to	
<	Less than	lowest
<=	Less than or equal to	
>	Greater than	
>=	Greater than or equal to	

Operators in *expression* having the same precedence associate left-to-right.

[*Example*: Given that X is a bookmark for the text 4, and Y is a bookmark for the text 2, the field

```
=((-1 + X^2) * 3 - Y)/2
```

produces the result `21.5`. *end example*]

The equality, inequality, and relational operators yield 1 for true and 0 for false. An expression with value 0 tests logically false while one with any non-zero value tests true.

2.16.3.4 Functions

A *function* is a predefined procedure that computes and returns a result. Functions defined below with a parameter list of *list* accept two or more arguments separated by commas (,) or semicolons (;). As to which separator is permitted, is defined by the document's listSeparator (§2.15.1.56) element. Arguments to functions can be constants, formulas, or bookmark names that refer to constants or formulas. The functions AVERAGE, COUNT, MAX, MIN, PRODUCT, and SUM can also accept references to table cells as arguments. In the context of a table cell, functions taking a *list* also accept a single argument that designates a named-list of contiguous cells (§2.16.3.5). Function names are not case-sensitive, and white space can occur between a function's name and its argument list, if any.

The functions supported are as follows:

Functions	
Function	Description
ABS(*x*)	Returns the absolute value of *x*.
AND(*x*,*y*)	Returns 1 if the logical expressions *x* and *y* are both true; otherwise, it returns 0.
AVERAGE(*list*)	Returns the average value of the items in *list*.
COUNT(*list*)	Returns the number of items in *list*.
DEFINED(*x*)	Returns 1 if the expression *x* is well formed; otherwise, it returns 0.

Functions	
FALSE	Returns 0.
INT(*x*)	Returns the value of the integer part of *x*.
MAX(*list*)	Returns the largest value in *list*.
MIN(*list*)	Returns the smallest value in *list*.
MOD(*x*,*y*)	Returns the value *x* - *ny*, for some integer *n* such that, if *y* is nonzero, the result has the same sign as *x* and magnitude less than the magnitude of *y*. If *y* is zero, a diagnostic shall be issued. (*y* need not be a whole number.) [*Example*: MOD(21,5) results in 1 MOD(21,-5) results in 1 MOD(-21,5) results in -1 MOD(-21,-5) results in -1 *end example*]
NOT(*x*)	Returns 0 if the logical expression *x* is true, or 1 if the expression is false.
OR(*x*,*y*)	Returns 1 if either or both logical expressions *x* and *y* are true; otherwise, it returns 0.
PRODUCT(*list*)	Returns the result of multiplying together all members in *list*.
ROUND(*x*,*y*)	Returns the value of *x* rounded to the specified number of decimal places indicated by floor(*y*), where floor has the mathematical meaning. If *y* is negative, any fractional part is discarded and the integer part of the value is rounded to the corresponding power of 10.
SIGN(*x*)	Returns 1 if *x* is positive; returns 0 if *x* is zero; and returns –1 if *x* is negative.
SUM(*list*)	Returns the sum of the items in *list*.
TRUE	Returns 1.

2.16.3.5 Table cell references

Items in a WordprocessingML table are organized into rows and columns with the box formed by the intersection of a row and column being called a *cell*. Cells have names such as A1, A2, B1, B2, and so on, with the letter representing a column and the number representing a row. The cell at the top-left corner of each table is named A1. Column letters are not case-sensitive.

A *cell reference* shall be one of the following:

- The name of a cell.
- A comma-separated set of cell names.
- A cell range where a colon (:) is used to separate the first and last cells in a designated range of cells that has a contiguous rectangular shape. Specifying a row or column's name only as the first and last cell in a

range, selects that whole row or column, regardless of the number of rows and columns the table has now or might have in the future.

An *expression* inside a table's cell can have operands that are references to other cells in that table.

[*Example*: Consider a table with three rows (1, 2, and 3) and two columns (A and B):

`A1 + B1`	Returns the sum of the contents of cells A1 and B1.
`SUM(A1,B2,A3)`	Returns the sum of the contents of the list of cells.
`SUM(B1:B3)`	Returns the sum of the contents of all cells between B1 and B3, inclusive.
`SUM(B:B)`	Returns the sum of the contents of all cells in column B (even if new rows are added later).
`SUM(A1:B2)`	Returns the sum of the contents of all (four) cells in the rectangular grid delimited by A1 and B2, inclusive.
`SUM(1:1,2:2)`	Returns the sum of the contents of all cells in rows 1 and 2.

end example]

When used in a table cell, the functions taking a *list* argument can have a single argument of `ABOVE`, `BELOW`, `LEFT`, or `RIGHT`, spelled in any case combination. Such lists designate, respectively, all the cells above, below, to the left of, or to the right of that cell. However, the designated range terminates if a cell with blank or non-numeric contents is reached, except that if the first cell is blank, it is treated as containing 0. [*Example*: Given the following table:

`12`	`=COUNT(BELOW)`	
	10	
2	20	`=SUM(LEFT)`
3	xxx	
`=AVERAGE(ABOVE)`	40	

`AVERAGE(ABOVE)` results in 2.5, the average of cells A4 and A3; `COUNT(BELOW)` results in 2, B2 and B3; and `SUM(LEFT)` results in 22, the sum of B3 and A3. *end example*]

An *expression* used outside a table or in a cell of one table can refer to cells in a second table by making a bookmark to that second table and qualifying cell names in that table by their table name using the form

(*tableBookmarkName cellReference*)

[*Example*: Given that `Table1` is a bookmark for a 3x2 table, `=SUM(Table1 A1:A3)` book results in the sum of column A's cells. *end example*]

2.16.4 Field formatting

The result of a field has a *format*, either by default or because that field contains a *formatting-switch*. There are three kinds of field formatting: date and time (§2.16.4.1), numeric (§2.16.4.2), and general (§2.16.4.3).

2.16.4.1 Date and time formatting

date-and-time-formatting-switch:
\@ [''] *switch-argument* ['']

A *date-and-time-formatting-switch* specifies the format of a date or time result. [*Note*: This switch is sometimes called a *picture* switch because it allows the use of symbols to represent the format of the field result. *end note*] If the result of a field is not a date or time, this switch has no effect.

Quotation marks are required around *switch-argument* if it contains white space; otherwise, they are optional.

If no *date-and-time-formatting-switch* is present, a date or time result is formatted in an implementation-defined manner.

A date and time *switch-argument* is made up of a series of *picture items*.

Date and Time Formatting Picture Items	
Picture Item	Description
d	Formats the day of(the week or day of the month as a number without a leading 0 for single-digit days.
dd	Formats the day of the week or day of the month as a number with a leading 0 for single-digit days.
ddd	Formats the day of the week or month in its abbreviated form according to the language specified by the lang element (§2.3.2.18) on the run containing the field instructions.
dddd	Formats the day of the week as its full name according to the language specified by the lang element (§2.3.2.18) on the run containing the field instructions.
M	Formats the month as a number without a leading 0 for single-digit months.
MM	Formats the month as a number with a leading 0 for single-digit months.
MMM	Formats the month in its abbreviated form according to the language specified by the lang element (§2.3.2.18) on the run containing the field instructions.
MMMM	Formats the month as its full name according to the language specified by the lang element (§2.3.2.18) on the run containing the field instructions.
yy	Formats the year as two digits with a leading 0 for years 0–9.
yyyy	Formats the year as four digits.

In the following time formats, a lowercase h indicates that time is based on a 12-hour clock, while uppercase H indicates time is based on a 24-hour clock.

Time Formatting Picture Items	
Picture Item	Description
h or H	Formats the hour without a leading 0 for single-digit hours.
hh or HH	Formats the hour with a leading 0 for single-digit hours.
m	Formats the minutes without a leading 0 for single-digit minutes.
mm	Formats the minutes with a leading 0 for single-digit minutes.
am/pm or AM/PM	Formats using an am/AM or pm/PM suffix.

Miscellaneous Formatting Picture Items	
Picture Item	Description
Other character	Includes the specified character in the result at that position. [*Note*: Commonly used characters are colon (:), hyphen (-), asterisk (*), slash (/), and space. *end note*]
'*text*'	Includes *text* in the result.
`*numbered-item*`	Includes, in Arabic numerals, the number of the preceding item numbered as a caption or resulting from a SEQ field (§2.16.5.63). *numbered-item* shall be the same name as *identifier* in that SEQ field.

[*Example*: When updated in a US-English context on the date and time shown below, the following fields produced these results:

DATE \@ "M/d/yyyy"	1/3/2006
DATE \@ "dddd, MMMM dd, yyyy"	Tuesday, January 03, 2006
DATE \@ "MMMM d, yyyy"	January 3, 2006
DATE \@ "M/d/yy"	1/3/06
DATE \@ "yyyy-MM-dd"	2006-01-03
DATE \@ "d-MMM-yy"	3-Jan-06
DATE \@ "M.d.yyyy"	1.3.2006
DATE \@ "MMM. d, yy"	Jan. 3, 06
DATE \@ "d MMMM yyyy"	3 January 2006

DATE \@ "MMMM yy"	January 06
DATE \@ "MMM-yy"	Jan-06
DATE \@ "M/d/yyyy h:mm am/pm"	1/3/2006 5:28 PM
DATE \@ "M/d/yyyy h:mm:ss am/pm"	1/3/2006 5:28:34 PM
DATE \@ "h:mm am/pm"	5:28 PM
DATE \@ "h:mm:ss am/pm"	5:28:34 PM
DATE \@ "HH:mm"	17:28
DATE \@ "'Today is 'HH:mm:ss"	Today is 17:28:34

end example]

2.16.4.2 Numeric formatting

numeric-formatting-switch:
\# ["] *switch-argument* ["]

A *numeric-formatting-switch* specifies the format of a numeric result. If the result of a field is not a number, this switch has no effect.

Quotation marks are required around *switch-argument* if it contains white space; otherwise, they are optional.

If no *numeric-formatting-switch* is present, a numeric result is formatted without leading spaces or trailing fractional zeros. If the result is negative, a leading minus sign is present. If the result is a whole number, no radix point is present.

A numeric *switch-argument* is made up of a series of *picture items*.

Numeric Formatting Picture Items	
Picture Item	Description
0	Specifies the requisite numeric positions to display in the result. If the result does not include a digit in that position, 0 is displayed. [*Example*: In a US-English context, =4+5 \# 00.00 displays "09.00". *end example*]
#	Specifies the requisite numeric positions to display in the result. If the result does not include a digit in that position, a space is displayed. Extra fractional digits are rounded off. [*Example*: =9+6 \# $### displays "$ 15". *end example*]
x	Drops digits to the left of the x placeholder. If the placeholder is to the right of the decimal point, the result is rounded to that place. [*Example*: In a US-English context, =111053+111439 \# x## displays "492", =1/8 \# 0.00x displays "0.125", and =3/4 \# .x displays ".8". *end example*]
.	Indicates the radix-point position. [*Example*: In a US-English context, =95.4 \# $###.00 displays "$ 95.40. *end example*] The radix-point

Numeric Formatting Picture Items	
	character displayed is locale-specific.
,	Separates groups of three digits. [*Example*: In a US-English context, =2456800 \# $#,###,### displays "2,456,800". *end example*] The separator character displayed is locale-specific.
-	Prepends a minus sign to a negative result, or prepends a space if the result is positive or 0. [*Example*: =80-90 \# -## displays "-10", while =90-80 \# -## displays " 80". *end example*]
+	Prepends a plus sign to a positive result, a minus sign to a negative result, or a space if the result is 0. [*Example*: =90-80 \# +## displays "+10", and =80-90 \# +## displays "-10". *end example*]
Other character	Includes the specified character in the result at that position. [*Example*: =33 \# ##% displays "33%". *end example*]
'*text*'	Includes text in the result. [*Example*: In a US-English context, if Price is a bookmark for 26.5, =Price*15% \# "##0.00 'is the sales tax'" displays "$ 3.98 is the sales tax". *end example*]
`*numbered-item*`	Includes, in Arabic numerals, the number of the preceding item numbered as a caption or resulting from a SEQ field (§2.16.5.63). *numbered-item* shall be the same name as *identifier* in that SEQ field. [*Example*: =SUM(A1:D4) \# "##0.00 'is the total of Table' `table`" displays "456.34 is the total of Table 2". *end example*]
positive-result ; *negative-result*	Specifies different sets of picture items for positive and negative results. A zero value uses the positive picture. [*Example*: =Sales95 \# $#,##0.00;-$#,##0.00 displays that bookmark's positive values using $#,##0.00, and it's negative values using -$#,##0.00. *end example*]
positive-result ; *negative-result* ; *zero-result*	Specifies different sets of picture items for positive, negative, and zero results. [*Example*: =Sales95 \# $#,##0.00;-$#,##0.00;$0 displays that bookmark's positive values using $#,##0.00, it's negative values using -$#,##0.00, and its zero values using $0. *end example*]

2.16.4.3 General formatting

general-formatting-switch:
* ["] *switch-argument* ["]

A *general-formatting-switch* specifies a variety of formats for a numeric or text result. If the result type of a field does not correspond to the format specified, this switch has no effect.

Quotation marks are required around *switch-argument* if it contains white space; otherwise, they are optional.

A *switch-argument* is made up of a series of *picture items*.

General Formatting Switch Arguments	
Switch Argument	Description
AIUEO	Formats a numeric result using hiragana characters in the traditional a-i-u-e-o

General Formatting Switch Arguments	
	order. [*Example*: `1 \* AIUEO` results in ア. *end example*] Corresponds to an ST_NumberFormat enumeration value of `aiueo`.
`ALPHABETIC`	Formats a numeric result as one or more occurrences of an uppercase alphabetic Latin character. Value 1 results in the letter A, value 2 results in the letter B, and so on up to value 26, which results in the letter Z. For values greater than 26, 26 is repeatedly subtracted from the value until the result is 26 or less. The result value determines which letter to use, and the same letter is repeated for each time 26 was subtracted from the original value. [*Example*: `=54 \* ALPHABETIC` results in "BBB" as subtracting 26 from 54 two times, results in the value 2, which is represented by the letter B. *end example*]
`alphabetic`	Formats a numeric result as one or more occurrences of an lowercase alphabetic Latin character. Value 1 results in the letter a, value 2 results in the letter b, and so on up to value 26, which results in the letter z. For values greater than 26, 26 is repeatedly subtracted from the value until the result is 26 or less. The result value determines which letter to use, and the same letter is repeated for each time 26 was subtracted from the original value. [*Example*: `=52 \* alphabetic` results in "zz" as subtracting 26 from 52 one time, results in the value 26, which is represented by the letter z.. *end example*]
`Arabic`	Formats a numeric result using Arabic cardinal numerals. [*Example*: For page 123, `PAGE \* Arabic` results in "١٢٣". *end example*] Corresponds to an ST_NumberFormat enumeration value of `decimal`.
`ARABICABJAD`	Formats a numeric result using ascending Abjad numerals. [*Example*: `12 \* ARABICABJAD` results in ك. *end example*] Corresponds to an ST_NumberFormat enumeration value of `arabicAbjad`.
`ARABICALPHA`	Formats a numeric result using characters in the Arabic alphabet. [*Example*: `12 \* ARABICABJAD` results in س. *end example*] Corresponds to an ST_NumberFormat enumeration value of `arabicAlpha`.
`ArabicDash`	Formats a numeric result using Arabic cardinal numerals, with a prefix of "- " and a suffix of " -". [*Example*: For page 123, `PAGE \* ArabicDash` results in "- ١٢٣ -". *end example*] Corresponds to an ST_NumberFormat enumeration value of `numberInDash`.
`BAHTTEXT`	Formats a numeric result using the given Thai style. [*Example*: `1 \* BAHTTEXT` results in หนึ่งบาทถ้วน. *end example*]
`Caps`	Capitalizes the first letter of each word. [*Example*: `USERNAME "mary smith" \* Caps` results in "Mary Smith", whereas `USERNAME "marysmith" \* Caps` results in "Marysmith". *end example*]
`CardText`	Formats a numeric result as lowercase cardinal text. [*Example*: For page 123, `PAGE \* CardText` results in "one hundred twenty-three". *end example*] Corresponds to an ST_NumberFormat enumeration value of `cardinalText`.
`CHARFORMAT`	See the discussion following this table.
`CHINESENUM1`	Formats a numeric result using ascending numbers from the Chinese counting system. [*Example*: `10 \* CHINESENUM1` results in 十. *end example*]

General Formatting Switch Arguments	
	Corresponds to an ST_NumberFormat enumeration value of chineseCounting.
CHINESENUM2	Formats a numeric result using sequential numbers from the Chinese simplified legal format. [*Example*: 123 * CHINESENUM2 results in 壹佰贰拾叁. *end example*] Corresponds to an ST_NumberFormat enumeration value of chineseLegalSimplified.
CHINESENUM3	Formats a numeric result using sequential numbers from the Chinese counting thousand system. [*Example*: 10 * CHINESENUM3 results in 一百二十三. *end example*] Corresponds to an ST_NumberFormat enumeration value of chineseCountingThousand.
CHOSUNG	Formats a numeric result using sequential numbers from the Korean Chosung format. [*Example*: 1 * CHOSUNG results in ㄱ. *end example*] Corresponds to an ST_NumberFormat enumeration value of chosung.
CIRCLENUM	Formats a numeric result using decimal numbering enclosed in a circle, using the enclosed alphanumeric glyph character for numbers in the range 1–20. For non-negative numbers outside this range, formats them as with ARABIC. [*Example*: 12 * CIRCLENUM results in ⑫. *end example*] Corresponds to an ST_NumberFormat enumeration value of decimalEnclosedCircle.
DBCHAR	Formats a numeric result using double-byte Arabic numbering. [*Example*: 123 * DBCHAR results in １２３. *end example*] Corresponds to an ST_NumberFormat enumeration value of decimalFullWidth.
DBNUM1	Formats a numeric result using sequential digital ideographs, using the appropriate character. [*Example*: 12 * DBNUM1 results in 一二. *end example*] Corresponds to an ST_NumberFormat enumeration value of ideographDigital.
DBNUM2	Formats a numeric result using sequential numbers from the Korean counting system. [*Example*: 12 * DBNUM2 results in 십이. *end example*] Corresponds to an ST_NumberFormat enumeration value of koreanCounting.
DBNUM3	Formats a numeric result using sequential numbers from the Japanese legal counting system. [*Example*: 12 * DBNUM3 results in 壱拾弐. *end example*] Corresponds to an ST_NumberFormat enumeration value of japaneseLegal.
DBNUM4	Formats a numeric result using sequential numbers from the Japanese digital ten thousand counting system. [*Example*: 12 * DBNUM4 results in 一二. *end example*] Corresponds to an ST_NumberFormat enumeration value of japaneseDigitalTenThousand.
DollarText	Formats a numeric result in the following form: *integer-part-as-cardinal-text* and *nn*/100 The fractional part is rounded to two decimal places, *nn*, and is formatted using

General Formatting Switch Arguments	
	Arabic cardinal numerals. [*Example*: `=1234.567 \* DollarText` results in "`one thousand two hundred thirty-four and 57/100`". *end example*]
`FirstCap`	Capitalizes the first letter of the first word. [*Example*: `USERNAME "mary smith" \* FirstCap` results in "`Mary smith`". *end example*]
`GANADA`	Formats a numeric result using sequential numbers from the Korean Ganada format. [*Example*: `12 \* GANADA` results in 타. *end example*] Corresponds to an ST_NumberFormat enumeration value of `ganada`.
`GB1`	Formats a numeric result using decimal numbering followed by a period, using the enclosed alphanumeric glyph character. [*Example*: `12 \* GB1` results in 12.. *end example*] Corresponds to an ST_NumberFormat enumeration value of `decimalEnclosedFullstop`.
`GB2`	Formats a numeric result using decimal numbering enclosed in parenthesis, using the enclosed alphanumeric glyph character. [*Example*: `12 \* GB2` results in ⑿. *end example*] Corresponds to an ST_NumberFormat enumeration value of `decimalEnclosedParen`.
`GB3`	Formats a numeric result using decimal numbering enclosed in a circle, using the enclosed alphanumeric glyph character. Once the specified sequence reaches 11, the numbers may be replaced with non-enclosed equivalents. [*Example*: `12 \* GB3` results in 12. *end example*] Corresponds to an ST_NumberFormat enumeration value of `decimalEnclosedCircleChinese`.
`GB4`	Formats a numeric result using decimal numbering enclosed in a circle, using the enclosed alphanumeric glyph character. Once the specified sequence reaches 11, the numbers may be replaced with non-enclosed equivalents. [*Example*: `12 \* GB4` results in 12. *end example*] Corresponds to an ST_NumberFormat enumeration value of `ideographEnclosedCircle`.
`HEBREW1`	Formats a numeric result using Hebrew numerals. [*Example*: `123 \* HEBREW1` results in קכג. *end example*] Corresponds to an ST_NumberFormat enumeration value of `hebrew1`.
`HEBREW2`	Formats a numeric result using the Hebrew alphabet. [*Example*: `123 \* HEBREW2` results in תתתתתמ. *end example*] Corresponds to an ST_NumberFormat enumeration value of `hebrew2`.
`Hex`	Formats the numeric result using uppercase hexadecimal digits. [*Example*: For page 355, `PAGE \* Hex` results in "`FF`". *end example*] Corresponds to an ST_NumberFormat enumeration value of `hex`.
`HINDIARABIC`	Formats a numeric result using Hindi numbers. [*Example*: `123 \* HINDIARABIC` results in ١٢٣. *end example*] Corresponds to an ST_NumberFormat enumeration value of `hindiNumbers`.
`HINDICARDTEXT`	Formats a numeric result using sequential numbers from the Hindi counting

General Formatting Switch Arguments	
	system. [*Example*: `123 \* HINDICARDTEXT` results in एक सौ तेईस. *end example*] Corresponds to an ST_NumberFormat enumeration value of `hindiCounting`.
`HINDILETTER1`	Formats a numeric result using Hindi vowels. [*Example*: `123 \* HINDILETTER1` results in ठठठठ. *end example*] Corresponds to an ST_NumberFormat enumeration value of `hindiVowels`.
`HINDILETTER2`	Formats a numeric result using Hindi consonants. [*Example*: `123 \* HINDILETTER2` results in ओओओओओओओ. *end example*] Corresponds to an ST_NumberFormat enumeration value of `hindiConsonants`.
`IROHA`	Formats a numeric result using the Japanese iroha. [*Example*: `12 \* IROHA` results in 才. *end example*] Corresponds to an ST_NumberFormat enumeration value of `iroha`.
`KANJINUM1`	Formats a numeric result using a Japanese style using sequential digital ideographs, using the appropriate character. [*Example*: `12 \* KANJINUM1` results in 一二. *end example*]
`KANJINUM2`	Formats a numeric result using the Japanese counting system. [*Example*: `12 \* KANJINUM2` results in 十二. *end example*] Corresponds to an ST_NumberFormat enumeration value of `japaneseCounting`.
`KANJINUM3`	Formats a numeric result using the Japanese legal counting system. [*Example*: `12 \* KANJINUM3` results in 壱拾弐. *end example*]
`Lower`	All letters are lowercase. [*Example*: `USERNAME "Mary Smith" \* Lower` results in "`mary smith`". *end example*]
`MERGEFORMAT`	See the discussion following this table.
`Ordinal`	Formats a numeric result using lowercase ordinal Arabic numerals. [*Example*: `=32 \* Ordinal` results in "`32nd`". *end example*] Corresponds to an ST_NumberFormat enumeration value of `ordinal`.
`OrdText`	Formats a numeric result as lowercase ordinal text. Apart from being used to round off the whole number part, the fractional part is not used. [*Example*: `=1234.567 \* OrdText` results in "`one thousand two hundred thirty-fifth`". *end example*] Corresponds to an ST_NumberFormat enumeration value of `ordinalText`.
`Roman`	Formats a numeric result using uppercase Roman numerals. [*Example*: For page 123, `PAGE \* Roman` results in "`CXXIII`". *end example*] Corresponds to an ST_NumberFormat enumeration value of `upperRoman`.
`roman`	Formats a numeric result using lowercase Roman numerals. [*Example*: For page 123, `PAGE \* roman` results in "`cxxiii`". *end example*] Corresponds to an ST_NumberFormat enumeration value of `lowerRoman`.
`SBCHAR`	Formats a numeric result using single-byte Arabic numbering. [*Example*: `123 \* SBCHAR` results in 123. *end example*]

General Formatting Switch Arguments	
	Corresponds to an ST_NumberFormat enumeration value of `decimalHalfWidth`.
THAIARABIC	Formats a numeric result using Thai numbers. [*Example*: `123 \* THAIARABIC` results in ๑๒๓. *end example*] Corresponds to an ST_NumberFormat enumeration value of `thaiNumbers`.
THAICARDTEXT	Formats a numeric result using sequential numbers from the Thai counting system. [*Example*: `123 \* THAICARDTEXT` results in หนึ่งร้อยยี่สิบสาม. *end example*] Corresponds to an ST_NumberFormat enumeration value of `thaiCounting`.
THAILETTER	Formats a numeric result using Thai letters. [*Example*: `123 \* THAILETTER` results in ฮฮฮ. *end example*] Corresponds to an ST_NumberFormat enumeration value of `thaiLetters`.
Upper	All letters are uppercase. [*Example*: `USERNAME "Mary Smith" \* Upper` results in "MARY SMITH". *end example*]
VIETCARDTEXT	Formats a numeric result using Vietnamese numerals. [*Example*: `12 \* VIETCARDTEXT` results in mười hai. *end example*] Corresponds to an ST_NumberFormat enumeration value of `vietnameseCounting`.
ZODIAC1	Formats a numeric result using sequential numerical traditional ideographs. [*Example*: `1 \* ZODIAC1` results in 甲. *end example*] Corresponds to an ST_NumberFormat enumeration value of `ideographTraditional`.
ZODIAC2	Formats a numeric result using sequential zodiac ideographs. [*Example*: `1 \* ZODIAC2` results in 子. *end example*] Corresponds to an ST_NumberFormat enumeration value of `ideographZodiac`.
ZODIAC3	Formats a numeric result using sequential traditional zodiac ideographs. [*Example*: `1 \* ZODIAC3` results in 甲子. *end example*] Corresponds to an ST_NumberFormat enumeration value of `ideographZodiacTraditional`.

The general formatting switch argument CHARFORMAT is used to set the visual appearance of a field's value by setting the first run in that field's *field-type* name to the desired state using any of the elements that can be directly nested inside the run property element, rPr. [*Example*: In a US-English context, on January 4, 2006, the field DATE * CHARFORMAT results in "1/4/2006". However, if the D in DATE is made bold, the field **D**ATE * CHARFORMAT results in "**1/4/2006**". If that D is made italic, the field *D*ATE * CHARFORMAT results in "*1/4/2006*". If that D is made bold, underlined, and red, the field **D**ATE * CHARFORMAT results in "**1/4/2006**".

The XML for the bold, underlined, red case is as follows:

```
<w:r>
  <w:fldChar w:fldCharType="begin"/>
</w:r>
<w:r>
  <w:instrText xml:space="preserve"> </w:instrText>
</w:r>
<w:r …>
  <w:rPr>
    <w:b/>
    <w:color w:val="ED1C24"/>
    <w:u w:val="single"/>
  </w:rPr>
  <w:instrText>D</w:instrText>
</w:r>
<w:r>
  <w:instrText xml:space="preserve">ATE </w:instrText>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="separate"/>
</w:r>
<w:r …>
  <w:rPr>
    <w:b/>
    <w:color w:val="ED1C24"/>
    <w:u w:val="single"/>
  </w:rPr>
  <w:t>1/4/2006</w:t>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="end"/>
</w:r>
```

end example]

If a format specified directly in the first run of a field's *field-type* name conflicts with a general formatting switch, the general formatting switch is ignored. [*Example*: If the first run is set in small caps and the switch `\* Lower` is also used, that switch is ignored. *end example*]

The general formatting switch argument `MERGEFORMAT` is used to apply formatting directly to part of a result such that when that result is updated, the formatting is preserved. The formatting is expressed in XML using an `rPr` element on the run that contains the most recently updated field result. [*Example*: Consider the following field:

```
TIME \@ "HH:mm:ss" \* MERGEFORMAT
```

When it is updated, the result might be 12:22:27, for example. If the seconds part of the displayed field result is underlined, as in 12:22:27, when that field is next updated, the seconds underlining is preserved. If MERGEFORMAT is omitted, the rPr element on the run that contains the most recently updated field result is ignored.

The XML generated for this is:

```
<w:r>
  <w:fldChar w:fldCharType="begin"/>
</w:r>
<w:r>
  <w:instrText xml:space="preserve"> TIME \@ "HH:mm:ss" \* MERGEFORMAT
</w:instrText>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="separate"/>
</w:r>
<w:r …>
  <w:t>17:02:</w:t>
</w:r>
<w:r …>
  <w:rPr>
    <w:u w:val="single"/>
  </w:rPr>
  <w:t>32</w:t>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="end"/>
</w:r>
```

end example]

2.16.5 Field definitions

The set of fields is divided into the following functional categories:

Category	Description	Fields
Date and Time	Inserts the current date and/or time, or date and/or time of some kind of event.	CREATEDATE (§2.16.5.16), DATE (§2.16.5.18), EDITTIME (§2.16.5.21), PRINTDATE (§2.16.5.54), SAVEDATE (§2.16.5.60), TIME (§2.16.5.72)
Document Automation	Compares values and takes action based on outcome, run macros, and sends a code to a printer.	COMPARE (§2.16.5.15), DOCVARIABLE (§2.16.5.20), GOTOBUTTON (§2.16.5.29), IF (§2.16.5.32), MACROBUTTON (§2.16.5.41), PRINT (§2.16.5.53)

Category	Description	Fields
Document Information	Inserts or stores information about the document.	AUTHOR (§2.16.5.4), COMMENTS (§2.16.5.14), DOCPROPERTY (§2.16.5.19), FILENAME (§2.16.5.23), FILESIZE (§2.16.5.24), INFO (§2.16.5.36), KEYWORDS (§2.16.5.37), LASTSAVEDBY (§2.16.5.38), NUMCHARS (§2.16.5.48), NUMPAGES (§2.16.5.49), NUMWORDS (§2.16.5.50), SUBJECT (§2.16.5.67), TEMPLATE (§2.16.5.71), TITLE (§2.16.5.73)
Equations and Formulas	Defines formulas and calculates results; inserts symbols.	= *formula* (§2.16.3), ADVANCE (§2.16.5.2), EQ (§2.16.5.22), SYMBOL (§2.16.5.68)
Form Fields	Allows the insertion of form fields.	FORMCHECKBOX (§2.16.5.26), FORMDROPDOWN (§2.16.5.27), FORMTEXT (§2.16.5.28)
Index and Tables	Defines entries for, and builds, a table of contents, table of figures, and table of authorities.	INDEX (§2.16.5.35), RD (§2.16.5.57), TA (§2.16.5.69), TC (§2.16.5.70), TOA (§2.16.5.74), TOC (§2.16.5.75), XE (§2.16.5.79)
Links and References	Inserts information from another place in the same document, from a different document or file, or from an AutoText entry.	AUTOTEXT (§2.16.5.8), AUTOTEXTLIST (§2.16.5.9), BIBLIOGRAPHY (§2.16.5.11), CITATION (§2.16.5.13) HYPERLINK (§2.16.5.31), INCLUDEPICTURE (§2.16.5.33), INCLUDETEXT (§2.16.5.34), LINK (§2.16.5.39), NOTEREF (§2.16.5.47), PAGEREF (§2.16.5.52), QUOTE (§2.16.5.56), REF (§2.16.5.58), STYLEREF (§2.16.5.66)
Mail Merge	Defines information that is to be used in a mail merge.	ADDRESSBLOCK (§2.16.5.1), ASK (§2.16.5.3), COMPARE (§2.16.5.15), DATABASE (§2.16.5.17), FILLIN (§2.16.5.25), GREETINGLINE (§2.16.5.30), IF (§2.16.5.32), MERGEFIELD (§2.16.5.42), MERGEREC (§2.16.5.43), MERGESEQ (§2.16.5.44), NEXT (§2.16.5.45), NEXTIF (§2.16.5.46), SET (§2.16.5.64), SKIPIF (§2.16.5.65)
Numbering	Specifies numbering for document items such as sections and pages; also bar codes.	AUTONUM (§2.16.5.5), AUTONUMLGL (§2.16.5.6), AUTONUMOUT (§2.16.5.7), BARCODE (§2.16.5.10), LISTNUM (§2.16.5.40), PAGE (§2.16.5.51), REVNUM (§2.16.5.59), SECTION (§2.16.5.61), SECTIONPAGES (§2.16.5.62), SEQ (§2.16.5.63)

Category	Description	Fields
User Information	Stores or inserts the name, initials, or address of the document user.	USERADDRESS (§2.16.5.76), USERINITIALS (§2.16.5.77), USERNAME (§2.16.5.78)

2.16.5.1 ADDRESSBLOCK

Syntax:

ADDRESSBLOCK [*switches*]

Description: Inserts a mail merge address block.

Field Value: The address block.

Switches: Zero or more of the following *field-specific-switch*es.

\c *field-argument*	*text* in this switch's *field-argument* specifies whether to include the name of the country/region: a value of 0 causes the country/region to be omitted; a value of 1 causes it to be included, and a value of 2 causes country/region to be included, but only if it is different from the value for \e.
\d	Specifies that the address is to be formatted according to the country/region of the recipient. If this switch is not used, then addresses are formatted according to some implementation-specific preference.
\e *field-argument*	*text* in this switch's *field-argument* specifies which country/region to exclude from the address block. [*Note*: This is useful when your mailing contains a mix of domestic and international recipients. *end note*] To exclude the names of more than one country or region, use a \e switch for each one.
\f *field-argument*	*text* in this switch's *field-argument* specifies the name and address format by providing a template of merge-field placeholders.
\l *field-argument*	*text* in this switch's *field-argument* specifies the language ID used to format the address. The default is to use the language ID of the first character of the document.

2.16.5.2 ADVANCE

Syntax:

ADVANCE [*switches*]

Description: Moves the starting point of text that follows the field to the right or left, up or down, or to a specific horizontal or vertical position. The switches used by this field can cause text to overlap. Text will not display if it is moved to the previous or next page, or beyond the print margins of the current page.

Field Value: None.

Switches: Zero or more of the following *field-specific-switch*es.

\d *field-argument*	Moves the text that follows the field down by the integral number of points specified by *text* in this switch's *field-argument*.
\l *field-argument*	Moves the text that follows the field left by the integral number of points specified by *text* in this switch's *field-argument*.
\r *field-argument*	Moves the text that follows the field right by the integral number of points specified by *text* in this switch's *field-argument*.
\u *field-argument*	Moves the text that follows the field up by the integral number of points specified by *text* in this switch's *field-argument*.
\x *field-argument*	Moves the text that follows the field the integral number of points specified by *text* in this switch's *field-argument* from the left edge of the column, frame, or text box.
\y *field-argument*	Moves the text that follows the field the integral number of points specified by *text* in this switch's *field-argument*. This shift is the vertical position relative to the page. The entire line of text that contains the field is moved. This switch is ignored if it specifies a location outside the page margins or if the switch is used inside any of the following: table, text box, footnote, endnote, annotation, header, or footer.

[*Example*: When the following fields are updated:

```
XX ADVANCE \u 6 XX ADVANCE \d 12 XX ADVANCE \l 20 + ADVANCE \x 150 ZZ
```

The results are:

```
XXXX
  XX+ X                ZZ
```

end example]

2.16.5.3 ASK

Syntax:

ASK *field-argument-1* *field-argument-2* [*switches*]

field-argument-1:
field-argument

field-argument-2:
field-argument

Description: Prompts the user to enter information and assigns the bookmark designated by *field-argument-1* to represent the user's response. *text* in *field-argument-2* specifies the prompt text, which is displayed in a dialog box. The prompt is displayed each time the ASK field is updated. A response remains assigned to the bookmark until a new response is entered.

Field Value: None.

Switches: Zero or more of the following *field-specific-switch*es.

`\d` *field-argument*	The *text* in this switch's *field-argument* specifies a default response if one is not entered. If no default response is specified, the most recent response is used. To specify a blank entry as the default, *field-argument* shall be `""`.
`\o`	When used in a mail merge main document, this causes the display of the prompt once instead of each time a new data record is merged. The same response is inserted in each resulting merged document.

[*Example*: When the following fields are updated and "John" is entered as the response,

```
ASK AskResponse "What is your first name?"
Hello REF AskResponse.
```

the result is

```
Hello John.
```

end example]

2.16.5.4 AUTHOR

Syntax:

AUTHOR [*field-argument*] [*switch*]

Description: Retrieves, and optionally sets, the document author's name, as recorded in the Creator element of the Core File Properties part or, if *field-argument* is present, the name specified by *text* in *field-argument*. Specifying a *field-argument* shall change Creator to *text*.

Field Value: The document author's name.

Switches: One of the following *general-formatting-switch*es: `\* Caps`, `\* FirstCap`, `\* Lower`, or `\* Upper`.

[*Example*: Consider the case in which the Creator element is as follows:

```
<Creator>William Jones</Creator>
```

and the following field is updated:

```
AUTHOR
```

The result is:

```
William Jones
```

Updating the following field:

```
AUTHOR "Tony Caruso"
```

causes the Creator element to take on the specified value. *end example*]

2.16.5.5 AUTONUM

Syntax:

AUTONUM [*switches*]

Description: In paragraphs formatted with one of the nine built-in heading styles, paragraph numbering restarts at 1 in each successive heading level. If headings that contain AUTONUM fields are followed by body text paragraphs that also contain AUTONUM fields, the paragraph numbering of the body text is restarted at 1 after each heading. If the headings don't contain AUTONUM fields, body text paragraphs that contain AUTONUM fields are numbered in a continuous, sequential series throughout the document. [*Note*: This field is supported for legacy reasons, It is recommended that LISTNUM (§2.16.5.40) be used instead. *end note*]

The XML generated for a complex field implementation shall not have the optional field value stored.

Field Value: A new paragraph number in ascending sequential order.

Switches: Zero or one of the *general-formatting-switch*es, or zero or more of the following *field-specific-switch*es.

\s *field-argument*	*text* in this switch's *field-argument* specifies the separator character to be used. If \s is omitted, a period (.) is used.

[*Example*: When the following fields are updated:

```
AUTONUM
AUTONUM \* Arabic \s :
AUTONUM \* alphabetic \s " "xxx
AUTONUM \* ROMAN
AUTONUM \* OrdText
```

The results are:

```
1.
2:
c xxx
IV.
fifth.
```

end example]

2.16.5.6 AUTONUMLGL

Syntax:

AUTONUMLGL [*switches*]

Description: For legal and technical publications, use the nine built-in heading styles to format headings in the document, and then insert an AUTONUMLGL field at the beginning of each heading paragraph. The numbers reflect the heading levels that correspond to the heading styles. If an AUTONUMLGL field is inserted in paragraphs of body text paragraphs not formatted with built-in heading styles, the number of the preceding heading is included in the paragraph number. [*Note*: This field is supported for legacy reasons, It is recommended that LISTNUM (§2.16.5.40) be used instead. *end note*]

This field only makes sense in terms of multi-level headings. Given the following headings:

Heading 1
Heading 2
Heading 2
Heading 1

this field allows

1. Heading 1
1.1. Heading 2
1.2. Heading 2
2. Heading 1

At each level, the numbering sequence does two things—it increments specific to that level, and it includes the value from the previous level.

The XML generated for a complex field implementation shall not have the optional field value stored.

Field Value: A new paragraph number in ascending sequential order.

Switches: Zero or one of the *general-formatting-switch*es, or zero or more of the following *field-specific-switch*es.

\e	Removes the trailing separator (period).
\s *field-argument*	*text* in this switch's *field-argument* specifies the separator character to be used. If \s is omitted, a period (.) is used.

[*Example*: When the following fields are updated:

```
AUTONUMLGL
AUTONUMLGL \* Arabic \s :
AUTONUMLGL \* alphabetic \s " "xxx
AUTONUMLGL \* ROMAN
AUTONUMLGL \e xxx
```

The results are:

```
1.
2:
c xxx
IV.
5xxx
```

end example]

2.16.5.7 AUTONUMOUT

Syntax:

AUTONUMOUT

Description: Use the nine built-in heading styles to format headings in the document, and then insert an AUTONUMOUT field at the beginning of each heading paragraph. The numbers reflect the heading levels that correspond to the heading styles. [*Note*: This field is supported for legacy reasons, It is recommended that LISTNUM (§2.16.5.40) be used instead. *end note*]

The XML generated for a complex field implementation shall not have the optional field value stored.

This field allows the numbering to be incremented based on the heading level. Given the following:

{AutoNumOut} Heading 1
{AutoNumOut} Heading 2
{AutoNumOut} Heading 2
{AutoNumOut} Heading 1

results in

I. Heading 1
A. Heading 2
B. Heading 2
II. Heading 1

Field Value: A paragraph number.

Switches: None.

[*Example*: When the following fields are updated:

```
AUTONUMOUT
AUTONUMOUT
```

The results are:

```
1.
2.
```

end example]

2.16.5.8 AUTOTEXT

Syntax:

AUTOTEXT *field-argument*

Description: Inserts the AutoText entry whose name is specified by *text* in *field-argument*.

Regarding XML generation, the field result is the value of the autotext. [*Note*: This can be arbitrarily complex and involve VML *end note*]

Field Value: The specified AutoText entry.

Switches: None.

[*Example*: Assuming the following entries are defined with values of current page number, salutation, and a notice:

```
AUTOTEXT "- PAGE -"
AUTOTEXT "Yours truly,"
AUTOTEXT Confidential
```

when evaluated, they might produce the following results.

```
 - 13 -
Yours truly,
CONFIDENTIAL
```

end example]

2.16.5.9 AUTOTEXTLIST

Syntax:

AUTOTEXTLIST *field-argument* [*switches*]

Description: Creates a shortcut menu based on AutoText entries in the active template. The list can vary based on the styles applied to the AutoText entries. *text* in *field-argument* is inserted into the document.

A complex field XML implementation shall be used, and the *field-argument* text shall be placed in one or more runs between the separate and end parts of the fldChar element.

Field Value: *text* in *field-argument*.

Switches: Zero or more of the following *field-specific-switch*es.

`\s` *field-argument*	Specifies that the list is to contain entries based on the style specified by *text* in this switch's *field-argument*. Without this switch, entries of the current paragraph style appear. If there are no entries for the current style, all entries appear. The style can be a paragraph style or a character style.
`\t` *field-argument*	*text* in this switch's *field-argument* specifies the text to show in the ScreenTip.

[*Example*: The field:

```
{ AUTOTEXTLIST "List of salutations" \s Salutation \t "Choose a salutation" }
```

causes the following to be displayed: In the document, the Salutation list; in the ScreenTip, `Choose a salutation`, and on the shortcut menu, the list of entries whose style is `Salutation`.

end example]

2.16.5.10 BARCODE

Syntax:

BARCODE *field-argument* [*switch*]

Description: Produces a postal bar code is a machine-readable form of address used by the U.S. Postal Service. The barcode is in the form of either a POSTNET delivery-point bar code or a Facing Identification Mark (FIM). *text* in *field-argument* can be either a postal address or a bookmark name. In the case of a postal address, all that is needed is a 5-digit or 9-digit ZIP code; the rest of the address is superfluous.

Field Value: A postal bar code.

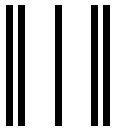
Switches: Zero or more of the following *field-specific-switch*es.

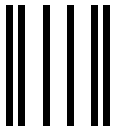
`\b`	Indicates that *text* in *field-argument* is the name of a bookmark.
`\f` *field-argument*	Inserts a Facing Identification Mark (FIM). *text* in this switch's *field-argument* shall be either "A" (courtesy reply mark) or "C" (business reply mark).
`\u`	Indicates that *text* in *field-argument* is a U.S. postal address.

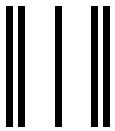
[*Example*: Consider the case in which `PostalAddress` is the name of a bookmark for the text "2051 Swans Neck Way, Reston VA 20191". When the following fields are updated:

```
BARCODE 20191
BARCODE 20191 \u
BARCODE 20191-4023 \u
BARCODE "2051 Swans Neck Way, Reston VA 20191" \u
BARCODE "2051 Swans Neck Way, Reston VA 20191" \f A
BARCODE 20191 \f C
BARCODE PostalAddress \b \f A
```

The results are:







end example]

2.16.5.11 BIBLIOGRAPHY

Syntax:

```
BIBLIOGRAPHY [ switch ]
```

Description: Retrieves and displays the contents of the document's Bibliography part in the bibliographic style specified within the SelectedStyle attribute of the Sources (§7.6.2.60) element of the Bibliography part.

Field Value: The formatted bibliographic data for all sources in the current document.

Switches: The following *field-specific-switch*.

\l *field-argument*	The *text* in this switch's *field-argument* specifies the locale that shall be used in to format the bibliographic sources in the document that do not specify a locale

	using the LCID (§7.6.2.39) element.
\f *field-argument*	The *text* in this switch's *field-argument* specifies the locale that shall be used to filter the bibliographic data to only the sources in the document that have a value matching *field-argument* in the LCID (§7.6.2.39) element.
\m *field-argument*	The *text* in this switch's *field-argument* specifies that only the source with a Tag (§7.6.2.65) element value matching *field-argument* shall be displayed in the bibliography.

[*Example:* Consider a document with bibliographic data for this Office Open XML Standard:

Author: Ecma International
Title: Office Open XML Document Interchange Specification
Year: 2006
Month: October

the following field is updated:

```
BIBLIOGRAPHY /l 1033
```

The result for MLA is:

```
Ecma International. Office Open XML Document Interchange Specification. October
2006.
```

And for APA:

```
Ecma International. (2006, October). Office Open XML Document Interchange
Specification.
```

end example]

2.16.5.12 BIDIOUTLINE

Syntax:

```
BIDIOUTLINE
```

Description: Sets the output as being right-to-left. This field is like AUTONUMOUT (§2.16.5.7), except for differences in Arabic/Hebrew numbering [*Example*: For Heading 2, BIDIOUTLINE results in א, while AUTONUMOUT results in A. *end example*]

Field Value: A paragraph number.

Switches: None.

2.16.5.13 CITATION

Syntax:

CITATION *field-argument* [*switch*]

Description: Displays the contents of the Source (§7.6.2.59) element with a Tag (§7.6.2.65) element value matching *field-argument* using the bibliographic style specified within the SelectedStyle attribute of the Sources (§7.6.2.60) element of the Bibliography part.

Field Value: The comments relating to the current document.

Switches: The following *field-specific-switch*.

\l *field-argument*	The *text* in this switch's *field-argument* specifies the locale which shall be used in conjunction with the specified bibliographic style to format the citation in the document.
\f *field-argument*	The *text* in this switch's *field-argument* specifies the prefix which shall be prepended to the citation.
\s *field-argument*	The *text* in this switch's *field-argument* specifies the suffix which shall be appended to the citation.
\p *field-argument*	The *text* in this switch's *field-argument* specifies the page number associated with the citation.
\v *field-argument*	The *text* in this switch's *field-argument* specifies the volume number associated with the citation.
\n	Specifies that the author information shall be suppressed from the citation.
\t	Specifies that the title information shall be suppressed from the citation.
\y	Specifies that the year information shall be suppressed from the citation.
\m *field-argument*	The *text* in this switch's *field-argument* specifies the Tag (§7.6.2.65) element value for another source to be included in this citation's field result.

[*Example:* Consider a case with bibliographic data for this Office Open XML Standard:

Tag: Ecma01
Author: Ecma International
Title: Office Open XML Document Interchange Specification
Year: 2006
Month: October

the following field is updated:

```
CITATION Ecma01 /l 1033
```

The result for MLA is:

```
(Ecma International)
```

And for APA:

```
(Ecma International, 2006)
```

end example]

2.16.5.14 COMMENTS

Syntax:

COMMENTS [*field-argument*] [*switch*]

Description: Retrieves, and optionally sets, the comments relating to the current document, as recorded in the Description element of the Core File Properties part or, if *field-argument* is present, the comments specified by *text* in *field-argument*. Specifying a field-argument shall change Description to *text*.

Field Value: The comments relating to the current document.

Switches: One of the following *general-formatting-switch*es: `\* Caps`, `\* FirstCap`, `\* Lower`, or `\* Upper`.

[*Example*: Consider the case in which the Description element is as follows:

```
<Description>Once upon a time, in a land far, far away ...</Description>
```

and the following field is updated:

COMMENTS

The result is:

Once upon a time, in a land far, far away ...

Updating the following field:

COMMENTS "I came, I saw, I was not impressed."

causes the Description element to take on the specified value. *end example*]

2.16.5.15 COMPARE

Syntax:

COMPARE Expression-1 Operator Expression-2

Expression-1:
expression

Expression-2:
expression

Description: Compares the values designated by *Expression-1* and *Expression-2* using the operator designated by *Operator*. [*Note*: This field can be used to create compound logical comparisons with AND and OR functions in a formula, and then by using the result of the formula in an IF field. *end note*]

Operator can be any one of the six relational and equality operators specified for *operator* (§2.16.3.3).

If *Operator* is = or <>, *Expression-2* can contain a question mark (?) to represent any single character, or an asterisk (*) to represent any string of characters. The expression shall be enclosed in quotation marks so that it is compared as a character string. If an asterisk is used in *Expression-2*, the portion of *Expression-1* that corresponds to the asterisk, plus any remaining characters in *Expression-2*, shall NOT exceed 128 characters.

Field Value: 1 if the comparison is true, or 0 if the comparison is false.

Switches: None.

[*Example*: Consider the case in which the IF field in the following example is inserted into a mail merge main document. The COMPARE fields examine the data fields CustomerNumber and CustomerRating as each data record is merged. The OR function of the formula returns the value 1 if at least one of the data fields indicates poor credit, in which case the first text in quotation marks is printed:

```
{ IF { = OR ( { COMPARE { MERGEFIELD CustomerNumber } >= 4 },
  { COMPARE { MERGEFIELD CustomerRating } <= 9 } ) } = 1
  "Credit not acceptable" "Credit acceptable"}
```

The following COMPARE field results in the value 1 if any value in the PostalCode data field is the range 98500–98599:

```
{ COMPARE "{ MERGEFIELD PostalCode }" = "985*" }
```

end example]

2.16.5.16 CREATEDATE

Syntax:

CREATEDATE [switches *]*

Description: Retrieves the date and time at which the document was created, as recorded in the DateCreated element of the Core File Properties part. By default, the Gregorian calendar is used and the *date-and-time-formatting-switch* used is implementation-defined.

Field Value: The date and time at which the document was created.

Switches: Zero or one *date-and-time-formatting-switch* and zero or one of the following *field-specific-switch*es.

\h	Use the Hijri/Lunar calendar.
\s	Use the Saka Era calendar.

[*Example*: Consider the case in which the DateCreated element is as follows:

```
<DateCreated>2006-01-05T03:31:00Z</DateCreated>
```

and the following fields are updated in a US-English context that is UTC -5:

CREATEDATE
CREATEDATE \@ "dddd, MMMM dd, yyyy HH:mm:ss"
CREATEDATE \@ "dddd, MMMM dd, yyyy HH:mm:ss" \h
CREATEDATE \@ "dddd, MMMM dd, yyyy HH:mm:ss" \s

the results are:

```
1/4/2006 10:31:00 PM
Wednesday, January 04, 2006 22:31:00
AlArbia'a, Thoul Hijjah 04, 1426 22:31:00
Budhavara, Pausa 14, 1927 22:31:00
```

end example]

2.16.5.17 DATABASE

Syntax:

DATABASE [switches *]*

Description: Inserts the results of a database query into a WordprocessingML table. If the number of columns is 62 or more, the field inserts the results of a query in columns separated by tabs. The DATABASE field contains all the information needed to connect to a database and perform an SQL query. Each time the field is updated, the database is queried again.

Field Value: The results of a database query as a WordprocessingML table.

Switches: Zero or more of the following *field-specific-switch*es.

\b *field-argument*	The *text* in this switch's *field-argument* specifies which attributes of the format set by the \l switch are to be applied to the table. If the \l switch is blank, the \b switch value shall be 16 (AutoFit). *text* can have a value that is the bitwise-or of any combination of the following: 0, None 1, Borders 2, Shading 4, Font 8, Color 16, AutoFit 32, Heading Rows 64, Last Row

	128, First Column 256, Last Column
\c *field-argument*	The *text* in this switch's *field-argument* specifies a connection to the data.
\d *field-argument*	The *text* in this switch's *field-argument* specifies the complete path and file name of the database. Used for all database queries except a query to an SQL database table using ODBC.
\f *field-argument*	The *text* in this switch's *field-argument* specifies the integral record number of the first data record to insert
\h	Inserts the field names from the database as column headings in the resulting table.
\l *field-argument*	The *text* in this switch's *field-argument* specifies the format that is to be applied to the result of the database query. If this switch is used and the \b switch doesn't specify the table attributes, an unformatted table is inserted.
\o *field-argument*	Inserts data at the beginning of a merge. By adding the \o switch to the database field, it will only get the data for the database field at the beginning of a merge instead of once for each record merged. This is a performance optimization and should only be used when the database field doesn't rely on record specific information to gather.
\s *field-argument*	The *text* in this switch's *field-argument* specifies a set of SQL instructions. Each quotation mark in the instructions shall be preceded by a backslash (\).
\t *field-argument*	The *text* in this switch's *field-argument* specifies the integral record number of the last data record to insert.

[*Example*: The following field results from a query to a database through ODBC:

```
{ DATABASE \d "C:\\Data\\Sales93.mdb" \c "DSN=MS Access Database;
  DBQ=C:\\Data\\Sales93.mdb; FIL=RedISAM"
  \s "select * from \"Customer List\"" \f "2445" \t "2486" \l "2"
```

end example]

2.16.5.18 DATE

Syntax:

DATE [switches *]*

Description: Retrieves the current date and time. By default, the Gregorian calendar is used, and the *date-and-time-formatting-switch* used is implementation-defined.

Field Value: The current date and time.

Switches: Zero or one *date-and-time-formatting-switch* and zero or one of the following *field-specific-switch*es.

\h	Use the Hijri/Lunar calendar.
\l	If no *date-and-time-formatting-switch* is used, the date shall use the date format last used by the hosting application when inserting a new DATE field.
\s	Use the Saka Era calendar.

[*Example*: Consider the case in which the following fields are updated in a US-English context on the given date and time:

DATE
DATE \@ "dddd, MMMM dd, yyyy HH:mm:ss"
DATE \@ "dddd, MMMM dd, yyyy HH:mm:ss" \h
DATE \@ "dddd, MMMM dd, yyyy HH:mm:ss" \s

the results are:

```
1/5/2006
Thursday, January 05, 2006  19:09:01
AlKhamis, Thoul Hijjah 05, 1426  19:09:01
Bruhaspathivara, Pausa 15, 1927  19:09:01
```

end example]

[*Note*: For some *date-and-time-formatting-switch*es, the DATE and TIME (§2.16.5.72) fields can produce the same result. *end note*]

2.16.5.19 DOCPROPERTY

Syntax:

DOCPROPERTY docprop-category [field-argument *] [* switches *]*

docprop-category:
AUTHOR | *BYTES* | *CATEGORY* | *CHARACTERS* | *CHARACTERSWITHSPACES*
| *COMMENTS* | *COMPANY* | *CREATETIME* | *HYPERLINKBASE* | *KEYWORDS*
| *LASTPRINTED* / *LASTSAVEDBY* | *LASTSAVEDTIME* | *LINES* | *MANAGER*
| *NAMEOFAPPLICATION* | *ODMADOCID* | *PAGES* | *PARAGRAPHS* | *REVISIONNUMBER*
| *SECURITY* | *SUBJECT* | *TEMPLATE* | *TITLE* | *TOTALEDITINGTIME* | *WORDS*

Description: Retrieves the indicated document information. For some combinations of DOCPROPERTY and *docprop-category*, there is an equivalent field, in which case, the description for the combination can be obtained from that field. For those combinations not having an equivalent field, the description is shown

directly. When used directly, some of the equivalent fields allow the value of the designated property to be changed. However, when the corresponding DOCPROPERTY field is used, such values shall not be changed. This is indicated in the following table by "Read-only operation."

docprop-category	Corresponding Field	Description
AUTHOR	AUTHOR (§2.16.5.4)	**Read-only operation.**
BYTES	FILESIZE (§2.16.5.24)	
CATEGORY	No equivalent	The contents of the Category element of the Core File Properties part.
CHARACTERS	NUMCHARS (§2.16.5.48)	
CHARACTERSWITHSPACES	No equivalent	**Like NUMCHARS, but includes all white space characters as well.**
COMMENTS	COMMENTS (§2.16.5.11)	**Read-only operation.**
COMPANY	No equivalent	The contents of the Company element of the Application-Defined File Properties part.
CREATETIME	CREATEDATE (§2.16.5.16)	
HYPERLINKBASE	No equivalent	The contents of the HyperlinkBase element of the Application-Defined File Properties part.
KEYWORDS	No equivalent	The contents of the Keywords element of the Core File Properties part.
LASTPRINTED	PRINTDATE (§2.16.5.54)	
LASTSAVEDBY	LASTSAVEDBY (§2.16.5.38)	
LASTSAVEDTIME	SAVEDATE (§2.16.5.60)	
LINES	No equivalent	The contents of the Lines element of the Application-Defined File Properties part.
MANAGER	No equivalent	The contents of the Manager element of the Application-Defined File

docprop-category	Corresponding Field	Description
		Properties part.
NAMEOFAPPLICATION	No equivalent	The contents of the Application element of the Application-Defined File Properties part.
ODMADOCID		
PAGES	NUMPAGES (§2.16.5.49)	
PARAGRAPHS	No equivalent	The contents of the Paragraphs element of the Application-Defined File Properties part.
REVISIONNUMBER	REVNUM (§2.16.5.59)	
SECURITY	No equivalent	The contents of the DocSecurity element of the Application-Defined File Properties part.
SUBJECT	SUBJECT (§2.16.5.67)	**Read-only operation.**
TEMPLATE	TEMPLATE (§2.16.5.71)	
TITLE	TITLE (§2.16.5.73)	**Read-only operation.**
TOTALEDITINGTIME	EDITTIME (§2.16.5.21)	
WORDS	No equivalent	The contents of the Words element of the Application-Defined File Properties part.

Field Value: The indicated document information.

2.16.5.20 DOCVARIABLE

Syntax:

DOCVARIABLE field-argument

Description: Inserts the string assigned to the document variable designated by *text* in *field-argument*. Each WordprocessingML document has a collection of variables. This field is used to access and display the contents of docVar (§2.15.1.30) elements in the Document Settings part.

Field Value: The value of the specified document variable.

Switches: None.

2.16.5.21 EDITTIME

Syntax:

EDITTIME [switch *]*

Description: Retrieves the total editing time, in minutes, since the document was created, as recorded in the TotalTime element of the Application-Defined File Properties part. By default, the *numeric-formatting-switch* or *general-formatting-switch* used is implementation-defined.

Field Value: The total editing time, in minutes.

Switches: Zero or one *numeric-formatting-switch* or *general-formatting-switch*.

[*Example*: Consider the case in which the TotalTime element is as follows:

```
<TotalTime>930</TotalTime>
```

and the following fields are updated in a US-English context:

```
EDITTIME
EDITTIME \* OrdText
EDITTIME \# "#,##0"
```

the results are:

```
930
nine hundred thirtieth
 930
```

end example]

2.16.5.22 EQ

Syntax:

EQ eq-primary-switch *[* switches *] ([* eq-argument-list *])*

eq-argument-list is a list of arguments separated using a separator character. For implementations using a period (.) as the radix point, the separator character is a comma (,). For implementations using a comma (,) as the radix point, the separator character is a semicolon (;).

Description: Computes the specified mathematical equation.

Field Value: The result of the specified mathematical equation. [*Note*: The result of an EQ field can be used as an argument in another EQ field's *eq-argument-list*. *end note*]

Switches: One of the following *eq-primary-switch*es: `\a`, `\b`, `\d`, `\f`, `\i`, `\l`, `\o`, `\r`, `\s`, and `\x`. Each of these switches has one or more subswitches, as shown below.

\a produces an array using the argument values in *eq-argument-list* (which are in row-major order) and the *field-specific-switch*es below:

\ac	Alignment is centered in each array column.
\al	Alignment is left in each array column.
\ar	Alignment is right in each array column.
\co *field-argument*	The number of columns in the array is specified by *text* in this switch's *field-argument*. In the absence of this switch, the number is 1.
\hs *field-argument*	Adds the integral number of points of horizontal spacing specified by *text* in this switch's *field-argument* between columns.
\vs *field-argument*	Adds the integral number of points of vertical spacing specified by *text* in this switch's *field-argument* between lines.

\b brackets the single element in *eq-argument-list* in a size appropriate for that element. The default form of brackets is parentheses. The *field-specific-switch*es below may be used:

\bc *char*	Uses the character designated by *char* as both the left and right bracket character. However, if *char* is {, [, (, or <, that character is used for the left bracket, and },],), or >, respectively, is used for right bracket.
\lc *char*	Uses the character designated by *char* as the left bracket character.
\rc *char*	Uses the character designated by *char* as the right bracket character.

\d Controls where the next character following the EQ field is drawn (that is, the displacement). *eq-argument-list* shall have no arguments. The *field-specific-switch*es below may be used:

\ba *field-argument*	Draws to the left (backward) the integral number of points specified by *text* in this switch's *field-argument*.
\fo *field-argument*	Draws to the right (forward) the integral number of points specified by *text* in this switch's *field-argument*.
\li	Underlines the space up to the next character.

\f Creates a fraction with the first argument as numerator and the second argument as denominator, centered above and below the division line, respectively. *eq-argument-list* shall have exactly two arguments. There are no *field-specific-switch*es for this switch.

\i Creates an integral using the specified or default symbol and three elements. The first argument is the lower limit, the second is the upper limit, and the third is the integrand. *eq-argument-list* shall have exactly three arguments. The *field-specific-switch*es below may be used:

\fc *char*	Uses the character designated by *char* as the fixed-height character for

	the symbol.
\in	Uses an inline format with the limits displayed to the right of the symbol instead of above and below it.
\pr	Uses the symbol Capital pi and creates a product.
\su	Uses the symbol Capital sigma and creates a summation.
\vc *char*	Uses the character designated by *char* as the variable-height character for the symbol. The symbol matches the height of the third argument.

\lf Creates a list from an arbitrary number of arguments. There are no *field-specific-switch*es for this switch.

\o Using an arbitrary number of arguments, displays each successive argument on top of the previous one. Each character is displayed within an invisible character box, with the switches being available to align the boxes on top of one another. The *field-specific-switch*es below may be used:

\ac	Alignment character box center (the default).
\al	Alignment character box left.
\ar	Alignment character box right.

\r Creates a radical. *eq-argument-list* shall have either one or two arguments. If it has one argument, the result is the square root of that argument. If it has two arguments, the result is the *n*th root of the second argument, where *n* is the first argument. There are no *field-specific-switch*es for this switch.

\s Creates a subscript or superscript. One or more arguments are permitted. If more than one element is specified, the elements are stacked and left-aligned. The *field-specific-switch*es below may be used:

\ai *field-argument*	Adds space above a line in a paragraph by the integral number of points specified by *text* in this switch's *field-argument*. The default is 2 points.
\di *field-argument*	Adds space below a line in a paragraph by the integral number of points specified by *text* in this switch's *field-argument*.
\do *field-argument*	Moves a single argument below the adjacent text by the integral number of points specified by *text* in this switch's *field-argument*. The default is 2 points.
\up *field-argument*	Moves a single argument above the adjacent text by the integral number of points specified by *text* in this switch's *field-argument*.

\x Creates one or more border segments around a single argument. By default, all four borders are added. *eq-argument-list* shall have no arguments. The *field-specific-switch*es below may be used:

\bo	Draws a horizontal border below the argument.
\le	Draws a vertical border to the left of the argument.

\ri	Draws a vertical border to the right of the argument.
\to	Draws a horizontal border above the argument.

[*Example*: When the following fields are updated:

EQ \a \co 2 \ac \hs 10 (1000, 20, A, Sunday)
EQ \b \bc \| (-100) EQ \b \bc \| (\r(3, a + b)
xx EQ \d \fo 20 () xx EQ \d \fo 30 \li ()xx
EQ \f (1, 32) EQ \f (7, 64)
EQ \i (0, ∞, x) EQ \i \su \in (0, 10, x) EQ \i \pr \in (0, 5, x)
EQ \i \fc \{ (0, 5, \f (x, 0.34)) EQ \i \vc \{ (0, 5, \f (x, 0.34))

EQ \l (0, 10)
EQ \b \lc \[\rc \) (\l (0, 10))
EQ \o (0, 0, 0) EQ \o (0, +) EQ \o \ar (0, |, _)
EQ \r (2) EQ \r (2, x)
a EQ \s \up (2) + b EQ \s \up (2)
a EQ \x (+) b a EQ \x \to \le (+) b a EQ \x \bo \ri (+) b

The results are:

1000 20
A Sunday

| -100 | $\left| \sqrt[3]{a + b} \right|$

xx xx_____ xx

$\frac{1}{32}$ $\frac{7}{64}$

$\int_0^{\infty} x$ $\sum_0^{10} x$ $\prod_0^{5} x$

$\{ \frac{x}{0.34} \right._0^5$

0, 10
[0, 10)

$\mathbf{0}\quad \not{0}\quad \underline{\emptyset}$

$\sqrt{2}\quad \sqrt[2]{x}$

$a^2 + b^2$

$a\boxed{+}b\quad a\sqrt{+}b\quad a\underline{+}b$

end example]

2.16.5.23 FILENAME

Syntax:

FILENAME [*switch*]

Description: Retrieves the name of the current document as stored on disk.

Field Value: The name of the current document.

Switches: One of the following *general-formatting-switch*es: * Caps, * FirstCap, * Lower, or * Upper, and zero or one of the following *field-specific-switch*es.

\p	Include the full file path name.

[*Example*: Consider the case in which the following fields are updated:

```
FILENAME \* Upper
FILENAME \p
```

the results might be:

```
FIELD DEMO SUITE.DOCX
E:\Std\OOXML\Fields\Field Demo Suite.docx
```

end example]

2.16.5.24 FILESIZE

Syntax:

FILESIZE [*switches*]

Description: Retrieves the size of the current document in bytes. [*Note*: This information is not stored inside the document's XML. It needs to be obtained from the file system. *end note*]

Field Value: The size of the current document in bytes.

Switches: Zero or one *numeric-formatting-switch* or *general-formatting-switch* and zero or one of the following *field-specific-switch*es.

\k	Round to the nearest kilobyte.
\m	Round to the nearest megabyte.

[*Example*: Consider the case in the document's size is 4,660,736 bytes and the following fields are updated:

```
FILESIZE \# #,##0
FILESIZE \k
FILESIZE \m
```

the results are:

```
4,660,736
4661
5
```

end example]

2.16.5.25 FILLIN

Syntax:

FILLIN [*field-argument*] [*switch*]

Description: Prompts the user to enter text. *text* in *field-argument* contains the prompt. The prompt is displayed each time the field is updated. When a new document is created based on a template containing FILLIN fields, those fields are updated automatically.

Field Value: The user's response.

Switches: Zero or more of the following *field-specific-switch*es.

\d *field-argument*	The *text* in this switch's *field-argument* specifies a default response if one is not entered. If no default response is specified, the most recent response is used. To specify a blank entry as the default, *field-argument* shall be "".
\o	When used in a mail merge main document, this causes the display of the prompt once instead of each time a new data record is merged. The same response is inserted in each resulting merged document.

[*Example*: The following FILLIN field helps the user fill in the correct information by displaying the patient name from the current data record:

```
FILLIN "Please enter the appointment time for MERGEFIELD PatientName :"
```

end example]

2.16.5.26 FORMCHECKBOX

Syntax:

FORMCHECKBOX

Description: Inserts a check box style form field which, when the editing of form fields is enabled using the documentProtection element (§2.15.1.28), can be checked and unchecked. An instance of this field shall be accompanied by a use of the ffData element (§2.16.17) which contains the form field's properties.

Field Value: A check box based on the properties of the ffData element (§2.16.17).

Switches: None.

[*Example*: Consider the following fields:

```
FORMCHECKBOX
```

Assuming the appropriate properties are used in the child XML elements of the field, a check box is displayed.

end example]

2.16.5.27 FORMDROPDOWN

Syntax:

FORMDROPDOWN

Description: Inserts a drop-down list style form field which, when the editing of form fields is enabled using the documentProtection element (§2.15.1.28), can be used to select an entry in the list. An instance of this field shall be accompanied by a use of the ffData element (§2.16.17) which contains the form field's properties.

Field Value: A drop-down list based on the properties of the ffData element (§2.16.17).

Switches: None.

[*Example*: Consider the following fields:

```
FORMDROPDOWN
```

Assuming the appropriate properties are used in the child XML elements of the field, a drop-down list is displayed.

end example]

2.16.5.28 FORMTEXT

Syntax:

FORMTEXT

Description: Inserts a text box style form field which, when the editing of form fields is enabled using the documentProtection element (§2.15.1.28), can be typed into. An instance of this field shall be accompanied by a use of the ffData element (§2.16.17) which contains the form field's properties.

Field Value: A text box based on the properties of the ffData element (§2.16.17).

Switches: None.

[*Example*: Consider the following fields:

```
FORMTEXT
```

Assuming the appropriate properties are used in the child XML elements of the field, a text box is displayed.

end example]

2.16.5.29 GOTOBUTTON

Syntax:

GOTOBUTTON *field-argument-1 field-argument-2*

field-argument-1:
expression

field-argument-2:
expression

Description: Inserts a jump command, such that when it is activated, the insertion point of the document is moved to the location specified by *text* in *field-argument-1*. *text* can be a bookmark, a page number, or some other item (as described below). The page number can be a reference resulting from a REF field. The other items than can be locations are:

a *n*	annotation
f *n*	footnote
l *n*	line
p *n*	page
s *n*	section

where *n* is an integer that designates the *n*th occurrence of the corresponding item (which is not necessarily the item numbered *n*).

text in *field-argument-2* is the text or graphic "button" that appears in the document, such that it can be selected to activate the jump. [*Note*: The BOOKMARK and INCLUDEPICTURE fields make for some interesting button possibilities. *end note*] The text or graphic shall appear on one line in the field result; otherwise, an error occurs.

Field Value: None.

Switches: None.

[*Example*: Consider the following fields:

```
GOTOBUTTON MyBookmark Dest
GOTOBUTTON p3 Page
GOTOBUTTON "f 2" Footnote
```

When the `Dest` "button" is activated, the insertion point becomes the location marked by `MyBookmark`. When `Page` is activated, the insertion point becomes the beginning of the third page. When Footnote is activated, the insertion point becomes the marker of the second footnote, at the place it is used in the document, not in any footnote list.

end example]

2.16.5.30 GREETINGLINE

Syntax:

GREETINGLINE [*switches*]

Description: Inserts a mail merge greeting line.

Field Value: The greeting line.

Switches: Zero or more of the following *field-specific-switch*es.

\c *field-argument*	The *text* in this switch's *field-argument* specifies the text to include in the merge field if the name field in the data source is blank.
\c *field-argument*	The *text* in this switch's *field-argument* specifies the format of the name included in the field.
\l *field-argument*	The *text* in this switch's *field-argument* specifies the language ID used to format the name. it defaults to the language ID of the first character of the document.

2.16.5.31 HYPERLINK

Syntax:

HYPERLINK *field-argument* [*switches*]

Description: When selected, causes control to jump to the location specified by *text* in *field-argument*. That location can be a bookmark or a URL.

Field Value: None.

Switches: Zero or more of the following *field-specific-switch*es.

`\l` *field-argument*	*text* in this switch's *field-argument* specifies a location in the file, such as a bookmark, where this hyperlink will jump.
`\m`	Appends coordinates to a hyperlink for a server-side image map.
`\n`	Causes the destination site to be opened in a new window.
`\o` *field-argument*	*text* in this switch's *field-argument* specifies the ScreenTip text for the hyperlink.
`\t` *field-argument*	*text* in this switch's *field-argument* specifies the target to which the link should be redirected. Use this switch to link from a frames page to a page that you want to appear outside of the frames page. The permitted values for *text* are: • `_top`, whole page (the default) • `_self`, same frame • `_blank`, new window • `_parent`, parent frame

[*Example*:

```
HYPERLINK http://www.cnu.edu/
HYPERLINK "E:\\ReadMe.txt"
```

end example]

2.16.5.32 IF

Syntax:

`IF` *Expression-1 Operator Expression-2 field-argument-1 field-argument-2*

Expression-1:
expression

Expression-2:
expression

field-argument-1:
expression

field-argument-2:
expression

Description: Compares the values designated by *Expression-1* and *Expression-2* using the operator designated by *Operator*.

Operator can be any one of the six relational and equality operators specified for *operator* (§2.16.3.3).

If *Operator* is = or <>, *Expression-2* can contain a question mark (?) to represent any single character, or an asterisk (*) to represent any string of characters. The expression shall be enclosed in quotation marks so that it is compared as a character string. If an asterisk is used in *Expression-2*, the portion of *Expression-1* that corresponds to the asterisk, plus any remaining characters in *Expression-2*, shall NOT exceed 128 characters.

Field Value: *field-argument-1* if the comparison is true; otherwise, *field-argument-2*.

Switches: None.

[*Example*: The following example specifies that if the customer order is greater than or equal to 100 units, the result is "Thanks"; but if the customer order is fewer than 100 units, the result is "The minimum order is 100 units":

```
{IF order >= 100 "Thanks" "The minimum order is 100 units" }
```

For other examples, see §2.16.2, and the COMPARE field (§2.16.5.15) and the QUOTE field (§2.16.5.55). *end example*]

2.16.5.33 INCLUDEPICTURE

Syntax:

INCLUDEPICTURE *field-argument* [*switches*]

Description: Retrieves the picture contained in the document named by *field-argument*. If *field-argument* contains white space, it shall be enclosed in double quotes. If *field-argument* contains any backslash characters, each one shall be preceded directly by another backslash character.

Field Value: The specified picture.

Switches: Zero or more of the following *field-specific-switch*es.

\c *field-argument*	If *text* in this switch's *field-argument* identifies the graphics filter to be used.
\d	Reduce the file size by not storing graphics data with the document.

[*Example*:

```
INCLUDEPICTURE "G:\\Photos\\Ellen in Oslo.jpg"
```

end example]

2.16.5.34 INCLUDETEXT

Syntax:

INCLUDETEXT *field-argument-1* [*field-argument-2*] [*switches*]

field-argument-1:
field-argument

field-argument-2:
field-argument

Description: Inserts all or part of the text and graphics contained in the document named by *field-argument-1*. If the document is a WordprocessingML document, the portion marked by the optional bookmark *field-argument-2* is inserted. If no such bookmark is specified here, the whole document is inserted. If the document is an XML file, the fragment referred to by an XPath expression in the \x switch is inserted. If no such switch is specified, the whole XML file is inserted.

If *field-argument-1* contains white space, it shall be enclosed in double quotes. If *field-argument-1* contains any backslash characters, each one shall be preceded directly by another backslash character.

Field Value: The specified text and graphics.

Switches: Zero or more of the following *field-specific-switch*es.

\!	Prevents this field from being updated unless all fields in the inserted text are first updated in their original document.
\c *field-argument*	Specifies that the file specified by *field-argument-2* shall be processed by a document filter whose name matches the corresponding *field-argument* value. Possible *field-argument* values are implementation-defined.
\n *field-argument*	The *text* in this switch's *field-argument* specifies a namespace mapping for XPath queries. This switch is required if the \x switch refers to an element by name in an XML file that declares a namespace.
\t *field-argument*	The *text* in this switch's *field-argument* specifies an XSLT for formatting XML data.
\x *field-argument*	The *text* in this switch's *field-argument* specifies the XPath for returning a fragment of data in an XML file.

[*Example*: The following field inserts the portion of the WordprocessingML document referred to by the bookmark Summary:

```
INCLUDETEXT "C:\\Winword\\Port Development RFP" Summary
```

The following field inserts the Name element of the XML document Resume.xml and applies the XSLT Display.xsl to it:

```
INCLUDETEXT "C:\\Resume.xml" \n xmlns:a=\"resume-schema\"
  \t "C:\\display.xsl" \x a:Resume/a:Name
```

end example]

2.16.5.35 INDEX

Syntax:

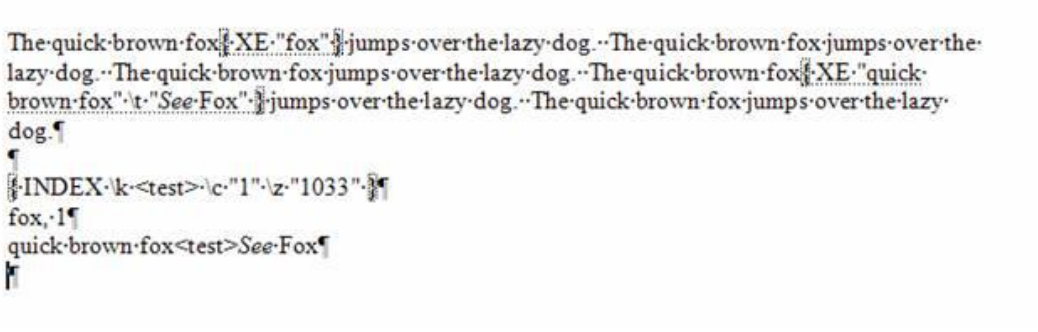
INDEX [*switches*]

Description: Builds an index using the index entries specified by XE fields (§2.16.5.79), and inserts that index at this place in the document. Each index entry and subentry is a separate paragraph unless the `\r` switch is used, in which case, an index entry and all its subentries together make up a paragraph.

Field Value: The index.

Switches: Zero or more of the following *field-specific-switch*es.

`\b` *field-argument*	Builds an index for the portion of the document marked by the bookmark indicated by *text* in this switch's *field-argument*.
`\c` *field-argument*	Builds an index having the number of columns per page specified by *text* in this switch's *field-argument*. That number can be 1–4. Without this switch, the number of columns is 1.
`\d` *field-argument*	The *text* in this switch's *field-argument* specifies a sequence of characters that is used to separate sequence numbers and page numbers when the `\s` switch is used. By default, a hyphen (-) is used.
`\e` *field-argument*	The *text* in this switch's *field-argument* specifies a sequence of characters that is used to separate an index entry and its first page number. By default, a comma (,) and space sequence is used. If *text* contains a horizontal tab character, the page number list is right justified in the column.
`\f` *field-argument*	Builds an index using only those entries having the entry type (§2.16.5.79) specified by *text* in this switch's *field-argument*. Without this switch, all entries included.
`\g` *field-argument*	The *text* in this switch's *field-argument* specifies a sequence of characters that is used to separate the start and end of a page range. By default, an en dash is used.
`\h` *field-argument*	Builds an index such that the *text* in this switch's *field-argument* occurs as a heading—formatted with the Index Heading style—at the start of each set of entries for any given letter. If the first letter in *text* is A or a, that letter is replaced with the corresponding letter for each letter set. To replace the default heading with a blank line, use a space as *text*. [*Example*:  *end example*]
`\k` *field-argument*	The *text* in this switch's *field-argument* specifies a sequence of characters that is used to separate an index entry and its cross reference (as produced by an XE entry (§2.16.5.79) having a `\t` switch). By default, a period (.) and space sequence is used. [*Example*:

	 *end example*]
`\l` *field-argument*	The *text* in this switch's *field-argument* specifies a sequence of characters that is used to separate two page numbers in a page number list. By default, a comma (,) and space sequence is used.
`\p` *field-argument*	Builds an index using only those entries whose first letter is in the range of letters specified by *text* in this switch's *field-argument*. The letter range has the form *startLetter*`-`*endLetter*. If *startLetter* is `!`, entries whose first character is not a letter, are also included, as are the letters starting from A. The letters in the range can be either upper- or lowercase.
`\r`	Runs subentries into the same line as the main entry. Colons (:) separate main entries from subentries; semicolons (;) separate subentries.
`\s` *field-argument*	The *text* in this switch's *field-argument* is used as a sequence name, and the sequence number is included along with the page number, these numbers being separated by a hyphen (-), by default. Use the `\d` switch to specify a separator character other than the default.
`\y`	Enables the use of yomi text for index entries.
`\z` *field-argument*	The *text* in this switch's *field-argument* is specifies the language ID used to generate the index as defined in the ST_LangCode (§2.18.52) simple type.

[*Example*: The index produced using the corresponding set of index entries and the field `INDEX \c "1" \e "`*tab*`" \g " to " \h "A" \z "1033"` is:

B

I

O

end example]

2.16.5.36 INFO

Syntax:

```
INFO info-category [ field-argument ] [ switches ]
info-category:
    AUTHOR | COMMENTS | CREATEDATE | EDITTIME | FILENAME | FILESIZE |
    | KEYWORDS | LASTSAVEDBY | NUMCHARS | NUMPAGES | NUMWORDS | PRINTDATE
    | REVNUM | SAVEDATE | SUBJECT | TEMPLATE | TITLE
```

A field of this kind is treated as if INFO was omitted and *info-category* was a *field-type name***.**

2.16.5.37 KEYWORDS

Syntax:

```
KEYWORDS [ field-argument ] [ switch ]
```

Description: Retrieves, and optionally sets, the document's keywords, as recorded in the Keywords element of the Core File Properties part or, if *field-argument* is present, the subject specified by *text* in *field-argument*. Specifying a *field-argument* shall change Keywords to *text*. The Keywords element contains a string of text whose format and semantics is unspecified by this Office Open XML Standard.

Field Value: The document's keywords

Switches: One of the following *general-formatting-switch*es: `\* Caps`, `\* FirstCap`, `\* Lower`, or `\* Upper`.

[*Example*: Consider the case in which the Keywords element is as follows:

```
<Keywords>switch, field, syntax</Keywords>
```

and the following field is updated:

KEYWORDS

The result is:

```
switch, field, syntax
```

Updating the following field:

```
KEYWORDS "field, formatting, switch, syntax"
```

causes the Subject element to take on the given value. *end example*]

2.16.5.38 LASTSAVEDBY

Syntax:

LASTSAVEDBY [*switch*]

Description: Retrieves the name of the user who last modified and saved the current document, as recorded in the LastModifiedBy element of the Core File Properties part.

Field Value: The name of the user who last modified and saved the current document.

Switches: One of the following *general-formatting-switch*es: `\* Caps`, `\* FirstCap`, `\* Lower`, or `\* Upper`.

[*Example*: Consider the case in which the LastModifiedBy element is as follows:

```
<LastModifiedBy>Elizabeth Martin</LastModifiedBy>
```

and the following field is updated:

```
LASTSAVEDBY \* Upper
```

the result might be:

```
ELIZABETH MARTIN
```

end example]

2.16.5.39 LINK

Syntax:

LINK *field-argument-1* *field-argument-2* [*field-argument-3*] [*switches*]

field-argument-1:
field-argument

field-argument-2:
field-argument

field-argument-3:
field-argument

Description: For information copied from another application, this field links that information to its original source file using OLE. The application type of the link information is specified by *field-argument-1*. The name and location of the source file is specified by *field-argument-2*. *field-argument-3* specifies the portion of the source file that's being linked. [*Example*: If the source file is a SpreadsheetML document, the reference might be to a cell reference or a named range. For a WordprocessingML document, it might be a bookmark. *end example*]

Field Value: None.

Switches: Zero or more of the following *field-specific-switch*es.

\a	Causes this field to be updated automatically.
\b	Inserts the linked object as a bitmap.
\d	Don't store the graphic data with the document, thus reducing the file size.
\f *field-argument*	Causes the linked object to update its formatting in a particular way, according to the integral value of *text* in this switch's *field-argument*. The possible values are: • 0 Maintain the formatting of the source file • 1 Not supported • 2 Match the formatting of the destination document • 3 Not supported • 4 Maintain the formatting of the source file, if the source file is a SpreadsheetML workbook • 5 Match the formatting of the destination document, if the source file is a SpreadsheetML workbook
\h	Inserts the linked object as HTML format text.
\p	Inserts the linked object as a picture.
\r	Inserts the linked object in rich-text format (RTF).
\t	Inserts the linked object in text-only format.
\u	Inserts the linked object as Unicode text.

[*Example*: The following example inserts a range of cells from a SpreadsheetML worksheet. The \a switch ensures that the information is updated in the WordprocessingML document whenever the worksheet is changed:

```
{ LINK Excel.Sheet.8 "C:\\My Documents\\Profits.xls"
   "Sheet1!R1C1:R4C4" \a \p }
```

end example]

2.16.5.40 LISTNUM

Syntax:

LISTNUM [*field-argument*] [*switches*]

Description: Computes the next integral number from the current or a specific series, or a specific number from the next or specific series. This field can be used anywhere in a paragraph, not just at its start. A LISTNUM field can be incorporated into numbering from a simple or outline-numbered list. *text* in *field-argument* is used to associates a LISTNUM field with a specific list. To emulate the behavior of the AUTONUM (§2.16.5.5), AUTONUMLGL (§2.16.5.6), and AUTONUMOUT (§2.16.5.7) fields, use the list names NumberDefault, LegalDefault, and OutlineDefault names, respectively. By default, the NumberFormat list is used.

The XML generated for a complex field implementation shall not have the optional field value stored.

There are nine levels of list, and, assuming `\s 1` for each, the result style used for each is as follows:

1	1)	4	(1)	7	1.
2	a)	5	(a)	8	a.
3	iii)	6	(iii)	9	iii.

Field Value: The next integral number from the current or a specific series, or a specific number from the next or specific series.

Switches: Zero or more of the following *field-specific-switch*es.

`\l` *field-argument*	*text* in this switch's *field-argument* is an integer that specifies the level in the list, overriding the default behavior of the field. If `\l` is omitted, for a new series, the default value is 3; otherwise, the current level is continued.
`\s` *field-argument*	*text* in this switch's *field-argument* is an integer that specifies the starting value for this field. If `\s` is omitted, for a new series, the default value is 1; otherwise, the current series numbering is continued.

[*Example*: When the following fields are updated:

```
We need to perform the following functions: LISTNUM  NumberDefault \l 3 \s 1 Get
approval for the project. LISTNUM Arrange for funding. LISTNUM Hire staff.
```

The results are:

```
We need to perform the following functions: i) Get approval for the
project. ii) Arrange for funding. iii) Hire staff.
```

When the following fields are updated:

```
LISTNUM NumberDefault \l 3 \s 1
LISTNUM
LISTNUM NumberDefault
LISTNUM NumberDefault \s 3
LISTNUM
LISTNUM NumberDefault \l 1
LISTNUM
LISTNUM NumberDefault \l 1 \s 1
```

```
LISTNUM LegalDefault \l 1 \s 1
LISTNUM LegalDefault
LISTNUM LegalDefault \l 1
LISTNUM LegalDefault \s 4
LISTNUM LegalDefault
```

The results are:

```
i)
ii)
iii)
iii)
iv)
2)
3)
1)
1.1.1.
1.1.2.
2.
2.1.4.
2.1.5.
```

end example]

2.16.5.41 MACROBUTTON

Syntax:

MACROBUTTON *field-argument-1 field-argument-2*

field-argument-1:
field-argument

field-argument-2:
field-argument

Description: Allows the macro or command designated by *text* in *field-argument-1* to be run. *text* in *field-argument-2* designates the text or graphic to appear as the "button" that is selected to run the macro or command.

Field Value: *field-argument-2*.

Switches: None.

2.16.5.42 MERGEFIELD

Syntax:

MERGEFIELD *field-argument* [*switch*]

Description: Retrieves the name of a data field designated by *text* in *field-argument* within the merge characters in a mail merge main document. When the main document is merged with the selected data source, information from the specified data field is inserted in place of the merge field.

The name designated by *text* shall match exactly the field name in the header record.

Field Value: The name of a data field designated by *text* in *field-argument*.

Switches: Zero or more of the following *field-specific-switch*es.

`\b` *field-argument*	The *text* in this switch's *field-argument* specifies the text to be inserted before the `MERGEFIELD` field if the field is not blank.
`\f` *field-argument*	The *text* in this switch's *field-argument* specifies the text to be inserted after the `MERGEFIELD` field if the field is not blank.
`\m`	Specifies that the `MERGEFIELD` field is a mapped field.
`\v`	Enables character conversion for vertical formatting.

[*Example*: Placing the following three MERGEFIELD fields together and using the `\f` switch ensures that the three fields have spaces between them, but only if the designated field information is present in the data source.

```
MERGEFIELD CoutesyTitle \f " " MERGEFIELD FirstName \f " "
  MERGEFIELD LastName
```

end example]

2.16.5.43 MERGEREC

Syntax:

MERGEREC

Description: Results in «MERGEREC». Use this in a mail merge to print the number of the corresponding merged data record in each resulting merged document. The number reflects the sequential order of the data records that were selected and possibly sorted for merging with the active main document. It does not indicate the actual order of the records as they occur in the physical data source. [*Note*: A personnel database might contain thousands of records. However, to send a form letter to employees who've reached their five-year anniversary with your company, you'd select as your data source only the records of those five-year employees, a much smaller set of records. To print a physical record number, you must include a record number field in the data source and insert the corresponding merge field in the main document. *end note*]

Field Value: «MERGEREC».

Switches: None.

[*Example*: The following example uses a MERGEREC field inside a formula to create unique invoice numbers. When the main document is merged with the data source, the number resulting from the MERGEREC field is added to the numbers representing the date and time the invoices are printed.

```
Invoice Number: = { PRINTDATE \@ "MMddyyyyHHmm" + MERGEREC }
```

The result might be:

```
Invoice Number for record 12, printed on Feb. 13, 2003 at 9:46:
  02132003094612
```

end example]

2.16.5.44 MERGESEQ

Syntax:

MERGESEQ

Description: Counts the number of data records that were successfully merged with the main document. Merged records are numbered starting from 1 each time documents are merged. [*Note*: The number might be different from the value inserted by the MERGEREC field. *end note*]

Field Value: The number of data records that were successfully merged with the main document.

Switches: None.

[*Example*: Consider the case in which only records 10–25 are merged. The MERGESEQ number corresponding to the first data record merged is 1, even though the MERGEREC number for that data record is 10. *end example*]

2.16.5.45 NEXT

Syntax:

NEXT

Description: Merges the next data record into the current resulting merged document, rather than starting a new merged document. [*Note*: This field is used when setting up a mailing label and envelope main document during a mail merge. *end note*]

Field Value: None.

Switches: None.

[*Example*: The following fields print three sets of names and phone numbers in each resulting merged document:

```
MERGEFIELD Name MERGEFIELD Phone
NEXT MERGEFIELD Name MERGEFIELD Phone
NEXT MERGEFIELD Name MERGEFIELD Phone
```

end example]

2.16.5.46 NEXTIF

Syntax:

NEXTIF *Expression-1 Operator Expression-2*

Expression-1:
expression

Expression-2:
expression

Description: Compares the values designated by *Expression-1* and *Expression-2* using the operator designated by *Operator*. If the comparison is true, the next data record is merged into the current merge document. (Merge fields that follow the NEXTIF in the main document are replaced by values from the next data record rather than the current data record.) If the comparison is false, the next data record is merged into a new merge document.

Operator can be any one of the six relational and equality operators specified for *operator* (§2.16.3.3).

A NEXTIF fields shall NOT be used in a footnote, an endnote, annotation, a header, a footer, or a data source. A NEXTIF field shall NOT be nested within any field.

Field Value: None.

Switches: None.

2.16.5.47 NOTEREF

Syntax:

NOTEREF *field-argument* [*switches*]

Description: Inserts the mark of the footnote or endnote that is marked by the bookmark specified by *text* in *field-argument*.

Field Value: The mark of the footnote or endnote.

Switches: Zero or more of the following *field-specific-switch*es.

\f	For a footnote, inserts the reference mark with the same character formatting as the Footnote Reference style. For an endnote, inserts the reference mark with the same character formatting as the Endnote Reference style.
\h	Inserts a hyperlink to the bookmarked endnote or footnote.
\p	Inserts the relative position of the footnote or endnote. If the NOTEREF field occurs before the bookmark, the result is "below". If the NOTEREF field occurs after the bookmark, the result is "above".

[*Example*: Consider the case in which a bookmark called `F10` marks the footnote of interest. When the field

```
… (see note { NOTEREF F10 }).
```

is updated, the result might be:

```
… (see note 5).
```

end example]

2.16.5.48 NUMCHARS

Syntax:

NUMCHARS [*switch*]

Description: Retrieves the number of characters in the current document, as recorded in the Characters element of the Application-Defined File Properties part.

Field Value: The number of characters in the current document.

Switches: Zero or one *numeric-formatting-switch* or *general-formatting-switch*.

[*Example*: Consider the case in the document has 6,183 words and the following fields are updated:

```
NUMCHARS
NUMCHARS \# #,##0
```

the results are:

```
6183
6,183
```

end example]

2.16.5.49 NUMPAGES

Syntax:

NUMPAGES [*switch*]

Description: Retrieves the number of pages in the current document, as recorded in the Pages element of the Application-Defined File Properties part.

Field Value: The number of pages in the current document.

Switches: Zero or one *numeric-formatting-switch* or *general-formatting-switch*.

[*Example*: Consider the case in the document has 19 pages and the following fields are updated:

```
NUMPAGES \# #,##0
NUMPAGES \* OrdText
```

the results are:

```
 19
nineteenth
```

end example]

2.16.5.50 NUMWORDS

Syntax:

NUMWORDS [*switch*]

Description: Retrieves the number of words in the current document, as recorded in the Words element of the Application-Defined File Properties part.

Field Value: The number of words in the current document.

Switches: Zero or one *numeric-formatting-switch* or *general-formatting-switch*.

[*Example*: Consider the case in the document has 1,243 words and the following fields are updated:

```
NUMWORDS
NUMWORDS \# #,##0
```

the results are:

```
1243
1,243
```

end example]

2.16.5.51 PAGE

Syntax:

PAGE [*switches*]

Description: Retrieves the number of the current page.

Field Value: The number of the current page.

Switches: Zero or more *general-formatting-switch*es.

[*Example*: When the current page number is 19 and the following fields are updated:

```
PAGE
PAGE \* ArabicDash
PAGE \* ALPHABETIC
PAGE \* roman
```

the results are:

```
19
- 19 -
S
xix
```

end example]

2.16.5.52 PAGEREF

Syntax:

PAGEREF *field-argument* [*switches*]

Description: Inserts the number of the page containing the bookmark specified by *text* in *field-argument* for a cross-reference.

Field Value: The number of the page containing the bookmark.

Switches: Zero or one of the *general-formatting-switch*es, zero or one of the *numeric-formatting-switch*es, and zero or more of the following *field-specific-switch*es.

\h	Creates a hyperlink to the bookmarked paragraph.
\p	Causes the field to display its position relative to the source bookmark. If the PAGEREF field is on the same page as the bookmark, it omits "on page #" and returns "above" or "below" only. If the PAGEREF field is not on the same page as the bookmark, the string "on page #" is used.

[*Example*: Consider the case in which a bookmark called `Worldpop1990` marks the table containing figures for 1990. When the field

```
The world population in 1991 was 5 billion; for 1990 figures,
see the table { PAGEREF Worldpop1990 \p }.
```

is updated, the position of the table is inserted in place of the field. The result is one of the following::

```
… see the table above.
… see the table below.
… see the table on page 27.
```

end example]

2.16.5.53 PRINT

Syntax:

PRINT *field-argument* [*switches*]

Description: Sends the printer-specific control code characters specified by *text* in *field-argument* to the selected printer.

Field Value: None.

Switches: Zero or more of the following *field-specific-switch*es.

\p *field-argument-1 field-argument-2*	Allows PostScript strings to be sent to the printer as native PostScript codes. PostScript commands embedded in the document are carried out in the order in which they are inserted. The y-coordinate space used for PostScript commands in as follows: The graphics origin (0,0) is in the lower-left corner of the page, and the positive directions are up and to the right. PostScript drawing instructions take place within a drawing rectangle. The graphics origin is translated to the lower-left corner of the drawing rectangle. *text* in this switch's *field-argument-1* defines the drawing rectangle on which the subsequent PostScript instructions operate. *text* in this switch's *field-argument-2* contains the PostScript instructions.

2.16.5.54 PRINTDATE

Syntax:

PRINTDATE [*switches*]

Description: Retrieves the date and time on which the document was last printed, as recorded in the LastPrinted element of the Core File Properties part. By default, the Gregorian calendar is used and the *date-and-time-formatting-switch* used is implementation-defined. For a document that has never been printed, the date and time corresponds to `0000-00-00T00:00:00` local time and each text component is XXX.

Field Value: The date and time on which the document was last printed.

Switches: Zero or one *date-and-time-formatting-switch* and zero or one of the following *field-specific-switch*es.

\h	Use the Hijri/Lunar calendar.
\s	Use the Saka Era calendar.

[*Example*: Consider the case in which the LastPrinted element is as follows:

```
<LastPrinted>2006-01-06T19:58:00Z</LastPrinted>
```

and the following fields are updated in a US-English context that is UTC -5:

```
PRINTDATE
PRINTDATE \@ "dddd, MMMM dd, yyyy  HH:mm:ss"
```

the results are:

```
1/6/2006 2:58:00 PM
Friday, January 06, 2006 14:58:00
```

For a document that has never been printed, the result is:

```
0/0/0000 0:00:00 AM
XXX, XXX 00, 0000  00:00:00
```

end example]

2.16.5.55 PRIVATE

Syntax:

PRIVATE

Description: Provides a private storage area. This field is used to store data for documents converted from other file formats. The field contains data needed for converting a document back to its original file format.

A PRIVATE field is formatted as hidden text.

Field Value: None.

Switches: None.

2.16.5.56 QUOTE

Syntax:

QUOTE *field-argument*

Description: Retrieves the text specified by *text* in *field-argument*. This text can include any other fields except AUTONUM, AUTONUMLGL, AUTONUMOUT, and SYMBOL.

Field Value: The specified text.

Switches: One or more of the *date-and-time-formatting-switch*, *general-formatting-switch*, or *date-and-time-formatting-switch*es, depending on the type of *field-argument*.

[*Example*: When the current month is January and the following field is updated:

```
Last month was QUOTE IF DATE \@ "M" = 1 "12" "= DATE \@ "M" - 1"/1/2000 \@
"MMMM".
```

the result is:

```
Last month was December.
```

end example]

2.16.5.57 RD

Syntax:

RD *field-argument* [*switch*]

Description: *field-argument* identifies a file to include when creating a table of contents, a table of authorities, or an index using a TOC (§2.16.5.75), TOA (§2.16.5.74), or INDEX field (§2.16.5.35). RD fields that reference a series of files must be in the same order as the files in the final document. If the location includes a long file name containing spaces, field-argument shall contain delimiting quotes. A single backslash in the file path shall be preceded directly by a backslash.

For a complex field implementation in XML the optional field-value storage is not needed.

Field Value: None.

Switches: One of the following *field-specific-switch*es:

\p	Indicates that the path is relative to the current document.

[*Example*: The following fields inserted into one document create a table of contents that includes entries from the three referenced documents:

```
{ TOC }
{ RD C:\\Manual\\Chapters\\Chapter1.doc }
{ RD C:\\Manual\\Chapters\\Chapter2.doc }
{ RD C:\\Manual\\Chapters\\Chapter3.doc }
```

end example]

2.16.5.58 REF

Syntax:

[REF] *field-argument* [*switches*]

Description: Inserts the text or graphics represented by the bookmark specified by *text* in *field-argument*. The bookmark shall be defined in the current document. Provided the bookmark name is not exactly the same as a field name, the REF prefix can be omitted. If the text marked by the bookmark contains a paragraph mark, the text preceding the REF field assumes the formatting of the paragraph in the bookmark.

Field Value: The specified text or graphics.

Switches: One of the following *general-formatting-switch*es: * Caps, * FirstCap, * Lower, or * Upper, and zero or one of the following *field-specific-switch*es.

\d *field-argument*	*text* in this switch's *field-argument* specifies the character sequence that is used to separate sequence numbers and page numbers.
\f	Increments footnote, endnote, and annotation numbers that are marked by the bookmark, and inserts the corresponding footnote,

	endnote, and comment text.
\h	Creates a hyperlink to the bookmarked paragraph.
\n	For a referenced paragraph, causes the field result to have the entire paragraph number without trailing periods. No information about prior levels is displayed unless it is included as part of the current level.
\p	Causes the field result to contain the position relative to the source bookmark using the word "above" or "below." If the REF field appears in the document before the bookmark, it evaluates to "below". If the REF field appears after the bookmark, it evaluates to "above". If the REF field appears within the bookmark, an error is returned. This switch can also be used in conjunction with the \n, \r, and \w switches, in which case, "above" or "below" is appended to the end of the field result.
\r	Inserts the entire paragraph number of the bookmarked paragraph in relative context—or relative to its position in the numbering scheme —without trailing periods.
\t	Causes the REF field to suppress non-delimiter or non-numerical text when used in conjunction with the \n, \r, or \w switch.
\w	Inserts the paragraph number of the bookmarked paragraph in full context from anywhere in the document.

[*Example*: The following field

```
REF _Ref116788778 \r \h
```

makes a hyperlink reference. This kind of field is commonly used, to indicate forward references within a document. *end example*]

2.16.5.59 REVNUM

Syntax:

REVNUM

Description: Retrieves the document's revision number (which indicates the number of times the document has been saved), as recorded in the Revision element of the Core File Properties part.

Field Value: The document's revision number.

Switches: None.

[*Example*: Consider the case in which the Revision element is as follows:

```
<Revision>11</Revision>
```

and the following field is updated:

```
REVNUM
```

The result is:

```
11
```

end example]

2.16.5.60 SAVEDATE

Syntax:

SAVEDATE [*switches*]

Description: Retrieves the date and time on which the document was last saved, as recorded in the DateModified element of the Core File Properties part. By default, the Gregorian calendar is used and the *date-and-time-formatting-switch* used is implementation-defined. For a document that has never been saved, the date and time corresponds to `0000-00-00T00:00:00` local time and each text component is XXX.

Field Value: The date and time on which the document was last saved.

Switches: Zero or one *date-and-time-formatting-switch* and zero or one of the following *field-specific-switch*es.

\h	Use the Hijri/Lunar calendar.
\s	Use the Saka Era calendar.

[*Example*: Consider the case in which the DateModified element is as follows:

```
<DateModified>2006-01-06T20:15:00Z</DateModified>
```

and the following fields are updated in a US-English context that is UTC -5:

```
SAVEDATE
SAVEDATE \@ "dddd, MMMM dd, yyyy  HH:mm:ss"
```

the results are:

```
1/6/2006 3:15:00 PM
Friday, January 06, 2006 15:15:00
```

For a document that has never been saved, the result is:

```
0/0/0000 0:00:00 AM
XXX, XXX 00, 0000  00:00:00
```

end example]

2.16.5.61 SECTION

Syntax:

SECTION [*switches*]

Description: Retrieves the number of the current section.

Field Value: The number of the current section.

Switches: Zero or more *general-formatting-switch*es.

[*Example*: When the current section number is 19 and the following fields are updated:

```
SECTION
SECTION \* ArabicDash
SECTION \* ALPHABETIC
SECTION \* roman
```

the results are:

```
19
- 19 -
S
xix
```

end example]

2.16.5.62 SECTIONPAGES

Syntax:

SECTIONPAGES [*switches*]

Description: Retrieves the number of the current page within the current section.

Field Value: The number of the current page within the current section.

Switches: Zero or more *general-formatting-switch*es.

[*Example*: When the current page number within the current section is 19 and the following fields are updated:

```
SECTIONPAGES
SECTIONPAGES \* ArabicDash
SECTIONPAGES \* ALPHABETIC
SECTIONPAGES \* roman
```

the results are:

```
19
- 19 -
S
xix
```

end example]

2.16.5.63 SEQ

Syntax:

SEQ *identifier* [*field-argument*] [*switches*]

Description: Sequentially numbers chapters, tables, figures, and other user-defined lists of items in a document. If an item and its SEQ field are added, deleted, or moved, updating the remaining SEQ fields in the document reflects the new sequence. A SEQ field in a header, footer, annotation, or footnote shall NOT affect the sequence numbering that results from SEQ fields in the document text.

[*Note*: The LISTNUM field also produces automatic numbering and may be a better alternative when creating a complex numbered list. *end note*]

identifier is the name assigned to the series of items that are to be numbered. [*Example*: *identifier* might be Equation, Figure, Table, or Thing, as the user deems appropriate for a caption. *end example*] *identifier* shall start with a Latin letter and shall consist of no more than 40 Latin letters, Arabic digits, and underscores. (See the TOC field (§2.16.5.75) switches \c and \s for uses of *identifier*.)

text in *field-argument* specifies a bookmark name that refers to an item elsewhere in the document rather than in the current location.

Field Value: The next number in the sequence.

Switches: Zero or one of the *numeric-formatting-switch*es, or zero or more of the following *field-specific-switch*es. If no *numeric-formatting-switch* is present, * Arabic is used.

\c	Repeats the closest preceding sequence number. [*Note*: This is useful for inserting chapter numbers in headers or footers. *end note*]
\h	Hides the field result unless a *general-formatting-switch* is also present.[*Note*: This switch can be used to refer to a SEQ field in a cross-reference without printing the number. *end note*]
\n	Inserts the next sequence number for the specified item. This is the default.
\r *field-argument*	Resets the sequence number to the integer number specified by *text* in this switch's *field-argument*.
\s *field-argument*	Resets the sequence number to the built-in (integer) heading level specified by *text* in this switch's *field-argument*.

[*Example*: When the following fields are updated:

```
SEQ Figure
SEQ Figure \* roman
SEQ Figure \n
SEQ Figure \c : …
SEQ Figure \h : …
SEQ Figure
SEQ Figure \r 1
SEQ Figure
```

the results are:

```
Figure 1
Figure ii
Figure 3
Figure 3: …
Figure : …
Figure 5
Figure 10
Figure 11
```

end example]

2.16.5.64 SET

Syntax:

SET *field-argument-1 field-argument-2*

field-argument-1:
field-argument

field-argument-2:
field-argument

Description: Defines the bookmark name specified by *field-argument-1* to represent the information specified by *field-argument-2*.

Field Value: None.

Switches: None.

[*Example*: Consider the following:

```
SET EnteredBy "Paul Smith"
SET UnitCost 25.00
SET Quantity FILLIN "Enter number of items ordered:"
SET SalesTax 10%
```

```
SET TotalCost = (UnitCost * Quantity) + ((UnitCost * Quantity) * SalesTax)
Total cost: TotalCost \# "$#0.00"
Thank you for your order,
EnteredBy
```

end example]

2.16.5.65 SKIPIF

Syntax:

SKIP *Expression-1 Operator Expression-2*

Expression-1:
expression

Expression-2:
expression

Description: Compares the values designated by *Expression-1* and *Expression-2* using the operator designated by *Operator*. If the comparison is true, SKIPIF cancels the current merge document, moves to the next in the data source, and starts a new merge document. If the comparison is false, the current merge document is continued.

Operator can be any one of the six relational and equality operators specified for *operator* (§2.16.3.3).

Field Value: None.

Switches: None.

[*Example*: Inserted into a mail merge main document, the following field examines the contents of the Order field for the current data record. If the field contains a number less than 100, no merged document is produced for that data record.

```
SKIPIF MERGEFIELD Order < 100
```

end example]

2.16.5.66 STYLEREF

Syntax:

STYLEREF *field-argument* [*switches*]

Description: Inserts the nearest piece of text prior to this field that is formatted by the style whose name is specified by *text* in *field-argument*. The style can be a paragraph style or a character style.

When this field is used in a header or footer, it results in the first or the last text formatted with the specified style on the current page, allowing for dictionary-style headers or footers.

The location at which a STYLEREF field is inserted determines the direction searched for the style, as follows:

- In document text, by default, the search goes backward from the STYLEREF field. If the style isn't found, the search goes forward from the STYLEREF field.
- In footnotes, annotations, and endnotes, the search goes backward from the footnote, annotation, or endnote reference mark. If the style isn't found, the search goes forward from the reference mark.
- In headers and footers in a printed document, the search is applied to the current page, by default, from top to bottom, for the specified style. If the style isn't found, the search goes from the top of the page to the beginning of the document, and then from the bottom of the page to the end of the document. If the \l switch is used, the search goes from the bottom of the page to the beginning and then to the end of the document.
- In headers and footers in an electronic document, the search goes on in the section that contains the STYLEREF field, from the beginning, for the specified style. If the style isn't found, the search goes from the end of the section to the end of the document.

Field Value: The nearest piece of text prior to this field that is formatted by the style whose name is specified by *text* in *field-argument*.

Switches: Zero or more of the following *field-specific-switch*es.

\l	Inserts the nearest text following the field.
\n	Inserts the paragraph number of the referenced paragraph exactly as it appears in the document.
\p	Inserts the relative position of the referenced paragraph as being "above" or "below".
\r	Inserts the paragraph number of the referenced paragraph exactly in relative context.
\t	When used with the \n, \r, or \w switch, causes non-delimiter and non-numerical text to be suppressed.
\w	Inserts the paragraph number of the referenced paragraph in full context, from anywhere in the document.

[*Example*: When the following field is inserted in a header, it displays the contents of the first paragraph formatted with the style "Heading 3" on the current page:

On this page: { STYLEREF "Heading 3" }

To print the first and last names that appear on each page in a membership directory, for example, apply a character style called Last Name to each member's last name. Then insert the following fields in the header:

{ STYLEREF "Last Name" } — { STYLEREF "Last Name" \l }

end example]

2.16.5.67 SUBJECT

Syntax:

SUBJECT [*field-argument*] [*switch*]

Description: Retrieves, and optionally sets, the document's subject, as recorded in the Subject element of the Core File Properties part or, if *field-argument* is present, the subject specified by *text* in *field-argument*. Specifying a *field-argument* shall change Subject to *text*.

Field Value: The document's subject.

Switches: One of the following *general-formatting-switch*es: `\* Caps`, `\* FirstCap`, `\* Lower`, or `\* Upper`.

[*Example*: Consider the case in which the Title element is as follows:

```
<Subject>A specification for fields</Subject>
```

and the following field is updated:

SUBJECT

The result is:

```
A specification for fields
```

Updating the following field:

```
SUBJECT "A specification for WordprocessingML Fields"
```

causes the Subject element to take on the given value. *end example*]

2.16.5.68 SYMBOL

Syntax:

SYMBOL *field-argument* [*switches*]

Description: Retrieves the character whose code point value is specified in decimal or hexadecimal (by using a leading 0x or 0X) by *text* in *field-argument*. The formatting switches over ride any formatting applied directly to the result.

The XML generated for a complex field implementation shall not have the optional field value stored.

Field Value: The specified character.

Switches: Zero or more of the following *field-specific-switch*es.

`\a`	Interprets *text* in *field-argument* as the value of an ANSI character.
`\f` *field-argument*	Interprets *text* in the switch's *field-argument* as the name of the font from

	which the character whose value is specified by *text* in the field's *field-argument*. By default, the font used is that for the current text run.
\h	Inserts the symbol without affecting the line spacing of the paragraph. If large symbols are inserted with this switch, text above the symbol may be overwritten.
\j	Interprets *text* in *field-argument* as the value of a SHIFT-JIS character.
\s *field-argument*	Interprets *text* in the switch's *field-argument* as the integral font size in points.
\u	Interprets *text* in *field-argument* as the value of a Unicode character.

[*Example*: Consider the case in which the following fields are updated:

```
SYMBOL 65
SYMBOL 66 \a
SYMBOL 67 \u
SYMBOL 0x20ac \u
SYMBOL 68
SYMBOL 68 \f Symbol
SYMBOL 40 \f Wingdings \s 24
```

the results are:

A
B
C
€
D
Δ
☎

end example]

2.16.5.69 TA

Syntax:

TA [*switches*]

Description: Defines the text and page number for a table of authorities entry, which is used by a TOA field (§2.16.5.74).

Field Value: None.

Switches: Zero or one of the following *field-specific-switch*es.

\b	Applies bold formatting to the page number for the entry. If the table of authorities style for the entry already has bold formatting, \b removes it.
\c *field-argument*	*text* in this switch's *field-argument* specifies the integral entry category, which is a number that corresponds to the order of categories. The number determines how citations are grouped in tables of authorities. If \c is omitted, category 1 is the default.
\i	Applies italic formatting to the page number for the entry. If the table of authorities' style for the entry already has italic formatting, \i removes it.
\l *field-argument*	*text* in this switch's *field-argument* defines the long citation for the entry.
\r *field-argument*	Inserts as the entry's page number the range of pages marked by the bookmark specified by *text* in this switch's *field-argument*.
\s *field-argument*	*text* in this switch's *field-argument* defines the short citation for the entry.

[*Example*: Given the following fields occurring on page 2:

```
TA \l "Hotels v. Leisure Time" \c 2
TA \l "Baldwin v. Alberti, 58 Wn. 2d 243 (1961)"
   \s "Baldwin v. Alberti" \c 1 \b
```

the table of authorities produced by `INDEX \e "tab" \c "1" \z "1033"` is:

Cases

Baldwin v. Alberti, 58 Wn. 2d 243 (1961)…….. 2

Statutes

Hotels v. Leisure Time…… 2

end example]

2.16.5.70 TC

Syntax:

TC *field-argument* [*switches*]

Description: Defines the text and page number for a table of contents (including a table of figures) entry, which is used by a TOC field (§2.16.5.75). The text of the entry is *text* in *field-argument*.

Field Value: None.

Switches: Zero or one of the following *field-specific-switch*es.

`\f` *field-argument*	The type of items collected in a particular contents list. Use a unique Type identifier (typically a letter from A-Z) for each type of list. For example, to build a list of illustrations, mark each illustration with a field such as `TC "Illustration 1" \f i`, where `i` indicates only illustration entries. If no type is specified, the entry is listed in a table of contents.
`\l` *field-argument*	The level of the TC entry. [*Example*: The field `TC "Entering Data" \l 4` marks a level-4 entry, and applies the built-in style TOC 4 to that entry in the table of contents. *end example*] If no level is specified, level 1 is assumed.
`\n`	Omits the page number for the entry.

2.16.5.71 TEMPLATE

Syntax:

TEMPLATE [*switch*]

Description: Retrieves the disk file name of the template used by the current document.

Field Value: The disk file name of the template used by the current document.

Switches: One of the following *general-formatting-switch*es: `\* Caps`, `\* FirstCap`, `\* Lower`, or `\* Upper`, and zero or one of the following *field-specific-switch*es.

`\p`	Include the full file path name.

[*Example*: Consider the case in which the following fields are updated:

```
TEMPLATE \* Upper
TEMPLATE \p
```

the results might be:

```
NORMAL.DOTM
C:\Templates\Normal.dotm
```

end example]

2.16.5.72 TIME

Syntax:

TIME [*switch*]

Description: Retrieves the current date and time. The Gregorian calendar is always used. By default, the *date-and-time-formatting-switch* used is implementation-defined.

Field Value: The current date and time.

Switches: Zero or one *date-and-time-formatting-switch*.

[*Example*: Consider the case in which the following fields are updated on the given date and time:

```
TIME
TIME \@ "dddd, MMMM dd, yyyy  HH:mm:ss"
```

the results are:

```
1:59 PM
Friday, January 06, 2006  13:59:421/5/2006
```

end example]

[*Note*: For some *date-and-time-formatting-switch*es, the DATE (§2.16.5.18) and TIME fields can produce the same result. *end note*]

2.16.5.73 TITLE

Syntax:

TITLE [*field-argument*] [*switch*]

Description: Retrieves, and optionally sets, the document's title, as recorded in the Title element of the Core File Properties part or, if *field-argument* is present, the name specified by *text* in *field-argument*. Specifying a *field-argument* shall change Title to *text*.

Field Value: The document's title.

Switches: One of the following *general-formatting-switch*es: * Caps, * FirstCap, * Lower, or * Upper.

[*Example*: Consider the case in which the Title element is as follows:

```
<Title>My Life's Story</Title>
```

and the following field is updated:

TITLE

The result is:

```
My Life's Story
```

Updating the following field:

```
TITLE "My Life, the Fantasy" \* Upper
```

causes the Title element to take on the value My Life, the Fantasy while the result is:

MY LIFE THE FANTASY

end example]

2.16.5.74 TOA

Syntax:

TOA [*switches*]

Description: Builds a table of authorities (that is, a list of the references in a legal document, such as references to cases, statutes, and rules, along with the numbers of the pages on which the references appear) using the entries specified by TA fields (§2.16.5.69).

Field Value: The table of authorities.

Switches: Zero or more of the following *field-specific-switch*es.

\b *field-argument*	Includes entries only from the portion of the document marked by the bookmark specified by *text* in this switch's *field-argument*.
\c *field-argument*	*Includes the entries whose integral category is that specified by text* in this switch's *field-argument*.
\d *field-argument*	Used in conjunction with \s to specify the character sequence that separates the sequence numbers and page numbers. If \d is omitted, a hyphen (-) is used.
\e *field-argument*	*text* in this switch's *field-argument* specifies the character sequence that separates a table of authorities entry and its page number. If \e is not specified, a tab stop with leader dots is used.
\f	Removes the formatting of the entry text in the document from the entry in the table of authorities.
\g *field-argument*	*text* in this switch's *field-argument* specifies the character sequence that separates the pages in a page range. If \g is omitted, an en dash (–) is used.
\h	Includes the category heading for the entries in a table of authorities.
\l *field-argument*	*text* in this switch's *field-argument* specifies the character sequence that separates multiple page references. If \l is omitted, a comma (,) and space are used.
\p	Replaces five or more different page references to the same authority with "passim", which is used to indicate that a word or passage occurs frequently in the work cited.
\s *field-argument*	Includes a case or section number before the page number. The entry shall be numbered with a SEQ field (§2.16.5.63), and *text* in this switch's *field-argument* shall match the identifier in the SEQ field.

[*Example*: See TA (§2.16.5.69). *end example*]

2.16.5.75 TOC

Syntax:

TOC [*switches*]

Description: Builds a table of contents (which can also be a table of figures) using the entries specified by TC fields (§2.16.5.70), their heading levels, and specified styles, and inserts that table at this place in the document. Each table entry is a separate paragraph.

Field Value: The table of contents.

Switches: Zero or more of the following *field-specific-switch*es.

\a *field-argument*	Includes captioned items, but omits caption labels and numbers. The identifier designated by *text* in this switch's *field-argument* corresponds to the caption label. Use \c to build a table of captions with labels and numbers.
\b *field-argument*	Includes entries only from the portion of the document marked by the bookmark named by *text* in this switch's *field-argument*.
\c *field-argument*	Includes figures, tables, charts, and other items that are numbered by a SEQ field (§2.16.5.63). The sequence identifier designated by *text* in this switch's *field-argument*, which corresponds to the caption label, shall match the identifier in the corresponding SEQ field.
\d *field-argument*	When used with \s, the *text* in this switch's *field-argument* defines the separator between sequence and page numbers. The default separator is a hyphen (-).
\f *field-argument*	Includes only those TC fields whose identifier exactly matches the *text* in this switch's *field-argument* (which is typically a letter).
\h	Makes the table of contents entries hyperlinks.
\l *field-argument*	Includes TC fields that assign entries to one of the levels specified by *text* in this switch's *field-argument* as a range having the form *startLevel*-*endLevel*, where *startLevel* and *endLevel* are integers, and *startLevel* has a value equal-to or less-than *endLevel*. TC fields that assign entries to lower levels are skipped.
\n *field-argument*	Without *field-argument*, omits page numbers from the table of contents. Page numbers are omitted from all levels unless a range of entry levels is specified by *text* in this switch's *field-argument*. A range is specified as for \l.
\o *field-argument*	Uses paragraphs formatted with all or the specified range of built-in heading styles. Headings in a style range are specified by *text* in this switch's *field-argument* using the notation specified as for \l, where each integer corresponds to the style with a style ID of

	HeadingX (e.g. 1 corresponds to Heading1). If no heading range is specified, all heading levels used in the document are listed.
\p *field-argument*	*text* in this switch's *field-argument* specifies a sequence of characters that separate an entry and its page number. The default is a tab with leader dots.
\s *field-argument*	For entries numbered with a SEQ field (§2.16.5.63), adds a prefix to the page number. The prefix depends on the type of entry. *text* in this switch's *field-argument* shall match the identifier in the SEQ field.
\t *field-argument*	Uses paragraphs formatted with styles other than the built-in heading styles. *text* in this switch's *field-argument* specifies those styles as a set of comma-separated doublets, with each doublet being a comma-separated set of style name and table of content level. \t can be combined with \o.
\u	Uses the applied paragraph outline level.
\w	Preserves tab entries within table entries.
\x	Preserves newline characters within table entries.
\z	Hides tab leader and page numbers in Web layout view.

[*Example*: The index produced using the corresponding set of index entries and the field TOC \o "3-3" \h \z \t "Heading 1,1,Heading 2,2,Appendix 1,1,Appendix 2,2,Unnumbered Heading,1" is:

1. Introduction …… 1

2. Syntax…….. 2

3. XML representation…….. 4

4. Formulas and expressions…… 6

4.1 Constants…… 6

4.2 Bookmarks…….. 6

4.3 Operators…….. 6

4.4 Functions…… 7

4.5 Table cell references…….. 8

…

Annex A. Index…… 12

end example]

2.16.5.76 USERADDRESS

Syntax:

USERADDRESS [*field-argument*] [*switch*]

Description: Retrieves the current user's postal address or, if *field-argument* is present, the address specified by *text* in *field-argument*. Specifying a field-argument shall not change the address of the current user.

Field Value: A postal address.

Switches: One of the following *general-formatting-switch*es: `\* Caps`, `\* FirstCap`, `\* Lower`, or `\* Upper`.

[*Example*: Given the current user's address, the following fields:

```
USERADDRESS
USERADDRESS "10 Top Secret Lane, Chiswick" \* Upper
```

produce results of:

```
114 Rue du Rhône
CH-1204 Geneva
Switzerland

10 TOP SECRET LANE, CHISWICK
```

end example]

2.16.5.77 USERINITIALS

Syntax:

USERINITIALS [*field-argument*] [*switch*]

Description: Retrieves the current user's initials or, if *field-argument* is present, the initials specified by *text* in *field-argument*. Specifying a field-argument shall not change the initials of the current user.

Field Value: The set of initials.

Switches: One of the following *general-formatting-switch*es: `\* Caps`, `\* FirstCap`, `\* Lower`, or `\* Upper`.

[*Example*: Given a current user with initials "DW", the following fields:

```
USERNAME \* Lower
USERNAME "JaJ"
USERNAME "jaj" \* Upper
```

produce results of:

```
dw
JaJ
JAJ
```

end example]

2.16.5.78 USERNAME

Syntax:

USERNAME [*field-argument*] [*switch*]

Description: Retrieves the current user's name or, if *field-argument* is present, the name specified by *text* in *field-argument*. Specifying a field-argument shall not change the name of the current user.

Field Value: The name.

Switches: One of the following *general-formatting-switch*es: `\* Caps`, `\* FirstCap`, `\* Lower`, or `\* Upper`.

[*Example*: Given a current user of "David Williams", the following fields:

```
USERNAME \* Lower
USERNAME "John Jones"
USERNAME "Mary Smith" \* Upper
```

produce results of:

```
david williams
John Jones
MARY SMITH
```

end example]

2.16.5.79 XE

Syntax:

XE *field-argument* [*switches*]

Description: Defines the text and page number for an index entry, which is used by an INDEX field (§2.16.5.35). The text of the entry is *text* in *field-argument*. To indicate a subentry, the main entry text and the subentry text shall be separated by a colon (:). Subentries beyond one level are permitted.

Field Value: None.

Switches: Zero or one of the following *field-specific-switch*es.

\b	Applies bold formatting to the entry's page number. However, if the index style for that entry is already bold, this switch removes

	that formatting for that entry.
\f *field-argument*	The *text* in this switch's *field-argument* defines an index entry type. If an INDEX field has the same \f switch and *field-argument*, this entry is included in the resulting index; otherwise, it is excluded.
\i	Applies italic formatting to the entry's page number. However, if the index style for that entry is already italic, this switch removes that formatting for that entry.
\r *field-argument*	Instead of the entry's page number, uses the range of pages marked by the bookmark specified by *text* in this switch's *field-argument*.
\t *field-argument*	Uses *text* from *field-argument* in place of a page number. [*Note*: Useful for "See …" or "See also …" entries. *end note*]
\y *field-argument*	Specifies that the *text* from *field-argument* defines the yomi (first phonetic character for sorting indexes) for the index entry.

[*Example*: Given the following fields spread over a series of pages, and a multi-page bookmark called OOXMLPageRange:

```
XE "Office Open XML" \b
XE "syntax" \f "Introduction"
XE "behavior:implementation-defined" \b
XE "Office Open XML" \i
XE "behavior:implementation-defined:documenting" \b
XE "grammar" \f "Introduction" \b
XE "Office Open XML"
XE "item: package-relationship" \t "See package-relationship item"
XE "XML" \r OOXMLPageRange
XE "grammar" \f "Introduction"
XE "production" \f "Introduction"
```

the index produced by INDEX \e "*tab*" \c "1" \z "1033" is:

behavior

implementation-defined ……. **2**

documenting …….. **3**

item

package-relationship …….. See package-relationship item

Office Open XML …… **2**, *3*, 4

end example]

2.16.6 calcOnExit (Recalculate Fields When Current Field Is Modified)

This element specifies that the current contents of all fields within the current WordprocessingML document shall be recalculated from their field codes when the contents of the parent form field are modified. [*Note*: It is at the discretion of an application to determine the scope of a single modification, for example, when the user moves the insertion point in a user interface, or after each keystroke, etc. *end note*]

If this element is omitted, then modification of the contents of the current field shall not result in all fields in the current document being recalculated.

[*Example*: Consider the following WordprocessingML fragment for the contents of two fields in a document:

```
<w:bookmarkStart w:name="Text1" … />
<w:fldSimple w:instr="FORMFIELDTEXT">
  <w:ffData>
    <w:calcOnExit/>
    …
  </w:ffData>
  <w:r>
    <w:t>1</w:t>
  </w:r>
</w:fldSimple>
<w:bookmarkEnd w:name="Text1" … />
<w:fldSimple w:instr="=Text1+10">
  <w:r>
    <w:t>11</w:t>
  </w:r>
</w:fldSimple>
```

The first field above (the text form field) has a current value of `1`, but also has the calcOnExit element present (therefore inheriting its default attribute value of `true`). This means that if the value of this form field is changed to `10`, that all fields in the document shall automatically be updated, resulting in the second field's value being automatically changed to `20`. *end example*]

Parent Elements
ffData (§2.16.17)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.16.7 checkBox (Checkbox Form Field Properties)

This element specifies a set of properties which shall be associated with the parent FORMCHECKBOX checkbox form field (§2.16.5.26) within the document.

If the parent form field is not a checkbox (i.e. its field code does not have a value of FORMCHECKBOX), then these properties may be ignored.

[*Example*: Consider the following WordprocessingML fragment for the properties of a checkbox form field:

```
<w:ffData>
  <w:checkBox>
    <w:size w:val="20" />
    <w:checked w:val="true" />
  </w:checkBox>
</w:ffData>
```

The checkBox element specifies that it contains a set of properties for the parent checkbox form field. In this case, these properties specify that the size of the checkbox shall be exactly 10 points via the size element

(§2.16.30), and that the current state of the checkbox shall be checked via the checked element (§2.16.8). *end example*]

Parent Elements
ffData (§2.16.17)

Child Elements	Subclause
checked (Checkbox Form Field State)	§2.16.8
default (Default Checkbox Form Field State)	§2.16.12
size (Checkbox Form Field Size)	§2.16.30
sizeAuto (Automatically Size Form Field)	§2.16.31

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FFCheckBox">
   <sequence>
      <choice>
         <element name="size" type="CT_HpsMeasure"/>
         <element name="sizeAuto" type="CT_OnOff"/>
      </choice>
      <element name="default" type="CT_OnOff" minOccurs="0"/>
      <element name="checked" type="CT_OnOff" minOccurs="0"/>
   </sequence>
</complexType>
```

2.16.8 checked (Checkbox Form Field State)

This element specifies the current state for a checkbox form field. This value shall be used to specify the current value for a checkbox as explicitly chosen for that checkbox, as opposed its default value, which is specified using the default element (§2.16.12).

If this element is omitted, then the parent form field checkbox has no state, and its state shall be determined based on the value of the default element in the checkbox form field properties.

[*Example*: Consider the following WordprocessingML fragment for the properties of a checkbox form field:

```
<w:ffData>
  <w:checkBox>
    <w:checked w:val="true" />
  </w:checkBox>
</w:ffData>
```

The checked element specifies that the current state of the checkbox is checked (via an attribute value of `true`). *end example*]

Parent Elements

Parent Elements
checkBox (§2.16.7)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.16.9 ddList (Drop-Down List Form Field Properties)

This element specifies a set of properties which shall be associated with the parent FORMDROPDOWN drop-down list form field (§2.16.5.27) within the document.

If the parent form field is not a drop-down list (i.e. its field code does not have a value of FORMDROPDOWN), then these properties may be ignored.

[*Example*: Consider the following WordprocessingML fragment for the properties of a drop-down list form field:

```
<w:ffData>
  <w:ddList>
    <w:listEntry w:val="One" />
    <w:listEntry w:val="Two" />
    <w:listEntry w:val="Three" />
  </w:ddList>
</w:ffData>
```

The ddList element specifies that it contains a set of properties for the parent drop-down list form field. In this case, these properties specify that the drop-down list shall contain three entries of `One`, `Two`, and `Three` via the listEntry elements (§2.16.26). *end example*]

Parent Elements
ffData (§2.16.17)

Child Elements	Subclause
default (Default Drop-Down List Item Index)	§2.16.11
listEntry (Drop-Down List Entry)	§2.16.26
result (Drop-Down List Selection)	§2.16.29

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FFDDList">
   <sequence>
      <element name="result" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="default" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="listEntry" type="CT_String" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

2.16.10 default (Default Text Box Form Field String)

This element specifies the default string for the parent text box form field. This string is the content which shall be displayed in the document story within this form field if its current run contents are empty (i.e. there is not actual content within the text box). If the type (§2.16.34) of the current form field is `calculation`, then this string shall hold the calculation to be performed.

If this element is omitted, then the current text box form field shall not have a default value.

[*Example*: Consider the following WordprocessingML fragment for a text box form field:

```
<w:fldSimple w:instr="FORMTEXT">
  <w:ffData>
    <w:textInput>
      <w:default w:val="No content."/>
    </w:textInput>
  </w:ffData>
</w:fldSimple>
```

The default element specifies the default value of the text box form field to be `No content.` Since the form field does not contain any value, this is the content which shall be displayed when the contents of the form field are displayed by an application. *end example*]

Parent Elements
textInput (§2.16.33)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.16.11 default (Default Drop-Down List Item Index)

This element specifies the zero-based index of the default entry for the parent drop-down list form field. This index value is the value within the drop-down list which shall be displayed in the document story within this form field if no element is selected (i.e. the result element (§2.16.29) is omitted).

If this element is omitted, then the current drop-down list form field shall have a default value of `0` (its first entry). If the attribute value references an index value which does not exist (i.e. a negative number or a number that exceeds the number of items in the drop-down list), then this value may be ignored and the current drop-down list form field shall have a default value of `0` (its first entry).

[*Example*: Consider the following WordprocessingML fragment for a drop-down list form field:

```
<w:fldSimple w:instr="FORMDROPDOWN">
  <w:ffData>
    <w:ddList>
      <w:default w:val="1" />
      <w:listEntry w:val="One" />
      <w:listEntry w:val="Two" />
      <w:listEntry w:val="Three" />
    </w:ddList>
  </w:ffData>
</w:fldSimple>
```

The default element specifies the index of the default value of the drop-down list form field to be 1. Since the form field does not contain a result element, this is the index of the content which shall be displayed when the contents of the form field are displayed by an application. In this case, the resulting default value text is Two. *end example*]

Parent Elements
ddList (§2.16.9)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.16.12 default (Default Checkbox Form Field State)

This element specifies the default checkbox state for the parent checkbox form field. This value determines the checkbox state when its current run contents are empty (i.e. there is not actual content within the drop-down list).

If this element is omitted, then the current checkbox form field shall have a default value of `0` (unchecked).

[*Example*: Consider the following WordprocessingML fragment for a checkbox list form field:

```
<w:fldSimple w:instr="FORMDROPDOWN">
  <w:ffData>
    <w:checkBox>
      <w:default w:val="true" />
    </w:checkBox>
  </w:ffData>
</w:fldSimple>
```

The default element specifies the default state of the checkbox form field to be `true`. Since the form field does not contain any run content, this is the state which shall be displayed when the contents of the form field are displayed by an application. *end example*]

Parent Elements
checkBox (§2.16.7)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned `off`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.16.13 delInstrText (Deleted Field Code)

This element specifies that this run contains deleted field codes (§2.16.5) within a complex field in the document. The delInstrText element shall be used for all runs containing field codes which are part of a region of text that is contained in a deleted region using the del element (§2.13.5.12).

If this element is not contained within a del element, then the document is invalid. If this element is contained within a run which is not part of a complex field's field codes, then it should be handled as regular deleted text.

[*Example*: Consider a complex field within a WordprocessingML document which was changed from a text box form field to a checkbox form field with revision tracking enabled. This field would therefore be represented as follows:

```
<w:fldchar w:type="begin" />
<w:ins>
  <w:r>
    <w:instrText>FORMCHECKBOX</w:instrText>
  </w:r>
  <w:del>
    <w:r>
      <w:delInstrText>FORMFIELDTEXT</w:delInstrText>
    </w:r>
  </w:del>
<w:fldChar w:type="seperate" />
…
<w:fldChar w:type="end" />
```

The deleted field code is contained in a delInstrText node, while the inserted (and current) field code is contained in an instrText node. *end example*]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

Attributes	Description
space (Content Contains Significant Whitespace) Namespace:	Specifies how white space should be handled for the contents of this element using the W3C space preservation rules. [*Example*: Consider the following run contained within a WordprocessingML document:

Attributes	Description
http://www.w3.org/XML/1998/namespace	<w:r> <w:t>significant whitespace </w:t> </w:r> Although there are three spaces on each side of the text content in the run, that whitespace has not been specifically marked as significant, therefore it is removed when this run is added to the document. *end example*] The possible values for this attribute are defined by the type in the namespace.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Text">
   <simpleContent>
      <extension base="ST_String">
         <attribute ref="xml:space" use="optional"/>
      </extension>
   </simpleContent>
</complexType>
```

2.16.14 enabled (Form Field Enabled)

This element specifies whether the parent form field shall behave as though it is enabled or disabled when it is displayed in the document. This setting shall have no effect on the behavior of this form field unless the document's Settings part specifies that the documentProtection element for the current document is in a state allowing the filling in of form fields.

If this element is omitted, then the parent form field shall be in its enabled state when the document settings specify that the document allows the filling-in of form fields.

[*Example*: Consider the following WordprocessingML fragment for a text box form field:

```
<w:fldSimple w:instr="FORMTEXT">
  <w:ffData>
    <w:enabled w:val="false"/>
    <w:textInput>
      …
    </w:textInput>
  </w:ffData>
</w:fldSimple>
```

The enabled element specifies that the state of the current text box form field is disabled; therefore this text box shall not be editable within the current document even when the state of the documentProtection element specifically allows the editing of form fields. *end example*]

Parent Elements

Parent Elements
ffData (§2.16.17)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.16.15 entryMacro (Script Function to Execute on Form Field Entry)

This element specifies a subroutine in a scripting language which should be executed when the when the run contents of the parent form field are entered. The language and location of this subroutine may be determined using any method desired by an application. [*Note*: It is at the discretion of an application to determine the scope and timing of "entering" a form field, for example, when the user moves the insertion point in a user interface or upon each operation by an application without a user interface, etc. *end note*]

If this element is omitted, then no subroutine shall be associated with entering the run contents of the parent form field. If this element specifies a macro which cannot be located or is not supported by an application, then its value may be ignored, but shall not be lost upon resaving the file.

[*Example*: Consider the following WordprocessingML fragment for the properties of a checkbox form field:

```
<w:ffData>
  <w:entryMacro w:val="TestEntryFunction" />
  <w:checkBox>
    …
  </w:checkBox>
</w:ffData>
```

The entryMacro element specifies that any application which processes this file should attempt to locate and execute a scripting subroutine called `TestEntryFunction` when the contents of the checkbox are entered. If this subroutine cannot be located or executed, then this setting is silently ignored. *end example*]

Parent Elements
ffData (§2.16.17)

Attributes	Description

val (Name of Script Function)	Specifies the name of a single scripting subroutine which shall be associated with the parent element. Its use is specifies based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment for the properties of a form field:
	
	`<w:ffData>`
	`  <w:exitMacro w:val="HelloWorld" />`
	`</w:ffData>`
	
	The val attribute specifies that a script function called `HelloWorld` shall be used in the context of the parent element; in this case, to execute when the field is exited. *end example*]
	
	The possible values for this attribute are defined by the ST_MacroName simple type (§2.18.58).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MacroName">
   <attribute name="val" use="required" type="ST_MacroName"/>
</complexType>
```

2.16.16 exitMacro (Script Function to Execute on Form Field Exit)

This element specifies a subroutine in a scripting language which should be executed when the when the run contents of the parent form field are exited. The language and location of this subroutine may be determined using any method desired by an application. [*Note*: It is at the discretion of an application to determine the scope and timing of "exiting" a form field, for example, when the user moves the insertion point in a user interface or upon each operation by an application without a user interface, etc. *end note*]

If this element is omitted, then no subroutine shall be associated with exiting the run contents of the parent form field. If this element specifies a macro which cannot be located or is not supported by an application, then its value may be ignored, but shall not be lost upon resaving the file.

[*Example*: Consider the following WordprocessingML fragment for the properties of a form field:

```
<w:ffData>
  <w:exitMacro w:val="TestExitFunction" />
</w:ffData>
```

The exitMacro element specifies that any application which processes this file should attempt to locate and execute a scripting subroutine called `TestExitFunction` when the contents of the form field are exited. If this subroutine cannot be located or executed, then this setting is silently ignored. *end example*]

Parent Elements
ffData (§2.16.17)

Attributes	Description

val (Name of Script Function)	Specifies the name of a single scripting subroutine which shall be associated with the parent element. Its use is specifies based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment for the properties of a form field:
	
	`<w:ffData>`
	`  <w:exitMacro w:val="HelloWorld" />`
	`</w:ffData>`
	
	The val attribute specifies that a script function called `HelloWorld` shall be used in the context of the parent element; in this case, to execute when the field is exited. *end example*]
	
	The possible values for this attribute are defined by the ST_MacroName simple type (§2.18.58).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MacroName">
   <attribute name="val" use="required" type="ST_MacroName"/>
</complexType>
```

2.16.17 ffData (Form Field Properties)

This element specifies a set of properties which shall be associated with the parent form field within the document. This form field may be of any of the following types (with the associated field codes in parentheses):

- Checkbox (`FORMCHECKBOX`)
- Drop-down List (`FORMDROPDOWN`)
- Text box (`FORMTEXT`)

If this element is present and the field codes for the document do not specify a form field of one of these types, then the document shall be considered invalid.

If this element is omitted, then the properties associated with the parent form field shall be determined based on their default values.

[*Example*: Consider the following WordprocessingML fragment for a text box form field:

```
<w:fldSimple w:instr="FORMTEXT">
  <w:ffData>
    <w:name w:val="TextTextBox" />
    <w:enabled w:val="false"/>
    <w:textInput>
      <w:maxLength w:val="10" />
    </w:textInput>
  </w:ffData>
</w:fldSimple>
```

The ffData element specifies the set of properties for this text box form field; in this example, a form field name of TestTextBox via the name element (§2.16.28), a disabled state via the enabled element (§2.16.14), and a maximum character length of 10 characters via the maxLength element (§2.16.27). *end example*]

Parent Elements
fldChar (§2.16.18)

Child Elements	Subclause
calcOnExit (Recalculate Fields When Current Field Is Modified)	§2.16.6
checkBox (Checkbox Form Field Properties)	§2.16.7
ddList (Drop-Down List Form Field Properties)	§2.16.9
enabled (Form Field Enabled)	§2.16.14
entryMacro (Script Function to Execute on Form Field Entry)	§2.16.15
exitMacro (Script Function to Execute on Form Field Exit)	§2.16.16
helpText (Associated Help Text)	§2.16.23
name (Form Field Name)	§2.16.28
statusText (Associated Status Text)	§2.16.32
textInput (Text Box Form Field Properties)	§2.16.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FFData">
   <choice maxOccurs="unbounded">
      <element name="name" type="CT_FFName"/>
      <element name="enabled" type="CT_OnOff"/>
      <element name="calcOnExit" type="CT_OnOff"/>
      <element name="entryMacro" type="CT_MacroName" minOccurs="0" maxOccurs="1"/>
      <element name="exitMacro" type="CT_MacroName" minOccurs="0" maxOccurs="1"/>
      <element name="helpText" type="CT_FFHelpText" minOccurs="0" maxOccurs="1"/>
      <element name="statusText" type="CT_FFStatusText" minOccurs="0" maxOccurs="1"/>
      <choice>
         <element name="checkBox" type="CT_FFCheckBox"/>
         <element name="ddList" type="CT_FFDDList"/>
         <element name="textInput" type="CT_FFTextInput"/>
      </choice>
   </choice>
</complexType>
```

2.16.18 fldChar (Complex Field Character)

This element specifies the presence of a complex field character at the current location in the parent run. A *complex field character* is a special character which delimits the start and end of a complex field or separates its field codes from its current field result.

A complex field is defined via the use of the two required complex field characters: a *start character*, which specifies the beginning of a complex field within the document content; and an *end character*, which specifies the end of a complex field. This syntax allows multiple fields to be embedded (or "nested") within each other in a document.

As well, because a complex field may specify both its field codes and its current result within the document, these two items are separated by the optional *separator character*, which defines the end of the field codes and the beginning of the field contents. The omission of this character shall be used to specify that the contents of the field are entirely field codes (i.e. the field has no result).

[*Example*: Consider the following complex field definition within a WordprocessingML document:

```
<w:fldChar w:type="start" />
<w:r>
  <w:instrText>AUTHOR</w:instrText>
</w:r>
<w:fldChar w:type="separate" />
<w:r>
  <w:t>Rex Jaeschke</w:t>
</w:r>
<w:fldChar w:type="end" />
```

The three fldChar elements specify:

- The beginning of the field, using the type attribute value of `start`
- The separator between the field codes and the current field results, using the type attribute value of `separate`
- The end of the field, using the type attribute value of `end`

end example]

If a complex field character is located in an inappropriate location in a WordprocessingML document, then its presence shall be ignored and no field shall be present in the resulting document when displayed. Also, if a complex field is not closed before the end of a document story, then no field shall be generated and each individual run shall be processed as if the field characters did not exist (i.e. the contents of all field code run content shall not be displayed, and the field results shall be displayed as literal text).

[*Example*: Consider the following WordprocessingML document:

```
<w:body>
  <w:p>
    <w:fldChar w:type="start" />
    <w:r>
      <w:instrText>AUTHOR</w:instrText>
    </w:r>
    <w:fldChar w:type="separate" />
    <w:r>
      <w:t>Rex Jaeschke</w:t>
    </w:r>
  </w:p>
</w:body>
```

The complex field is technically incorrect since no end character exists in the main document story. The resulting content shall be interpreted as though no field characters exist, resulting in only the literal text `Rex Jaeschke` being displayed in the document. *end example*]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

Child Elements	Subclause
ffData (Form Field Properties)	§2.16.17
fldData (Custom Field Data)	§2.16.19
numberingChange (Previous Numbering Field Properties)	§2.13.5.29

Attributes	Description

Attributes	Description

dirty (Field Result Invalidated)	Specifies that this field has been flagged by an application to indicate that its current results are invalid (stale) due to other modifications made to the document, and these contents should be updated before they are displayed if this functionality is supported by the next processing application.
	
	[*Rationale*: This functionality allows applications with limited subsets of the full functionality of this Office Open XML Standard to process Word Open XML documents without needing to understand and update all fields based on the semantics for their field codes.
	
	For example, an application can add a new paragraph and flag the table of contents as dirty, without needing to understand anything about how to recalculate that field's content. *end rationale*]
	
	If this attribute is omitted, then its value shall be assumed to be `false`. If the type of the current field character is not `start`, then his setting may be ignored.
	
	[*Example*: Consider the following WordprocessingML for a complex field:
	
	`<w:fldChar w:type="start" w:dirty="true"/>`
	`<w:r>`
	`<w:instrText>TOC /l 1-3</w:instrText>`
	`</w:r>`
	`<w:fldChar w:type="separate"/>`
	…
	
	The dirty attribute value of `true` specifies that the contents of this field are no longer current based on the contents of the document, and should be recalculated whenever an application with this functionality reads the document. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

fldCharType (Field Character Type)	Specifies the type of the current complex field character in the document. [*Example*: Consider the following WordprocessingML for a complex field character: … `<w:fldChar w:type="separate" />` … The type attribute value of `separate` specifies that this is a complex field separator character; therefore it is being used to separate the field codes from the field contents in a complex field. *end example*] The possible values for this attribute are defined by the ST_FldCharType simple type (§2.18.33).
fldLock (Field	Specifies that the parent complex field shall not have its field result recalculated, even if

Attributes	Description

Should Not Be Recalculated)	an application attempts to recalculate the results of all fields in the document or a recalculation is explicitly requested.
	
	If this attribute is omitted, then its value shall be assumed to be `false`. If the type of the current field character is not `start`, then his setting may be ignored.
	
	[*Example*: Consider the following WordprocessingML for a complex field:
	
	`<w:fldChar w:type="start" w:fldLock="true"/>`
	`…`
	`<w:fldChar w:type="separate"/>`
	`<w:r>`
	`<w:t>field result</w:t>`
	`</w:r>`
	`<w:fldChar w:type="end" />`
	
	The fldLock attribute value of `true` specifies that the contents of this field shall remain `field result` regardless of the actual result of the current field codes. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FldChar">
   <choice>
      <element name="fldData" type="CT_Text" minOccurs="0" maxOccurs="1"/>
      <element name="ffData" type="CT_FFData" minOccurs="0" maxOccurs="1"/>
      <element name="numberingChange" type="CT_TrackChangeNumbering" minOccurs="0"/>
   </choice>
   <attribute name="fldCharType" type="ST_FldCharType" use="required"/>
   <attribute name="fldLock" type="ST_OnOff"/>
   <attribute name="dirty" type="ST_OnOff"/>
</complexType>
```

2.16.19 fldData (Custom Field Data)

This element specifies custom field data which shall be associated with the parent field. No information or semantics are applied to the contents of this data by this Office Open XML Standard, and therefore this field may be used as desired to store additional application-specific data with the field. However, applications should not lose the contents of this custom data if they do not understand or utilize it (i.e. the information should continue to be saved with the file).

If this element is omitted, then no custom field data is stored with the parent field. If the type attribute of the current field character is not `start`, then his setting may be ignored.

[*Example*: Consider the following WordprocessingML fragment for a complex field:

```
<w:fldChar w:type="start">
  <w:fldData xml:space="preserve">///3645ERKJHE</w:fldData>
</w:fldChar>
<w:r>
  <w:instrText>PRIVATE</w:instrText>
</w:r>
<w:fldChar w:type="separate" />
…
```

The fldData element contains custom data stored with this PRIVATE field (§2.16.5.55), the contents of which are determined by a hosting application. *end example*]

Parent Elements
fldChar (§2.16.18)

Attributes	Description

space (Content Contains Significant Whitespace)
 	
Namespace: http://www.w3.org/XML/1998/namespace
	Specifies how white space should be handled for the contents of this element using the W3C space preservation rules.
	
	[*Example*: Consider the following run contained within a WordprocessingML document:
	
	`<w:r>`
	`  <w:t>significant whitespace  </w:t>`
	`</w:r>`
	
	Although there are three spaces on each side of the text content in the run, that whitespace has not been specifically marked as significant, therefore it is removed when this run is added to the document. *end example*]
	
	The possible values for this attribute are defined by the type in the namespace.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Text">
   <simpleContent>
      <extension base="ST_String">
         <attribute ref="xml:space" use="optional"/>
      </extension>
   </simpleContent>
</complexType>
```

2.16.20 fldData (Custom Field Data)

This element specifies custom field data which shall be associated with the parent field. No information or semantics are applied to the contents of this data by this Office Open XML Standard, and therefore this field may be used as desired to store additional application-specific data with the field. However, applications should

not lose the contents of this custom data if they do not understand or utilize it (i.e. the information should continue to be saved with the file).

If this element is omitted, then no custom field data is stored with the parent field.

[*Example*: Consider the following WordprocessingML fragment for a simple field:

```
<w:fldSimple w:instr="PRIVATE">
  <w:fldData xml:space="preserve">///3645ERKJHE</w:fldData>
</w:fldSimple>
```

The fldData element contains custom data stored with this PRIVATE field (§2.16.5.55), the contents of which are determined by a hosting application. *end example*]

Parent Elements
fldSimple (§2.16.21)

Attributes	Description

space (Content Contains Significant Whitespace)
 	
Namespace: http://www.w3.org/XML/1998/namespace
	Specifies how white space should be handled for the contents of this element using the W3C space preservation rules.
	
	[*Example*: Consider the following run contained within a WordprocessingML document:
	
	`<w:r>`
	`  <w:t>significant whitespace  </w:t>`
	`</w:r>`
	
	Although there are three spaces on each side of the text content in the run, that whitespace has not been specifically marked as significant, therefore it is removed when this run is added to the document. *end example*]
	
	The possible values for this attribute are defined by the type in the namespace.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Text">
   <simpleContent>
      <extension base="ST_String">
         <attribute ref="xml:space" use="optional"/>
      </extension>
   </simpleContent>
</complexType>
```

2.16.21 fldSimple (Simple Field)

This element specifies the presence of a simple field at the current location in the document. The semantics of this field are defined via its field codes (§2.16.5).

[*Example*: Consider the following WordprocessingML fragment for a simple field:

```
<w:fldSimple w:instr="FILENAME">
  <w:r>
    <w:t>Example Document.docx</w:t>
  </w:r>
</w:fldSimple>
```

The fldSimple element defines a `FILENAME` field (§2.16.5.23) using the simple field syntax. The current field result for the field is `Example Document.docx`. *end example*]

Parent Elements
customXml (§2.5.1.5); fldSimple (§2.16.21); hyperlink (§2.16.24); p (§2.3.1.22); sdtContent (§2.5.2.35); smartTag (§2.5.1.9)

Child Elements	Subclause
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Inline-Level Custom XML Element)	§2.5.1.5
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
del (Deleted Run Content)	§2.13.5.12
fldData (Custom Field Data)	§2.16.20
fldSimple (Simple Field)	§2.16.21
hyperlink (Hyperlink)	§2.16.24
ins (Inserted Run Content)	§2.13.5.20
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26

Child Elements	Subclause
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
r (Text Run)	§2.3.2.23
sdt (Inline-Level Structured Document Tag)	§2.5.2.29
smartTag (Inline-Level Smart Tag)	§2.5.1.9
subDoc (Anchor for Subdocument Location)	§2.17.2.1

Attributes	Description
dirty (Field Result Invalidated)	Specifies that this field has been flagged by an application to indicate that its current results are invalid (stale) due to other modifications made to the document, and these contents should be updated before they are displayed if this functionality is supported by the next processing application. [*Rationale*: This functionality allows applications with limited subsets of the full functionality of this Office Open XML Standard to process Word Open XML documents without needing to understand and update all fields based on the semantics for their field codes. For example, an application can add a new paragraph and flag the table of contents as dirty, without needing to understand anything about how to recalculate that field's content. *end rationale*] If this attribute is omitted, then its value shall be assumed to be `false`. [*Example*: Consider the following WordprocessingML for a simple field: `<w:fldSimple w:instr="AUTHOR" w:dirty="true"/>` The dirty attribute value of `true` specifies that the contents of this field are no longer current based on the contents of the document, and should be recalculated whenever an application with this functionality reads the document. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).
fldLock (Field Should Not Be Recalculated)	Specifies that the parent field shall not have its field result recalculated, even if an application attempts to recalculate the results of all fields in the document or a recalculation is explicitly requested.

Attributes	Description

	If this attribute is omitted, then its value shall be assumed to be `false`.
	
	[*Example*: Consider the following WordprocessingML for a simple field:
	
	`<w:fldSimple w:instr="AUTHOR" w:fldLock="true">`
	`  <w:r>`
	`  <w:t>Rex Jaeschke</w:t>`
	`  </w:r>`
	`</w:fldSimple>`
	
	The fldLock attribute value of `true` specifies that the contents of this field shall remain `Rex Jaeschke` regardless of the actual result of the current field codes. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

instr (Field Codes)	Specifies the field codes for the simple field. The possible field codes are defined in §2.16.5.
	
	[*Example*: Consider the following WordprocessingML for a simple field:
	
	`<w:fldSimple w:instr="AUTHOR" w:fldLock="true">`
	`  <w:r>`
	`  <w:t>Rex Jaeschke</w:t>`
	`  </w:r>`
	`</w:fldSimple>`
	
	The instr attribute specifies the field codes for this simple field to be `AUTHOR`. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SimpleField">
   <sequence>
      <element name="fldData" type="CT_Text" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_PContent" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="instr" type="ST_String" use="required"/>
   <attribute name="fldLock" type="ST_OnOff"/>
   <attribute name="dirty" type="ST_OnOff"/>
</complexType>
```

2.16.22 format (Text Box Form Field Formatting)

This element specifies the field formatting which shall be applied to the contents of the parent form field whenever those contents are modified. The type of formatting which is applied to the field depends on the value of its type element (§2.16.34), as follows:

- When the type is equal to `currentDate`, `currentTime`, or `date`, a date formatting string using the syntax defined in §2.16.4.1
- When the type is equal to `calculated` or `number`, a number formatting string using the syntax defined in §2.16.4.2
- When the type is equal to regular, a text formatting string defined as follows:

Argument	Description
`Uppercase`	All letters are uppercase. [*Example*: `Mary Smith` results in `MARY SMITH`. *end example*]
`Lowercase`	All letters are lowercase. [*Example*: `Mary Smith` results in `mary smith`. *end example*]
`First capital`	Capitalizes the first letter of the first word. [*Example*: `Mary Smith` results in `Mary smith`. *end example*]
`Title case`	Capitalizes the first letter of each word. [*Example*: `Mary Smith` results in `Mary Smith`. *end example*]

[*Example*: Consider the following WordprocessingML fragment for the properties of a text box form field:

```
<w:ffData>
  <w:textInput>
    <w:type w:val="number" />
    <w:maxLength w:val="4" />
    <w:format w:val="0.00" />
  </w:textInput>
</w:ffData>
```

The format element specifies the field formatting which is applied to the input to the field (in this case, a grouping of number formatting picture items as the type element specifies a value of `number`). If a value of 8 was entered into this field, the formatted result would be `8.00`. *end example*]

Parent Elements
textInput (§2.16.33)

Attributes	Description
val (String Value)	Specifies that its contents will contain a string. The contents of this string are interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML fragment:

Attributes	Description
	<w:pPr> <w:pStyle w:val="heading1" /> </w:pPr> The value of the val attribute is the ID of the associated paragraph style's styleId. However, consider the following fragment: <w:sdtPr> <w:alias w:val="SDT Title Example" /> ... </w:sdtPr> In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.16.23 helpText (Associated Help Text)

This element specifies optional help text which shall be associated with the parent form field. The method or user interface by which this help text may be surfaced is not defined by this Office Open XML Standard.

If this element is omitted, then no help text shall be associated with the current form field.

[*Example*: Consider the following WordprocessingML fragment for a drop-down list form field:

```
<w:fldSimple w:instr="FORMDROPDOWN">
  <w:ffData>
    <w:helpText w:type="text" w:val="Example help text." />
    <w:ddList>
      …
    </w:ddList>
  </w:ffData>
</w:fldSimple>
```

The helpText element specifies the help text for the parent form field - in this case, literal help text consisting of the string `Example help text.` *end example*]

Parent Elements

Parent Elements
ffData (§2.16.17)

Attributes	Description

type (Help Text Type)	Specifies the type of help text which is specified by this element, defined by the simple type below.
	
	If this attribute is omitted, then its value shall be assumed to be `text`.
	
	[*Example*: Consider the following WordprocessingML fragment for a form field:
	
	`<w:ffData>`
	`  <w:helpText w:type="text" w:val="Example help text." />`
	`</w:ffData>`
	
	The type attribute has a value of `text`, which specifies that the text in the val attribute is the literal help text for this form field. *end example*]
	
	The possible values for this attribute are defined by the ST_InfoTextType simple type (§2.18.49).

val (Help Text Value)	Specifies the help text for the current form field. Based on the value of the type attribute, the contents of this field shall be interpreted as follows:
	• When the type attribute value is `text`, contains the literal help text for the form field.
	• When the type attribute value is `autoText`, contains the name of a glossary document entry which contains the help text for the form field.
	
	[*Example*: Consider the following WordprocessingML fragment for a form field:
	
	`<w:ffData>`
	`  <w:helpText w:type="autoText" w:val="HelpText" />`
	`</w:ffData>`
	
	The text in the val attribute is the name of a glossary document entry containing the help text for this form field, since the type attribute has a value of `autoText`. *end example*]
	
	The possible values for this attribute are defined by the ST_FFHelpTextVal simple type (§2.18.29).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FFHelpText">
   <attribute name="type" type="ST_InfoTextType"/>
   <attribute name="val" type="ST_FFHelpTextVal"/>
</complexType>
```

2.16.24 hyperlink (Hyperlink)

This element specifies the presence of a hyperlink at the current location in the document.

[*Example*: Consider the following WordprocessingML fragment for a hyperlink:

```
<w:hyperlink r:id="rId10">
  <w:r>
    <w:t>Click here</w:t>
  </w:r>
</w:hyperlink>
```

The hyperlink element defines a hyperlink whose display text is `Click here`, and whose target is specified by the relationship with an Id attribute value of `rId10`. *end example*]

Parent Elements
customXml (§2.5.1.5); fldSimple (§2.16.21); hyperlink (§2.16.24); p (§2.3.1.22); sdtContent (§2.5.2.35); smartTag (§2.5.1.9)

Child Elements	Subclause
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Inline-Level Custom XML Element)	§2.5.1.5
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
del (Deleted Run Content)	§2.13.5.12
fldSimple (Simple Field)	§2.16.21
hyperlink (Hyperlink)	§2.16.24
ins (Inserted Run Content)	§2.13.5.20
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23

Child Elements	Subclause
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
r (Text Run)	§2.3.2.23
sdt (Inline-Level Structured Document Tag)	§2.5.2.29
smartTag (Inline-Level Smart Tag)	§2.5.1.9
subDoc (Anchor for Subdocument Location)	§2.17.2.1

Attributes	Description

anchor (Hyperlink Anchor)	Specifies the name of a bookmark in the current document which shall be the target of this hyperlink.
	
	If this attribute is omitted, then the default behavior shall be to navigate to the start of the document. If a hyperlink target is also specified using the r:id attribute, then this attribute shall be ignored. If no bookmark exists in the current document with the given bookmark name, then the default behavior shall be to navigate to the start of the document.
	
	[*Example*: Consider the following WordprocessingML fragment for a hyperlink:
	
	`<w:hyperlink w:anchor="chapter3">`
	`  <w:r>`
	`  <w:t>Go to Chapter Three</w:t>`
	`  </w:r>`
	`</w:hyperlink>`
	
	The anchor attribute specifies that the target of the current hyperlink shall be the text contained within the bookmark `chapter3` within the document. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

docLocation (Location in Target Document)	Specifies a location in the target of the hyperlink that has no bookmarks. The method by which the contents of this attribute are linked to document text is outside the scope of this Office Open XML Standard.

Attributes	Description

	If this attribute is omitted, then no location shall be associated with the parent hyperlink. If the anchor attribute is also specified, then this attribute may be ignored when the hyperlink is invoked.
	
	[*Example*: Consider the following WordprocessingML fragment for a hyperlink:
	
	`<w:hyperlink r:id="rId9" w:docLocation="table">`
	`  <w:r>`
	`  <w:t>Click Here</w:t>`
	`  </w:r>`
	`</w:hyperlink>`
	
	The docLocation attribute specifies that the target of the current hyperlink shall be a region targeted by the string `table` within the target document. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

history (Add To Viewed Hyperlinks)	Specifies whether the target of the parent hyperlink (as specified via the r:id attribute) shall be added to a list of viewed hyperlinks when it is invoked.
	
	If this attribute is omitted, then its value shall be assumed to be `false`.
	
	[*Example*: Consider the following WordprocessingML fragment for a hyperlink:
	
	`<w:hyperlink r:id="rId9" w:history="true">`
	`  <w:r>`
	`  <w:t>http://example.com</w:t>`
	`  </w:r>`
	`</w:hyperlink>`
	
	The history attribute value of `true` specifies that the target of the current hyperlink shall be added to a list of visited hyperlinks when invoked within the document. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

id (Hyperlink Target)
 	
Namespace: .../officeDocument/2006/relationships
	Specifies the ID of the relationship whose target shall be used as the target for this hyperlink.
	
	If this attribute is omitted, then there shall be no external hyperlink target for the current hyperlink - a location in the current document may still be target via the anchor attribute. If this attribute exists, it shall supersede the value in the anchor attribute.
	
	[*Example*: Consider the following WordprocessingML fragment for a hyperlink:
	
	`<w:hyperlink r:id="rId9">`
	`  <w:r>`
	`  <w:t>http://example.com</w:t>`
	`  </w:r>`

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<pre></w:hyperlink></pre>
The id attribute value of rId9 specifies that relationship in the associated relationship part item with a corresponding Id attribute value shall be navigated to when this hyperlink is invoked. For example, if the following XML is present in the associated relationship part item:
<pre><Relationships xmlns="…">
  <Relationship Id="rId9" Mode="External"
Target=http://www.contoso.com />
</Relationships></pre>
The target of this hyperlink would therefore be the target of relationship rId9 - in this case, http://www.contoso.com. end example]
The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
</td></tr>
<tr><td>tgtFrame
(Hyperlink Target Frame)</td><td>
Specifies a frame within the parent HTML frameset for the target of the parent hyperlink when one exists. All values specified by this element shall be handled as follows:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>_top</td><td>Open hyperlink target in the full region of the current window.</td></tr>
<tr><td>_self</td><td>Open hyperlink target in the same frame as the hyperlink appears.</td></tr>
<tr><td>_parent</td><td>Open hyperlink target in the parent of the current frame, or the current frame if this frame has no parent.</td></tr>
<tr><td>_blank</td><td>Open hyperlink target in a new window.</td></tr>
<tr><td>all other values</td><td>Open hyperlink target in the frame with the specified name. If no frame exists with this name, open in the current frame.

If this string does not begin with an alphabetic character, it shall be ignored.</td></tr>
</table>
If this attribute is omitted, then no target frame information shall be associated with the
</td></tr>
</table>

Attributes	Description

	parent hyperlink. If the current document is not part of a frameset, then this information may be ignored.
	
	[*Example*: Consider the following WordprocessingML fragment for a hyperlink:
	
	`<w:hyperlink r:id="rId9" w:tgtFrame="_top">`
	`  <w:r>`
	`  <w:t>http://example.com</w:t>`
	`  </w:r>`
	`</w:hyperlink>`
	
	The tgtFrame attribute value of `_top` specifies that the target of this hyperlink shall be displayed in the full extents of the current window. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

tooltip (Associated String)	Specifies a string which may be surfaced in a user interface as associated with the parent hyperlink. The method by which this string is surfaced by an application is outside the scope of this Office Open XML Standard.
	
	If this attribute is omitted, then no associated string shall be linked to the parent hyperlink in the document.
	
	[*Example*: Consider the following WordprocessingML fragment for a hyperlink:
	
	`<w:hyperlink r:id="rId9" w:tooltip="Click here!">`
	`  <w:r>`
	`  <w:t>http://example.com</w:t>`
	`  </w:r>`
	`</w:hyperlink>`
	
	The tooltip attribute value specifies that the parent hyperlink has the associated string of `Click here!`, which may be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Hyperlink">
   <group ref="EG_PContent" minOccurs="0" maxOccurs="unbounded"/>
   <attribute name="tgtFrame" type="ST_String" use="optional"/>
   <attribute name="tooltip" type="ST_String" use="optional"/>
   <attribute name="docLocation" type="ST_String" use="optional"/>
   <attribute name="history" type="ST_OnOff" use="optional"/>
   <attribute name="anchor" type="ST_String" use="optional"/>
   <attribute ref="r:id"/>
</complexType>
```

2.16.25 instrText (Field Code)

This element specifies that this run contains field codes (§2.16.5) within a complex field in the document.

If this element is contained within a run which is not part of a complex field's field codes, then it and its contents should be treated as regular text. If this element is contained within a del element, then the document is invalid.

[*Example*: Consider a complex checkbox field within a WordprocessingML. This field would be represented as follows:

```
<w:fldchar w:type="begin" />
<w:r>
  <w:instrText>FORMCHECKBOX</w:instrText>
</w:r>
<w:fldChar w:type="seperate" />
…
<w:fldChar w:type="end" />
```

The field code is contained in a instrText node which occurs within the field codes portion of the complex field (i.e. before the separator character). *end example*]

Parent Elements
r (§7.1.2.87); r (§2.3.2.23)

Attributes	Description

space (Content Contains Significant Whitespace)
 	
Namespace: http://www.w3.org/XML/1998/namespace
	Specifies how white space should be handled for the contents of this element using the W3C space preservation rules.
	
	[*Example*: Consider the following run contained within a WordprocessingML document:
	
	`<w:r>`
	`  <w:t>significant whitespace  </w:t>`
	`</w:r>`
	
	Although there are three spaces on each side of the text content in the run, that whitespace has not been specifically marked as significant, therefore it is removed when this run is added to the document. *end example*]
	
	The possible values for this attribute are defined by the type in the namespace.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Text">
   <simpleContent>
      <extension base="ST_String">
         <attribute ref="xml:space" use="optional"/>
      </extension>
   </simpleContent>
</complexType>
```

2.16.26 listEntry (Drop-Down List Entry)

This element specifies the presence of a single drop-down list entry within the parent drop-down list form field in the document. The order of appearance of the series of listEntry elements in the WordprocessingML markup shall dictate the order of the entries in the drop-down list when it is displayed.

[*Example*: Consider the following WordprocessingML fragment for the properties of a drop-down list form field:

```
<w:ffData>
  <w:ddList>
    <w:listEntry w:val="One" />
    <w:listEntry w:val="Two" />
    <w:listEntry w:val="Three" />
  </w:ddList>
</w:ffData>
```

The three listEntry elements each specify one drop-down list entry for the parent drop-down list form field. In this case, these properties specify that the drop-down list shall contain three entries of `One`, `Two`, and `Three` in that order when displayed. *end example*]

Parent Elements
ddList (§2.16.9)

Attributes	Description

val (String Value)	Specifies that its contents will contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.

Attributes	Description

	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  ...`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§2.18.89).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String" use="required"/>
</complexType>
```

2.16.27 maxLength (Text Box Form Field Maximum Length)

This element specifies the maximum length of text which should be allowed within the parent text box form field before any formatting specified by the format element (§2.16.22). If the current contents of this field exceed the specified value when the document is loaded, that violation shall not result in an error, but the application shall prevent the addition of any additional characters until the contents are brought below that limit.

If this element is omitted, then there shall be no limit on the number of characters in the parent text box form field.

[*Example*: Consider the following WordprocessingML fragment for the properties of a text box form field:

```
<w:ffData>
  <w:textInput>
    <w:type w:val="number" />
    <w:maxLength w:val="4" />
    <w:format w:val="0.00" />
  </w:textInput>
</w:ffData>
```

The maxLength element specifies that the contents of this form field should not be allowed to exceed four characters when edited by an application. *end example*]

Parent Elements
textInput (§2.16.33)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
   <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.16.28 name (Form Field Name)

This element specifies the name of the current form field.

[*Example*: Consider the following WordprocessingML fragment for a text box form field:

```
<w:fldSimple w:instr="FORMTEXT">
  <w:ffData>
    <w:name w:val="FirstName"/>
    <w:textInput>
      …
    </w:textInput>
  </w:ffData>
</w:fldSimple>
```

The name element specifies that the name of the current form field is `FirstName`. *end example*]

Parent Elements
ffData (§2.16.17)

Attributes	Description
val (Form Field Name Value)	Specifies the name of the form field. If this attribute is omitted, then the parent form field shall have no name.

Attributes	Description

	[*Example*: Consider the following WordprocessingML fragment for a form field:
	
	`<w:ffData>`
	`  <w:name w:val="ExampleFieldName"/>`
	`</w:ffData>`
	
	The val attribute specifies that the name of the current form field is `ExampleFieldName`. *end example*]
	
	The possible values for this attribute are defined by the ST_FFName simple type (§2.18.30).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FFName">
   <attribute name="val" type="ST_FFName"/>
</complexType>
```

2.16.29 result (Drop-Down List Selection)

This element specifies the zero-based index of the currently selected entry for the parent drop-down list form field.

If this element is omitted, then the current drop-down list form field shall have the selection specified by the value of the default element (§2.16.11). If the attribute value references an index value which does not exist (i.e. a negative number or a number that exceeds the number of items in the drop-down list), then this value may be ignored and the current drop-down list form field shall have the selection specified by the value of the default element.

[*Example*: Consider the following WordprocessingML fragment for a drop-down list form field:

```
<w:fldSimple w:instr="FORMDROPDOWN">
  <w:ffData>
    <w:ddList>
      <w:default w:val="1" />
      <w:result w:val="2" />
      <w:listEntry w:val="One" />
      <w:listEntry w:val="Two" />
      <w:listEntry w:val="Three" />
    </w:ddList>
  </w:ffData>
</w:fldSimple>
```

The result element specifies the index of the currently selected value of the drop-down list form field to be 2. In this case, the resulting default value text is Three. *end example*]

Parent Elements
ddList (§2.16.9)

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute will contain a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_DecimalNumber: <w:… w:val="1512645511" /> The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§2.18.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DecimalNumber">
    <attribute name="val" type="ST_DecimalNumber" use="required"/>
</complexType>
```

2.16.30 size (Checkbox Form Field Size)

This element specifies the exact size for the parent checkbox form field. The resulting field shall be displayed in this point size regardless of the size specified by the formatting of its corresponding content in the document via the style hierarchy.

[*Example*: Consider the following WordprocessingML fragment for the properties of a checkbox form field:

```
<w:ffData>
  <w:checkBox>
    <w:size w:val="20" />
    <w:checked w:val="true" />
  </w:checkBox>
</w:ffData>
```

The size element specifies that the checkbox shall be displayed in a ten point font size, regardless of the formatting which would normally be applied to this text via the style hierarchy. *end example*]

Parent Elements
checkBox (§2.16.7)

Attributes	Description

val (Half Point Measurement)	Specifies a positive measurement specified in half-points (1/144 of an inch).
	
	The contents of this attribute value are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:rPr>`
	`  <w:sz w:val="28" />`
	`</w:rPr>`
	
	The value of the val attribute is the font size of the run's contents.
	
	However, consider the following fragment:
	
	`<w:rPr>`
	`  <w:kern w:val="30" />`
	`</w:rPr>`
	
	In this case, the value in the val attribute is the minimum size for which font characters shall be automatically kerned.
	
	In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_HpsMeasure simple type (§2.18.48).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HpsMeasure">
   <attribute name="val" type="ST_HpsMeasure" use="required"/>
</complexType>
```

2.16.31 sizeAuto (Automatically Size Form Field)

This element specifies that the parent checkbox form field shall be formatted using the point size which is applied to its field characters via the style hierarchy.

[*Example*: Consider the following WordprocessingML fragment for the properties of a checkbox form field:

```
<w:r>
  <w:rPr>
    <w:sz w:val="40"/>
  </w:rPr>
  <w:fldChar w:type="begin">
  <w:ffData>
    <w:checkBox>
      <w:sizeAuto />
      <w:checked w:val="true" />
    </w:checkBox>
  </w:ffData>
</w:r>
…
```

The sizeAuto element specifies that the checkbox shall be displayed in the point size of the formatting which would normally be applied to this text via the style hierarchy. In this case, this size is the twenty points specified via the direct formatting on the parent run. *end example*]

Parent Elements
checkBox (§2.16.7)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: <w:… w:val="off"/> The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.16.32 statusText (Associated Status Text)

This element specifies optional status text which shall be associated with the parent form field. The method or user interface by which this status text may be surfaced is not defined by this Office Open XML Standard.

If this element is omitted, then no status text shall be associated with the current form field.

[*Example*: Consider the following WordprocessingML fragment for a drop-down list form field:

```
<w:fldSimple w:instr="FORMDROPDOWN">
  <w:ffData>
    <w:statusText w:type="text" w:val="Example status text." />
    <w:ddList>
      …
    </w:ddList>
  </w:ffData>
</w:fldSimple>
```

The statusText element specifies the status text for the parent form field - in this case, literal text consisting of the string `Example status text.` *end example*]

Parent Elements
ffData (§2.16.17)

Attributes	Description

type (Status Text Type)	Specifies the type of status text which is specified by this element, defined by the simple type below.
	
	If this attribute is omitted, then its value shall be assumed to be `text`.
	
	[*Example*: Consider the following WordprocessingML fragment for a form field:
	
	`<w:ffData>`
	`  <w:statusText w:type="text" w:val="Example status text." />`
	`</w:ffData>`
	
	The type attribute has a value of `text`, which specifies that the text in the val attribute is the literal status text for this form field. *end example*]
	
	The possible values for this attribute are defined by the ST_InfoTextType simple type (§2.18.49).

val (Status Text Value)	Specifies the status text for the current form field. Based on the value of the type attribute, the contents of this field shall be interpreted as follows: • When the type attribute value is `text`, contains the literal status text for the form field.

Attributes	Description

	• When the type attribute value is autoText, contains the name of a glossary document entry which contains the status text for the form field.
	
	[*Example*: Consider the following WordprocessingML fragment for a form field:
	
	`<w:ffData>`
	`  <w:statusText w:type="autoText" w:val="MyStatusText" />`
	`</w:ffData>`
	
	The text in the val attribute is the name of a glossary document entry containing the status text for this form field, since the type attribute has a value of autoText. *end example*]
	
	The possible values for this attribute are defined by the ST_FFStatusTextVal simple type (§2.18.31).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FFStatusText">
   <attribute name="type" type="ST_InfoTextType"/>
   <attribute name="val" type="ST_FFStatusTextVal"/>
</complexType>
```

2.16.33 textInput (Text Box Form Field Properties)

This element specifies a set of properties which shall be associated with the parent FORMTEXT text box form field (§2.16.5.28) within the document.

If the parent form field is not a text box (i.e. its field code does not have a value of FORMTEXT), then these properties may be ignored.

[*Example*: Consider the following WordprocessingML fragment for the properties of a text box form field:

```
<w:ffData>
  <w:textInput>
    <w:maxLength w:val="4" />
    <w:type w:val="number" />
  </w:textInput>
</w:ffData>
```

The textInput element specifies that it contains a set of properties for the parent text box form field. In this case, these properties specify that the drop-down list shall contain no more than four characters via the maxLength element (§2.16.27), and that its contents shall contain a number via the type element (§2.16.34). *end example*]

Parent Elements

Parent Elements
ffData (§2.16.17)

Child Elements	Subclause
default (Default Text Box Form Field String)	§2.16.10
format (Text Box Form Field Formatting)	§2.16.22
maxLength (Text Box Form Field Maximum Length)	§2.16.27
type (Text Box Form Field Type)	§2.16.34

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FFTextInput">
   <sequence>
      <element name="type" type="CT_FFTextType" minOccurs="0"/>
      <element name="default" type="CT_String" minOccurs="0"/>
      <element name="maxLength" type="CT_DecimalNumber" minOccurs="0"/>
      <element name="format" type="CT_String" minOccurs="0"/>
   </sequence>
</complexType>
```

2.16.34 type (Text Box Form Field Type)

This element specifies the type of the contents of the current text box form field. This element shall not be used to prevent the successful loading of any contents in the field, but shall be used to parse the formatting specified in the format element (§2.16.22) and should be used to prevent the addition of illegal content when its contents are edited by an application.

If this element is omitted, then its default value shall be assumed to be `regular`.

[*Example*: Consider the following WordprocessingML fragment for the properties of a text box form field:

```
<w:ffData>
  <w:textInput>
    <w:type w:val="number" />
    <w:maxLength w:val="4" />
    <w:format w:val="0.00" />
  </w:textInput>
</w:ffData>
```

The type element specifies that the contents of this form field should be handled as a `number` by an application. *end example*]

Parent Elements
textInput (§2.16.33)

Attributes	Description

val (Text Box Form Field Type Values)	Specifies the type of the text box form field, as defined by the simple type referenced below.
	
	[*Example*: Consider the following WordprocessingML fragment for the properties of a text box form field:
	
	`<w:ffData>`
	`  <w:textInput>`
	`  <w:type w:val="currentDate" />`
	`  </w:textInput>`
	`</w:ffData>`
	
	The val attribute value of `currentDate` specifies that the contents of this form field should be the current date when the field is updated. *end example*]
	
	The possible values for this attribute are defined by the ST_FFTextType simple type (§2.18.32).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FFTextType">
   <attribute name="val" type="ST_FFTextType" use="required"/>
</complexType>
```

2.17 Miscellaneous Topics

This section covers topics not covered elsewhere within the WordprocessingML documentation.

2.17.1 Text Box Content

All VML-based drawing objects (except for connectors) support the addition of rich WordprocessingML content within their extents. When WordprocessingML contents have been added to a VML drawing object, the resulting text is contained within a *text box*.

When WordprocessingML content is contained within a text box, it is allowed within the object by specifying the VML textbox element (§6.1.2.22), which contains within it a single txbxContent element that contains all of the desired WordprocessingML content.

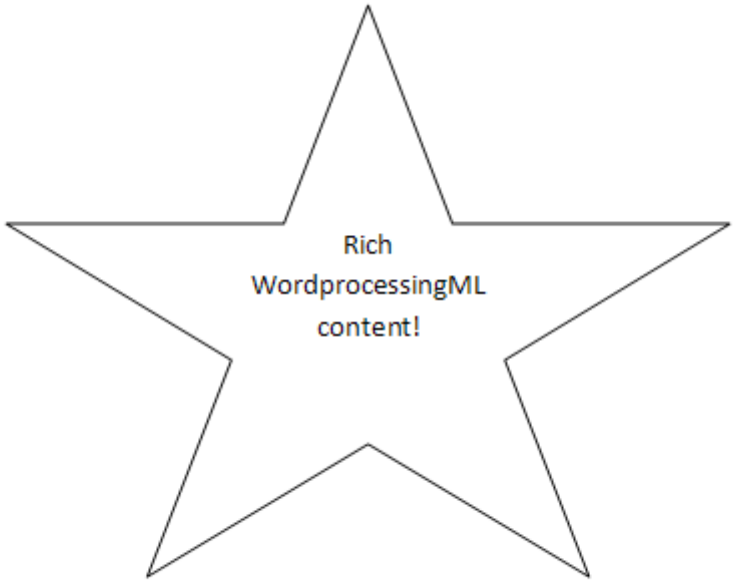
2.17.1.1 txbxContent (Rich Text Box Content Container)

This element specifies that its contents shall be any rich WordprocessingML content, and that this content is the rich contents of a drawing object defined using the Vector Markup Language (VML) syntax (§6.1).

If this element contains within any of its contents any of the following content, then the document shall be considered non-conformant:

- References to other WordprocessingML document stories (comments, footnotes, endnotes)
- Additional txbxContent elements (as part of nested VML objects)

[*Example*: Consider a WordprocessingML document consisting of a single VML shape element (§6.1.2.19) (in this case, a star) that contains within it some WordprocessingML content:



That drawing object now contains a text box, and so uses the syntax for that text box:

```
<v:shape id="_x0000_s1026" type="#_x0000_t12" style="…">
  <v:textbox>
    <w:txbxContent>
      <w:p>
        <w:pPr>
          <w:jc w:val="center"/>
        </w:pPr>
        <w:r>
          <w:t>Rich WordprocessingML content!</w:t>
        </w:r>
      </w:p>
    </w:txbxContent>
  </v:textbox>
</v:shape>
```

The txbxContent element is the container for the WordprocessingML contained within the text box inside that shape - once inside this element any content (subject to the restrictions defined above) may be used. *end example*]

Parent Elements

Parent Elements
textbox (§6.1.2.22)

Child Elements	Subclause
altChunk (Anchor for Imported External Content)	§2.17.3.1
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
customXml (Block-Level Custom XML Element)	§2.5.1.6
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
del (Deleted Run Content)	§2.13.5.12
ins (Inserted Run Content)	§2.13.5.20
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
p (Paragraph)	§2.3.1.22
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
proofErr (Proofing Error Anchor)	§2.13.8.1
sdt (Block-Level Structured Document Tag)	§2.5.2.30
tbl (Table)	§2.4.36

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TxbxContent">
   <group ref="EG_BlockLevelElts" minOccurs="1" maxOccurs="unbounded"/>
</complexType>
```

2.17.2 Subdocuments

Within a WordprocessingML document, it is sometimes necessary to break a large document into two or more separate WordprocessingML document files, allowing each of these files to be distributed, edited, and handled independently.

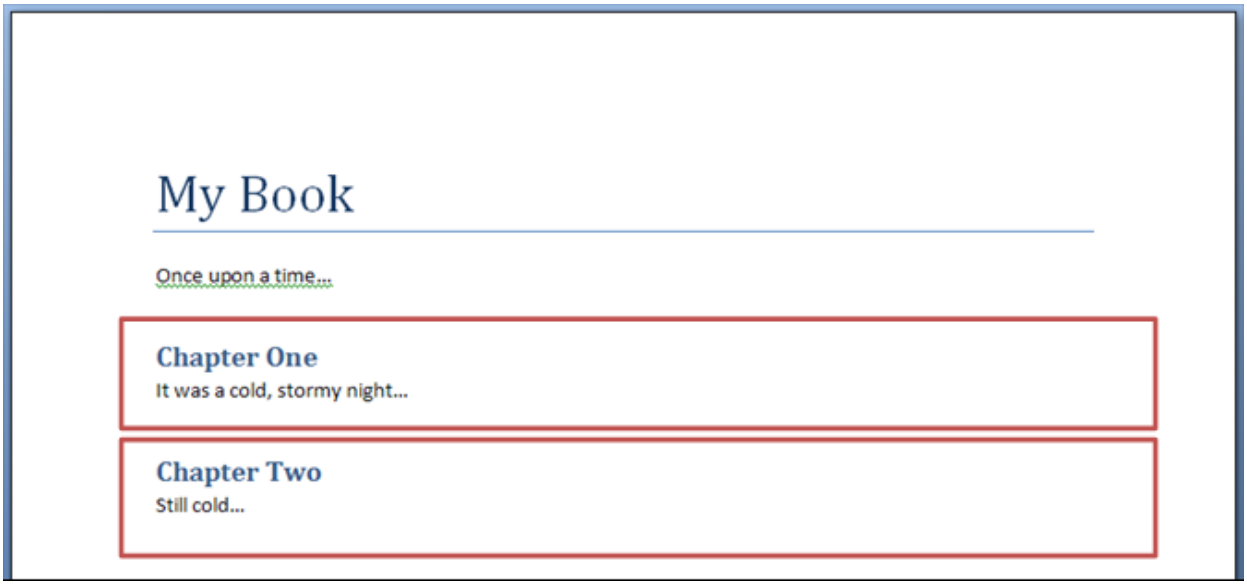
[*Example*: A book might consist of five chapters, each edited by a separate author. The editor for the book would therefore desire to create six WordprocessingML documents - one for each author to work on their chapter, and a main document which collates the content of the five chapters appropriately. *end example*]

When a WordprocessingML document is comprised of other WordprocessingML documents in this way, the resulting documents are called a master document and its subdocuments.

- A *subdocument* is a WordprocessingML document - there is no specific information in a document which classifies it as such, other than that it is incorporated into another document.
- A *master document* is a document which incorporates one or more subdocuments (as well as optional WordprocessingML content) to create a larger document

[*Example*: Consider a WordprocessingML document which is being used to write a book:

To allow this document to be written by multiple authors, each chapter in the book is placed in a separate file (the sections highlighted in red below):



The result is three WordprocessingML documents:

- A master document (containing the title of the book, the first paragraph, and references to the subdocuments for each chapter)
- Two subdocuments (one for each chapter)

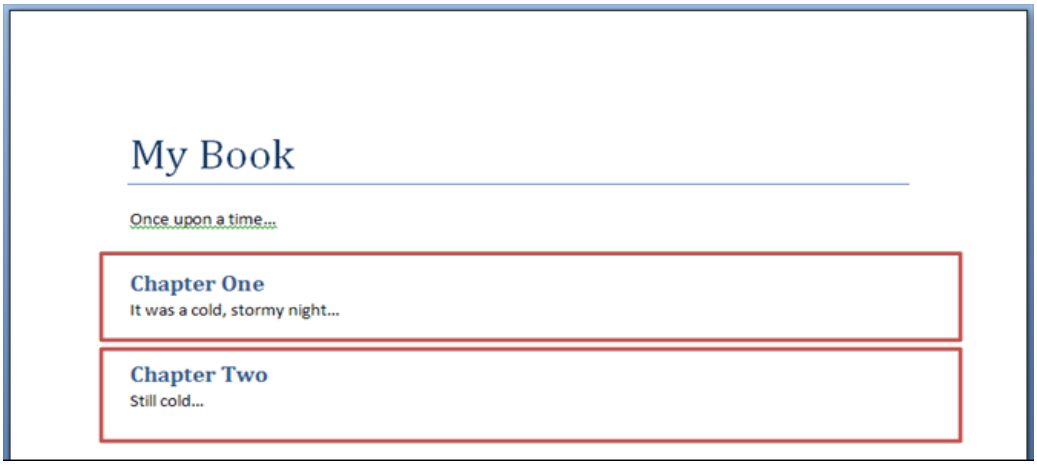
end example]

2.17.2.1 subDoc (Anchor for Subdocument Location)

This element specifies a location within a master document for the insertion of the contents of a specified subdocument. The specified subdocument's contents should appear at the specified location within the master document as needed, but shall remain part of the separate file specified by the subdocument location. The location of the subdocument shall be specified by the relationship whose Id attribute matches the id attribute on this element.

If the relationship type of the relationship specified by this element is not http://schemas.openxmlformats.org/officeDocument/2006/subDocument, is not present, or does not have a TargetMode attribute value of `External`, then the document shall be considered non-conformant.

[*Example*: Consider a book consisting of three chapters, two of which have been divided into subdocuments as follows (the red rectangle indicates the bounds of each subdocument's contents):



The resulting master document would consist of its own WordprocessingML content as well as subdocument anchors in the appropriate locations:

```
<w:body>
  <w:p>
    …
    <w:r>
      <w:t>My Book</w:t>
    </w:r>
  </w:p>
  <w:p>
    <w:r>
      <w:t>Once upon a time…</w:t>
    </w:r>
  </w:p>
  <w:subDoc r:id="subDocRel1" />
  <w:subDoc r:id="subDocRel2" />
  <w:sectPr>
    …
  </w:sectPr>
</w:body>
```

The two subDoc elements specify that the subdocuments targeted by the relationships with an ID of subDocRel1 and subDocRel2 shall be imported in that order after the content of the first two paragraphs of content. Examining the contents of the corresponding relationship part item, we can see the targets for those relationships:

```
<Relationships … >
  …
  <Relationship Id="subDocRel1" TargetMode="External"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/subDoc
ument" Target="Chapter1.docx" />
  <Relationship Id="subDocRel2" TargetMode="External"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/subDoc
ument" Target="Chapter2.docx" />
  …
</Relationships>
```

The corresponding relationship part item shows that the two files to be imported are located in the same location as the current file and name `Chapter1.docx` and `Chapter2.docx` respectively. *end example*]

Parent Elements
customXml (§2.5.1.5); fldSimple (§2.16.21); hyperlink (§2.16.24); p (§2.3.1.22); sdtContent (§2.5.2.35); smartTag (§2.5.1.9)

Attributes	Description
id (Relationship to Part) Namespace: …/officeDocument/2006/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the type required by the parent element: • http://schemas.openxmlformats.org/officeDocument/2006/relationships/footer for the footerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/header for the headerReference element • http://schemas.openxmlformats.org/officeDocument/2006/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://schemas.openxmlformats.org/officeDocument/2006/relationships/printerSettings for the printerSettings element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Rel">
   <attribute ref="r:id" use="required"/>
</complexType>
```

2.17.3 External Content Import

When generating WordprocessingML documents, it is sometimes necessary to include existing document content (henceforth called *external content*) within the document. External content in a document is typically included because it was stored in a format other than the WordprocessingML format defined by this Office Open XML Standard.

In order to facilitate the inclusion of such content without requiring its conversion as a prerequisite to its inclusion in a document, WordprocessingML includes the facility for applications to implement the import of external content in any format as part of a WordprocessingML document. This functionality, called external content import, allows the inclusion of content of an arbitrary content type within the WordprocessingML package, which shall then be opened and merged into the main document when the package is consumed by applications which understand that content type.

[*Example*: Consider a WordprocessingML document which is being created based on the following existing HTML content:

```
<html … >
  <body style="margin-left:200px;margin-top:50px">
    <p>Paragraph one.</p>
    <blockquote style="border:5px solid #00FFFF">Paragraph in a
blockquote.</blockquote>
    <p>Paragraph two.</p>
  </body>
</html>
```

This content could be converted to its WordprocessingML equivalents using the XML syntax defined by this Office Open XML Standard, or a more basic tool can use the external content import to include the HTML document within a WordprocessingML package, allowing a subsequent consumer of that content to import the resulting HTML. When the resulting WordprocessingML package is opened, the HTML document shall be read (if it is an alternate format understood by the consuming application) and migrated into the appropriate location in the main WordprocessingML document. *end example*]

2.17.3.1 altChunk (Anchor for Imported External Content)

This element specifies a location within a document for the insertion of the contents of a specified file containing external content to be imported into the main WordprocessingML document. The specified file's contents should appear at the specified location within the document, and may henceforth be emitted as regular WordprocessingML without distinction to its origin. The location of the external content to be imported shall be specified by the relationship whose Id attribute matches the id attribute on this element.

If the relationship type of the relationship specified by this element is not http://schemas.openxmlformats.org/officeDocument/2006/afChunk, is not present, or does not have a TargetMode attribute value of `Internal`, then the document shall be considered non-conformant. If an application cannot process external content of the content type specified by the targeted part, then it should ignore the specified alternate content but continue to process the file. If possible, it should also provide some indication that unknown content was not imported.

[*Example*: Consider a WordprocessingML document consisting of contents which shall be imported from the following HTML document:

```
<html … >
  <body style="margin-left:200px;margin-top:50px">
    <p>Paragraph one.</p>
    <blockquote style="border:5px solid #00FFFF">Paragraph in a
blockquote.</blockquote>
    <p>Paragraph two.</p>
  </body>
</html>
```

The resulting WordprocessingML host document would consist of its own WordprocessingML content as well as an external content import anchor in the appropriate location:

```
<w:body>
  <w:altChunk r:id="altChunk1" />
  <w:p/>
  <w:sectPr>
    …
  </w:sectPr>
</w:body>
```

The altChunk element specifies that the external content targeted by the relationship with an ID of `altChunk1` shall be imported at the beginning of the document. Examining the contents of the corresponding relationship part item, we can see the targets for that relationship:

```
<Relationships … >
  …
  <Relationship Id="altChunk1" TargetMode="Internal"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/afChun
k" Target="import.htm" />
  …
</Relationships>
```

The corresponding relationship part item shows that the file to be imported is located next to the main document and is named `import.htm`. *end example*]

Parent Elements
body (§2.2.2); comment (§2.13.4.2); docPartBody (§2.12.6); endnote (§2.11.2); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); tc (§2.4.62); txbxContent (§2.17.1.1)

Child Elements	Subclause
altChunkPr (External Content Import Properties)	§2.17.3.2

Attributes	Description
id (Relationship to Part) Namespace: .../officeDocument/2006/relationships	Specifies the relationship ID to a specified part containing alternate content for import. If the specified relationship does not match the type required by the parent element, then this document shall be considered to be invalid. [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AltChunk">
   <sequence>
      <element name="altChunkPr" type="CT_AltChunkPr" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute ref="r:id" use="optional"/>
</complexType>
```

2.17.3.2 altChunkPr (External Content Import Properties)

This element specifies the set of properties which shall be applied to the import of the external content specified by the parent altChunk element. Within this Office Open XML Standard, only one property is specified.

[*Example*: Consider a WordprocessingML document consisting of contents which contains an external content import anchor in the appropriate location:

```
<w:body>
  <w:altChunk r:id="altChunk1">
    <w:altChunkPr>
      <w:matchSrc w:val="false" />
    </w:altChunkPr>
  </w:altChunk>
  <w:p/>
  <w:sectPr>
    …
  </w:sectPr>
</w:body>
```

The altChunkPr element specifies the set of properties applied to the external content import when importing the specified content. *end example*]

Parent Elements
altChunk (§2.17.3.1)

Child Elements	Subclause
matchSrc (Keep Source Formatting on Import)	§2.17.3.3

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AltChunkPr">
   <sequence>
      <element name="matchSrc" type="CT_OnOff" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

2.17.3.3 matchSrc (Keep Source Formatting on Import)

This element specifies if any style definitions present in the imported content shall be overridden by identical styles present in the host WordprocessingML document. If this element's val attribute is `true`, then any style exists in both the imported content and main document shall be maintained on the imported content by redefining the style name and/or ID as needed. Conversely, if this element's val attribute is false, any style which exists in both the imported content and main document shall apply the style form the main document in place of the style in the imported content.

If this element is omitted, then styles from the main document shall override identical styles from the imported content.

[*Example*: Consider a WordprocessingML document consisting of contents which contains an external content import anchor in the appropriate location:

```
<w:body>
  <w:altChunk r:id="altChunk1">
    <w:altChunkPr>
      <w:matchSrc w:val="true" />
    </w:altChunkPr>
  </w:altChunk>
  <w:p/>
  <w:sectPr>
    …
  </w:sectPr>
</w:body>
```

The matchSrc element has a val attribute value of true, which specifies that conflicting styles shall be maintained when importing the specified content. For example, if the Heading 1 style was defined in both places, then applications shall ensure that the resulting document does not lose either instance of its formatting as appropriate. *end example*]

Parent Elements
altChunkPr (§2.17.3.2)

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<w:… w:val="off"/>` The val attribute explicitly declares that the property is turned off. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§2.18.67).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

2.17.4 Roundtripping Alternate Content

Office Open XML defines a mechanism for the storage of content which is not defined by this Office Open XML Standard, for example extensions developed by future software applications which leverage the Open XML formats. This mechanism allows for the storage of a series of alternative representations of content, of which the consuming application should use the first alternative whose requirements are met.

[*Example*: Consider an application which creates a new paragraph property intended to make the colors of its text change colors randomly when it is displayed. This functionality is not defined in this Office Open XML Standard, and so the application might choose to create an alternative representation setting a different manual color on each character for clients which do not understand this extension using an AlternateContent block as follows:

```
<ve:AlternateContent xmlns:ve="…">
  <ve:Choice Requires="colors" xmlns:colors="urn:randomTextColors">
    <w:p>
      <w:pPr>
        <colors:random colors:val="true" />
      </w:pPr>
      <w:r>
        <w:t>Random colors!</w:t>
      </w:r>
    </w:p>
  </ve:Choice>
  <ve:Fallback>
    <w:p>
      <w:r>
        <w:rPr>
          <w:color w:val="FF0000" />
        </w:rPr>
        <w:t>R</w:t>
      </w:r>
      <w:r>
        <w:rPr>
          <w:color w:val="00FF00" />
        </w:rPr>
        <w:t>a</w:t>
      </w:r>
      …
    </w:p>
  </ve:Fallback>
</ve:AlternateContent>
```

The Choice element that requires the new color extensions uses the random element in its namespace, and the Fallback element allows clients that do not support this namespace to see an appropriate alternative representation. *end example*]

These alternate content blocks may occur at any location within a WordprocessingML document, and applications shall handle and process them appropriately (taking the appropriate choice).

However, WordprocessingML does not explicitly define a set of locations where applications shall attempt to store and roundtrip all non-taken choices whenever possible. This behavior is therefore application-defined.

[*Example*: If an application does not understand the colors extension, the resulting file (if alternate choices are to be preserved would appear as follows:

```
<ve:AlternateContent xmlns:ve="…">
  <ve:Choice Requires="colors" xmlns:colors="urn:randomTextColors">
    …
  </ve:Choice>
  <ve:Fallback>
    …
  </ve:Fallback>
</ve:AlternateContent>
```

The file would then appear as follows after the choice is processed:

```
<w:p>
  <w:r>
    <w:rPr>
      <w:color w:val="FF0000" />
    </w:rPr>
    <w:t>R</w:t>
  </w:r>
  <w:r>
    <w:rPr>
      <w:color w:val="00FF00" />
    </w:rPr>
    <w:t>a</w:t>
  </w:r>
  …
</w:p>
```

The state of the alternate choices (preserved or not) is dependent on the application hosting the file. Preserving the content involves storing each non-taken choice while the file is being edited, and writing out the file with an AlternateContent block when it is resaved. *end example*]

2.18 Simple Types

This is the complete list of simple types in the http://schemas.openxmlformats.org/wordprocessingml/2006/main namespace.

2.18.1 ST_AlgClass (Cryptographic Algorithm Classes)

This simple type specifies the possible classes of cryptographic algorithm used by protection. [*Note*: The initial version of this Office Open XML Standard only supports a single version - hash - but future versions may expand this as necessary. *end note*]

[*Example*: Consider a WordprocessingML document with the following information stored in one of its protection elements:

```
<w:… w:cryptAlgorithmClass="hash"
  w:cryptAlgorithmType="typeAny"
  w:cryptAlgorithmSid="1"
  w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" />
```

The cryptAlgorithmClass attribute value of hash specifies that the algorithm used for the password is a hashing algorithm. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
hash (Hashing)	Specifies that the algorithm is a hashing function, which creates a hash value for user-supplied input that is very difficult to reverse-engineer.

Referenced By
documentProtection@cryptAlgorithmClass (§2.15.1.28); writeProtection@cryptAlgorithmClass (§2.15.1.94)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_AlgClass">
   <restriction base="xsd:string">
      <enumeration value="hash"/>
   </restriction>
</simpleType>
```

2.18.2 ST_AlgType (Cryptographic Algorithm Types)

This simple type specifies the possible values for the type of cryptographic algorithm used by protection. [*Note*: The initial version of this Office Open XML Standard only supports a single type - typeAny - but future versions may expand this as necessary. *end note*]

[*Example*: Consider a WordprocessingML document with the following information stored in one of its protection elements:

```
<w:… w:cryptAlgorithmClass="hash"
  w:cryptAlgorithmType="typeAny"
  w:cryptAlgorithmSid="1"
  w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" />
```

The cryptAlgorithmType attribute value of `typeAny` specifies that any type of algorithm may have been used for the password. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
typeAny (Any Type)	Specifies that any type of cryptographic algorithm type may be used.

Referenced By
documentProtection@cryptAlgorithmType (§2.15.1.28); writeProtection@cryptAlgorithmType (§2.15.1.94)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_AlgType">
   <restriction base="xsd:string">
      <enumeration value="typeAny"/>
   </restriction>
</simpleType>
```

2.18.3 ST_AnnotationVMerge (Table Cell Vertical Merge Revision Type)

This simple type specifies the possible values for the vertical merge setting which applied to a table cell by a cell merge (or split) revision.

[*Example*: Consider a two row by two column table in which the cells in the second column are merged, and this change is tracked as a revision. The annotation on the last cell in the table would appear as follows:

```
<w:tc>
  <w:tcPr>
    <w:cellMerge … w:vmerge="cont" />
  </w:tcPr>
  …
</w:tc>
```

The vmerge attribute value of `cont` specifies that the revision on the table cell resulted in it being merged with the previous set of vertically merged cells above it (whether that was one cell or many). *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
cont (Vertically Merged Cell)	Specifies that the revision resulted in this cell being vertically merged with the cell above it.
rest (Vertically Split Cell)	Specifies that the revision resulted in this cell being vertically split from the one above it.

Referenced By
cellMerge@vMerge (§2.13.5.3); cellMerge@vMergeOrig (§2.13.5.3)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_AnnotationVMerge">
   <restriction base="xsd:string">
      <enumeration value="cont"/>
      <enumeration value="rest"/>
   </restriction>
</simpleType>
```

2.18.4 ST_Border (Border Styles)

This simple type specifies the types of borders which can be specified for WordprocessingML objects which have a border.

Borders can be separated into two types:

- *Line borders*, which specify a pattern to be used when drawing a line around the specified object.
- *Art borders*, which specify a repeated image to be used when drawing a border around the specified object.

Line borders may be specified on any object which allows a border, however, art borders may only be used as a border at the page level - the borders under the pgBorders element (§2.6.10).

[*Example*: Consider a left border resulting in the following WordprocessingML:

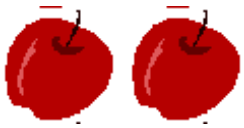
```
<w:left w:val="single" …/>
```

This border's val is `single`, indicating that the border style is a single line border. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description

Enumeration Value	Description
apples (Apples Art Border)	Specifies an art border consisting of a repeated image of an apple, as follows (showing two repetitions): 
archedScallops (Arched Scallops Art Border)	Specifies an art border consisting of a repeated image of a shell pattern, as follows (showing two repetitions): 
babyPacifier (Baby Pacifier Art Border)	Specifies an art border consisting of a repeated image of a baby pacifier, as follows (showing two repetitions): 
babyRattle (Baby Rattle Art Border)	Specifies an art border consisting of a repeated image of a baby rattle, as follows (showing two repetitions): 
balloons3Colors (Three Color Balloons Art Border)	Specifies an art border consisting of a repeated image of a set of balloons, as follows (showing two repetitions): 
balloonsHotAir (Hot Air Balloons Art Border)	Specifies an art border consisting of a repeated image of a hot air balloon, as follows (showing two repetitions): 
basicBlackDashes (Black Dash Art Border)	Specifies an art border consisting of a repeating image of a black and white background.

Enumeration Value	Description
	If the border is on the left or right, this image is as follows (repeated twice): If the border is on the top or bottom, this image is as follows (repeated twice): At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately:
basicBlackDots (Black Dot Art Border)	Specifies an art border consisting of a repeating image of a black dot on a white background. If the border is on the left or right, this image is as follows (showing two repetitions): If the border is on the top or bottom, this image is as follows (showing two repetitions): At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately:
basicBlackSquares (Black Square Art Border)	Specifies an art border consisting of a repeating image of a black and white background. If the border is on the left or right, this image is as follows (repeated twice):  If the border is on the top or bottom, this image is as follows (repeated twice):

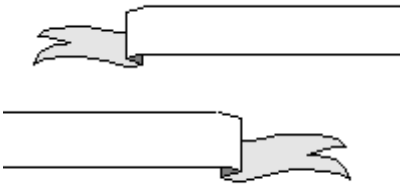
Enumeration Value	Description
	  At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately:
basicThinLines (Thin Line Art Border)	Specifies an art border consisting of a repeating image of a black and white background. If the border is on the left or right, this image is as follows:  If the border is on the top or bottom, this image is as follows:  At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
basicWhiteDashes (White Dash Art Border)	Specifies an art border consisting of a repeating image of a black and white background. If the border is on the left or right, this image is as follows (repeated twice):  If the border is on the top or bottom, this image is as follows (repeated twice): At any corner where two borders of this type intersect,

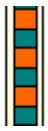
Enumeration Value	Description
	the intersection shall use the following image, rotated appropriately:
basicWhiteDots (White Dot Art Border)	Specifies an art border consisting of a repeating image of a white dot on a black background. If the border is on the left or right, this image is as follows (showing two repetitions):  If the border is on the top or bottom, this image is as follows (showing two repetitions):  At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
basicWhiteSquares (White Square Art Border)	Specifies an art border consisting of a repeating image of a black and white background. If the border is on the left or right, this image is as follows (repeated twice): If the border is on the top or bottom, this image is as follows (repeated twice): At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
basicWideInline (Wide Inline Art Border)	Specifies an art border consisting of a repeating image of a black and white background.

Enumeration Value	Description
	If the border is on the left or right, this image is as follows (showing for the left, flipped horizontally for the right border):  If the border is on the top or bottom, this image is as follows (showing for the top, flipped vertically for the bottom border):  At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
basicWideMidline (Wide Midline Art Border)	Specifies an art border consisting of a repeating image of a black and white background. If the border is on the left or right, this image is as follows (showing for the left, flipped horizontally for the right border):  If the border is on the top or bottom, this image is as follows (showing for the top, flipped vertically for the bottom border):  At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
basicWideOutline (Wide Outline Art Border)	Specifies an art border consisting of a repeating image

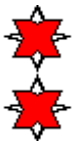
Enumeration Value	Description
	of a black and white background. If the border is on the left or right, this image is as follows (showing for the left, flipped horizontally for the right border):  If the border is on the top or bottom, this image is as follows (showing for the top, flipped vertically for the bottom border):  At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
bats (Bats Art Border)	Specifies an art border consisting of a repeated image of bats, as follows (showing two repetitions): 
birds (Birds Art Border)	Specifies an art border consisting of repeating images of birds. If the border is on the left or right, no border is displayed. If the border is on the top or bottom, this image is as follows (showing two repetitions): 
birdsFlight (Birds Flying Art Border)	Specifies an art border consisting of a repeated image of birds flying, as follows (showing two repetitions):

Enumeration Value	Description
	
cabins (Cabin Art Border)	Specifies an art border consisting of a repeated image of a cabin, as follows (showing two repetitions): 
cakeSlice (Cake Art Border)	Specifies an art border consisting of a repeated image of a piece of cake, as follows (showing two repetitions): 
candyCorn (Candy Corn Art Border)	Specifies an art border consisting of a repeated image of candy corn, as follows (showing two repetitions): 
celticKnotwork (Knot Work Art Border)	Specifies an art border consisting of a repeated image of a knot work pattern, as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
certificateBanner (Certificate Banner Art Border)	Specifies an art border consisting of a banner.

Enumeration Value	Description
	If the border is on the left or right, no border is displayed. If the border is on the top, this image is as follows (showing each end):  If this border is on the bottom, then the ends shall be flipped vertically.
chainLink (Chain Link Art Border)	Specifies an art border consisting of a repeating image of a chain link pattern. If the border is on the left or right, this image is as follows (showing two repetitions):  If the border is on the top or bottom, this image is as follows (showing two repetitions): 
champagneBottle (Champagne Bottle Art Border)	Specifies an art border consisting of a repeated image of a champagne bottle, as follows (showing two repetitions): 
checkedBarBlack (Black and White Bar Art Border)	Specifies an art border consisting of repeating images of a compass. If the border is on the left or right, this image is as follows (showing two repetitions on the left, the right would be flipped horizontally):

Enumeration Value	Description
	 If the border is on the top or bottom, this image is as follows (showing two repetitions on top, the bottom would be flipped vertically):  At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
checkedBarColor (Color Checked Bar Art Border)	Specifies an art border consisting of a repeating image of a colored pattern. If the border is on the left or right, this image is as follows (showing two repetitions):  If the border is on the top or bottom, this image is as follows (showing two repetitions):  At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
checkered (Checkerboard Art Border)	Specifies an art border consisting of a repeated image of a checkerboard, as follows (showing two repetitions): 
christmasTree (Christmas Tree Art Border)	Specifies an art border consisting of a repeated image of a Christmas tree, as follows (showing two

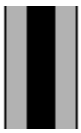
Enumeration Value	Description
	repetitions): 
circlesLines (Circles And Lines Art Border)	Specifies an art border consisting of repeating images of lines and circles. If the border is on the left or right, this image is as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image: 
circlesRectangles (Circles and Rectangles Art Border)	Specifies an art border consisting of a repeated image of a rectangular pattern, as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 

Enumeration Value	Description
classicalWave (Wave Art Border)	Specifies an art border consisting of a repeated image of a wave, as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
clocks (Clocks Art Border)	Specifies an art border consisting of a repeated image of a clock, as follows (showing two repetitions): 
compass (Compass Art Border)	Specifies an art border consisting of repeating images of a compass. If the border is on the left or right, this image is as follows (showing two repetitions):  If the border is on the top or bottom, this image is as follows (showing two repetitions):  At any corner where two borders of this type intersect, the intersection shall use the following image: 
confetti (Confetti Art Border)	Specifies an art border consisting of a repeated image of confetti, as follows (showing two repetitions):

Enumeration Value	Description
	
confettiGrays (Confetti Art Border)	Specifies an art border consisting of a repeated image of confetti, as follows (showing two repetitions): 
confettiOutline (Confetti Art Border)	Specifies an art border consisting of a repeated image of confetti, as follows (showing two repetitions): 
confettiStreamers (Confetti Streamers Art Border)	Specifies an art border consisting of a repeated image of confetti streamers, as follows (showing two repetitions): 
confettiWhite (Confetti Art Border)	Specifies an art border consisting of a repeated image of confetti, as follows (showing two repetitions): 
cornerTriangles (Corner Triangle Art Border)	Specifies an art border consisting of a repeated image of a line as follows: At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
couponCutoutDashes (Dashed Line Art Border)	Specifies an art border consisting of a dashed line, as follows: As well, this art border shall be rotated such that the

Enumeration Value	Description
	bottom of the image above is always nearest the text extents. If the top border is of this type, the border shall use the following image in the top left corner: 
couponCutoutDots (Dotted Line Art Border)	Specifies an art border consisting of a dotted line, as follows: As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. If the top border is of this type, the border shall use the following image in the top left corner:  If the bottom border is of this type, the border shall use the following image in the bottom right corner: 
crazyMaze (Maze Art Border)	Specifies an art border consisting of a repeated image of a maze-like pattern, as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image: 

Enumeration Value	Description
creaturesButterfly (Butterfly Art Border)	Specifies an art border consisting of a repeated image of a butterfly, as follows (showing two repetitions): 
creaturesFish (Fish Art Border)	Specifies an art border consisting of a repeated image of a fish, as follows (showing two repetitions): 
creaturesInsects (Insects Art Border)	Specifies an art border consisting of repeating images of insects. If the border is on the left or right, this image is as follows (showing two repetitions):  If the border is on the top or bottom, this image is as follows (showing two repetitions):  At any corner where two borders of this type intersect, the intersection shall use the following image: 
creaturesLadyBug (Ladybug Art Border)	Specifies an art border consisting of a repeated image of a ladybug, as follows (showing two repetitions): 
crossStitch (Cross-stitch Art Border)	Specifies an art border consisting of repeating images of a cross-stitch pattern, as follows (showing two repetitions): 

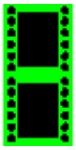
Enumeration Value	Description
cup (Cupid Art Border)	Specifies an art border consisting of a repeated image of Cupid, as follows (showing two repetitions): 
dashDotStroked (Dash Dot Strokes Line Border)	Specifies a line border consisting of a line with a series of alternating thin and thick strokes around the parent object. [*Example*: *end example*]
dashed (Dashed Line Border)	Specifies a line border consisting of a dashed line around the parent object. [*Example*: - - - - - - - - - - - - - - *end example*]
dashSmallGap (Dashed Line Border)	Specifies a line border consisting of a dashed line with small gaps around the parent object. [*Example*: ---------- *end example*]
decoArch (Archway Art Border)	Specifies an art border consisting of repeating images of an archway. If the border is on the left or right, this image is as follows (showing two repetitions):  If the border is on the top or bottom, this image is as follows (showing two repetitions):

Enumeration Value	Description
	 At any corner where two borders of this type intersect, the intersection shall use the following image: 
decoArchColor (Color Archway Art Border)	Specifies an art border consisting of repeating images of a color archway. If the border is on the left or right, this image is as follows (showing two repetitions):  If the border is on the top or bottom, this image is as follows (showing two repetitions):  At any corner where two borders of this type intersect, the intersection shall use the following image: 
decoBlocks (Blocks Art Border)	Specifies an art border consisting of repeating images of a series of blocks. If the border is on the left or right, this image is as follows (showing two repetitions):  If the border is on the top or bottom, this image is as

Enumeration Value	Description
	follows (showing two repetitions):  At any corner where two borders of this type intersect, the intersection shall use the following image, rotated accordingly: 
diamondsGray (Gray Diamond Art Border)	Specifies an art border consisting of a repeated image of diamonds, as follows (showing two repetitions): 
dotDash (Dot Dash Line Border)	Specifies a line border consisting of a alternating dotted and dashed line around the parent object. [*Example*: - - - - - - - - - - - - - - - *end example*]
dotDotDash (Dot Dot Dash Line Border)	Specifies a line border consisting of a alternating dotted, dotted, dashed line around the parent object. [*Example*: - - - - - - - - - - - - - - - - - - *end example*]
dotted (Dotted Line Border)	Specifies a line border consisting of a dotted line around the parent object. [*Example*: ..... *end example*]
double (Double Line Border)	Specifies a line border consisting of a double line around the parent object.

Enumeration Value	Description
	[*Example*: *end example*]
doubleD (Double D Art Border)	Specifies an art border consisting of repeating images of a pattern. If the border is on the left, this image is as follows (showing two repetitions):  If the border is on the right, this image is rotated as follows (showing two repetitions):  If the border is on the top, this image is rotated as follows (showing two repetitions):  If the border is on the bottom, this image is rotated as follows (showing two repetitions):  At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
doubleDiamonds (Diamond Art Border)	Specifies an art border consisting of a repeated image of a series of diamonds, as follows (showing two repetitions): 

Enumeration Value	Description
doubleWave (Double Wave Line Border)	Specifies a line border consisting of a double wavy line around the parent object. [*Example*:  *end example*]
earth1 (Earth Art Border)	Specifies an art border consisting of a repeated image of Earth, as follows (showing two repetitions): 
earth2 (Earth Art Border)	Specifies an art border consisting of a repeated image of Earth, as follows (showing two repetitions): 
eclipsingSquares1 (Shadowed Square Art Border)	Specifies an art border consisting of a repeated image of a shadowed square, as follows (showing two repetitions): 
eclipsingSquares2 (Shadowed Square Art Border)	Specifies an art border consisting of a repeated image of a shadowed square, as follows (showing two repetitions): 
eggsBlack (Painted Egg Art Border)	Specifies an art border consisting of a repeated image of a painted egg, as follows (showing two repetitions): 
fans (Fans Art Border)	Specifies an art border consisting of a repeated image of fans, as follows (showing two repetitions):

Enumeration Value	Description
	
film (Film Reel Art Border)	Specifies an art border consisting of repeating images of a film reel. If the border is on the left or right, this image is as follows (showing two repetitions):  If the border is on the top or bottom, this image is as follows (showing two repetitions):  At any corner where two borders of this type intersect, the intersection shall use the following image: 
firecrackers (Firecracker Art Border)	Specifies an art border consisting of a repeated image of a firecracker, as follows (showing two repetitions): 
flowersBlockPrint (Flowers Art Border)	Specifies an art border consisting of a repeated image of flowers, as follows (showing two repetitions): 
flowersDaisies (Daisy Art Border)	Specifies an art border consisting of a repeated image of a daisy, as follows (showing two repetitions): 
flowersModern1 (Flowers Art Border)	Specifies an art border consisting of a repeated image

Enumeration Value	Description
	of flowers, as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents.
flowersModern2 (Flowers Art Border)	Specifies an art border consisting of a repeated image of flowers, as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents.
flowersPansy (Pansy Art Border)	Specifies an art border consisting of a repeated image of a pansy, as follows (showing two repetitions): 
flowersRedRose (Red Rose Art Border)	Specifies an art border consisting of a repeated image of a red rose, as follows (showing two repetitions): 
flowersRoses (Roses Art Border)	Specifies an art border consisting of a repeated image of a rose, as follows (showing two repetitions): 
flowersTeacup (Flowers in a Teacup Art Border)	Specifies an art border consisting of a repeated image of flowers in a teacup, as follows (showing two repetitions): 
flowersTiny (Small Flower Art Border)	Specifies an art border consisting of a repeated image

Enumeration Value	Description
	of small flowers, as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents.
gems (Gems Art Border)	Specifies an art border consisting of a repeated image of a square pattern, as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image (shown from the left side - this image shall be flipped horizontally for the right side): 
gingerbreadMan (Gingerbread Man Art Border)	Specifies an art border consisting of a repeated image of a gingerbread man, as follows (showing two repetitions): 
gradient (Triangle Gradient Art Border)	Specifies an art border consisting of a repeated image of a triangle with a gradient pattern as follows (showing two repetitions): 
handmade1 (Handmade Art Border)	Specifies an art border consisting of an image with a handmade appearance. On the top (bottom is flipped vertically), this image shall be a stretched version of the following:

Enumeration Value	Description
	 On the left (right is flipped horizontally), this image shall be a stretched version of the following:
handmade2 (Handmade Art Border)	Specifies an art border consisting of an image with a handmade appearance. On the top (bottom is flipped vertically), this image shall be a stretched version of the following:  On the left (right is flipped horizontally), this image shall be a stretched version of the following:

Enumeration Value	Description
	
heartBalloon (Heart-Shaped Balloon Art Border)	Specifies an art border consisting of a repeated image of a heart-shaped balloon, as follows (showing two repetitions): 
heartGray (Gray Heart Art Border)	Specifies an art border consisting of a repeated image of a heart, as follows (showing two repetitions):
hearts (Hearts Art Border)	Specifies an art border consisting of a repeated image of hearts, as follows (showing two repetitions): 
heebieJeebies (Pattern Art Border)	Specifies an art border consisting of a repeated image

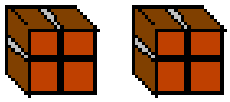
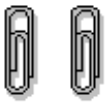
Enumeration Value	Description
	of a pattern as follows (showing two repetitions): 
holly (Holly Art Border)	Specifies an art border consisting of a repeated image of holly, as follows (showing two repetitions): 
houseFunky (House Art Border)	Specifies an art border consisting of a repeated image of a house, as follows (showing two repetitions): 
hypnotic (Circular Art Border)	Specifies an art border consisting of a repeated image of a series of circles, as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
iceCreamCones (Ice Cream Cone Art Border)	Specifies an art border consisting of a repeated image of an ice cream cone, as follows (showing two repetitions): 
inset (Inset Line Border)	Specifies a line border consisting of an inset set of lines around the parent object. [*Example*:

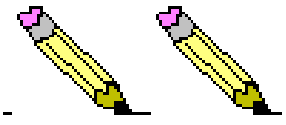
Enumeration Value	Description
	end example]
lightBulb (Light Bulb Art Border)	Specifies an art border consisting of a repeated image of a light bulb as follows (showing two repetitions): 
lightning1 (Lightning Art Border)	Specifies an art border consisting of a repeated image of lightning, as follows (showing two repetitions): 
lightning2 (Lightning Art Border)	Specifies an art border consisting of a repeated image of a lightning pattern, as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
mapleLeaf (Maple Leaf Art Border)	Specifies an art border consisting of a repeated image of a black and white image of a maple leaf, as follows (showing two repetitions): 
mapleMuffins (Muffin Art Border)	Specifies an art border consisting of a repeated image of a muffin, as follows (showing two repetitions):

Enumeration Value	Description
	
mapPins (Map Pins Art Border)	Specifies an art border consisting of a repeated image of a map pin, as follows (showing two repetitions): 
marquee (Marquee Art Border)	Specifies an art border consisting of a repeated image of a pattern as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
marqueeToothed (Marquee Art Border)	Specifies an art border consisting of a repeated image of a pattern as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
moons (Moon Art Border)	Specifies an art border consisting of repeating images of phases of the moon. If the border is on the top, this image is as follows (showing two repetitions): 

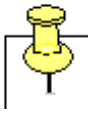
Enumeration Value	Description
	If the border is on the left, this image is as follows (showing two repetitions):  If the border is on the right, this image is as follows (showing two repetitions):  If the border is on the bottom, this image is as follows (showing two repetitions):  At the top-left corner where two borders of this type intersect, the intersection shall use the following image:  At the top-right corner where two borders of this type intersect, the intersection shall use the following image:  At the bottom-left corner where two borders of this type intersect, the intersection shall use the following image:  At the bottom-right corner where two borders of this

Enumeration Value	Description
	type intersect, the intersection shall use the following image: 
mosaic (Mosaic Art Border)	Specifies an art border consisting of a repeated image of a mosaic pattern, as follows (showing two repetitions): 
musicNotes (Musical Note Art Border)	Specifies an art border consisting of a repeated image of a musical note, as follows (showing two repetitions): 
nil (No Border)	Specifies that no border shall be applied to the current item.
none (No Border)	Specifies that no border shall be applied to the current item.
northwest (Patterned Art Border)	Specifies an art border consisting of a repeated image of a pattern, as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image: 
outset (Outset Line Border)	Specifies a line border consisting of an outset set of lines around the parent object. [*Example*: *end example*]
ovals (Oval Art Border)	Specifies an art border consisting of a repeated image

Enumeration Value	Description
	of a series of ovals, as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image: 
packages (Package Art Border)	Specifies an art border consisting of a repeated image of a package, as follows (showing two repetitions): 
palmsBlack (Black Palm Tree Art Border)	Specifies an art border consisting of a repeated image of a black and white palm tree, as follows (showing two repetitions): 
palmsColor (Color Palm Tree Art Border)	Specifies an art border consisting of a repeated image of a color palm tree, as follows (showing two repetitions): 
paperClips (Paper Clip Art Border)	Specifies an art border consisting of a repeated image of a paper clip, as follows (showing two repetitions): 

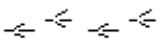
Enumeration Value	Description
papyrus (Papyrus Art Border)	Specifies an art border consisting of repeating images of ovals and an art corner. If the border is on the left or right, this image is as follows (showing two repetitions):  If the border is on the top or bottom, this image is as follows (showing two repetitions):  At any corner where two borders of this type intersect, the intersection shall use the following image: 
partyFavor (Party Favor Art Border)	Specifies an art border consisting of a repeated image of a party favor, as follows (showing two repetitions): 
partyGlass (Party Glass Art Border)	Specifies an art border consisting of a repeated image of a party glass, as follows (showing two repetitions): 
pencils (Pencils Art Border)	Specifies an art border consisting of a repeated image of a pencil, as follows (showing two repetitions): 

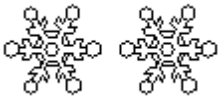
Enumeration Value	Description
people (Character Art Border)	Specifies an art border consisting of a repeated image of a character, as follows (showing two repetitions): 
peopleHats (Character With Hat Art Border)	Specifies an art border consisting of a repeated image of a character with a hat, as follows (showing two repetitions): 
peopleWaving (Waving Character Border)	Specifies an art border consisting of a repeated image of a character waving, as follows (showing two repetitions): 
poinsettias (Poinsettia Art Border)	Specifies an art border consisting of a repeated image of a poinsettia, as follows (showing two repetitions): 
postageStamp (Postage Stamp Art Border)	Specifies an art border consisting of repeating images of a stamp-like pattern, as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
pumpkin1 (Pumpkin Art Border)	Specifies an art border consisting of a repeated image of a pumpkin, as follows (showing two repetitions):

Enumeration Value	Description
	
pushPinNote1 (Push Pin Art Border)	Specifies an art border consisting of a black line, as follows: _____ As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. If the top border is of this type, the border shall use the following image in the top left corner: 
pushPinNote2 (Push Pin Art Border)	Specifies an art border consisting of a repeated image of a black line. If the border is on the top, left or right, then the image is as follows (showing two repetitions): _____ If the border is on the bottom, then the image is as follows (showing two repetitions):  If the top border is of this type, the border shall use the following image in the top left corner: 
pyramids (Pyramid Art Border)	Specifies an art border consisting of repeating images of a pyramid pattern, as follows (showing two repetitions):  As well, this art border shall be rotated such that the

Enumeration Value	Description
	bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
pyramidsAbove (Pyramid Art Border)	Specifies an art border consisting of a repeated image of a pyramid viewed from above, as follows (showing two repetitions): 
quadrants (Quadrants Art Border)	Specifies an art border consisting of a repeating image of a colored pattern. If the border is on the left or right, this image is as follows (showing two repetitions):  If the border is on the top or bottom, this image is as follows (showing two repetitions):  At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
rings (Rings Art Border)	Specifies an art border consisting of a repeated image of a ring, as follows (showing two repetitions): 
safari (Safari Art Border)	Specifies an art border consisting of a repeated image of a print pattern, as follows (showing two repetitions):

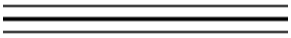
Enumeration Value	Description
	
sawtooth (Saw tooth Art Border)	Specifies an art border consisting of repeating images of a saw tooth pattern, as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
sawtoothGray (Gray Saw tooth Art Border)	Specifies an art border consisting of repeating images of a saw tooth pattern, as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
scaredCat (Scared Cat Art Border)	Specifies an art border consisting of a repeated image of a frightened cat, as follows (showing two repetitions): 
seattle (Umbrella Art Border)	Specifies an art border consisting of a repeated image

Enumeration Value	Description
	of an umbrella, as follows (showing two repetitions): 
shadowedSquares (Shadowed Squares Art Border)	Specifies an art border consisting of a repeated image of squares with a drop shadow, as follows (showing two repetitions): 
sharksTeeth (Shark Tooth Art Border)	Specifies an art border consisting of a repeating image of a shark tooth pattern. If the border is on the left or right, this image is as follows (showing two repetitions):  If the border is on the top or bottom, this image is as follows (showing two repetitions on the top, the bottom shall be rotated 180 degrees):  At any corner where two borders of this type intersect, the intersection shall use the left and right border.
shorebirdTracks (Bird Tracks Art Border)	Specifies an art border consisting of a repeated image of bird tracks, as follows  As well, this art border shall be rotated such that the bottom of each image is always nearest the text extents.
single (Single Line Border)	Specifies a line border consisting of a single line around the parent object. [Example: _____

Enumeration Value	Description
	end example]
skyrocket (Rocket Art Border)	Specifies an art border consisting of a repeated image of a rocket, as follows (showing two repetitions): 
snowflakeFancy (Snowflake Art Border)	Specifies an art border consisting of a repeated image of a snowflake, as follows (showing two repetitions): 
snowflakes (Snowflake Art Border)	Specifies an art border consisting of a repeated image of a snowflake, as follows (showing two repetitions): 
sombrero (Sombrero Art Border)	Specifies an art border consisting of a repeated image of a sombrero, as follows (showing two repetitions): 
southwest (Southwest-themed Art Border)	Specifies an art border consisting of a repeated image of a pattern as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image: 
stars (Stars Art Border)	Specifies an art border consisting of a repeated image of stars, as follows (showing two repetitions):

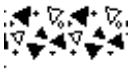
Enumeration Value	Description
	
stars3d (3-D Stars Art Border)	Specifies an art border consisting of a repeated image of three-dimensional stars, as follows (showing two repetitions): 
starsBlack (Stars Art Border)	Specifies an art border consisting of a repeated image of black stars, as follows (showing three repetitions): 
starsShadowed (Stars With Shadows Art Border)	Specifies an art border consisting of a repeated image of stars with a shadow effect, as follows (showing two repetitions): 
starsTop (Stars On Top Art Border)	Specifies an art border consisting of repeating images of stars on the top of the page. If the border is on the left or right, no border is displayed. If the border is on the top, this image is as follows:  If the border is on the bottom, this image is as follows: 
sun (Sun Art Border)	Specifies an art border consisting of a repeated image of the sun, as follows (showing two repetitions): 
swirligig (Whirligig Art Border)	Specifies an art border consisting of a repeated image as follows (showing two repetitions):

Enumeration Value	Description
	
thick (Single Line Border)	Specifies a line border consisting of a single line around the parent object. [*Example*: *end example*]
thickThinLargeGap (Thick, Thin Line Border)	Specifies a line border consisting of a thick line contained within a thin line with a large sized intermediate gap around the parent object. [*Example*:  *end example*]
thickThinMediumGap (Thick, Thin Line Border)	Specifies a line border consisting of a thick line contained within a thin line with a medium sized intermediate gap around the parent object. [*Example*:  *end example*]
thickThinSmallGap (Thick, Thin Line Border)	Specifies a line border consisting of a thick line contained within a thin line with a small intermediate gap around the parent object. [*Example*:  *end example*]
thinThickLargeGap (Thin, Thick Line Border)	Specifies a line border consisting of a thin line contained within a thick line contained within a thick thin with a large sized intermediate gap between each around the parent object. [*Example*: 

Enumeration Value	Description
	end example]
thinThickMediumGap (Thin, Thick Line Border)	Specifies a line border consisting of a thin line contained within a thick line contained within a thick thin with a medium sized intermediate gap between each around the parent object. [*Example*:  *end example*]
thinThickSmallGap (Thin, Thick Line Border)	Specifies a line border consisting of a thin line contained within a thick line contained within a thick thin with a small intermediate gap between each around the parent object. [*Example*:  *end example*]
thinThickThinLargeGap (Thin, Thick, Thin Line Border)	Specifies a line border consisting of a thin line contained within a thick line, contained within a thin line with a large sized intermediate gap around the parent object. [*Example*:  *end example*]
thinThickThinMediumGap (Thin, Thick, Thin Line Border)	Specifies a line border consisting of a thin line contained within a thick line, contained within a thin line with a medium sized intermediate gap around the parent object. [*Example*:  *end example*]

Enumeration Value	Description
thinThickThinSmallGap (Thin, Thick, Thin Line Border)	Specifies a line border consisting of a thin line contained within a thick line, contained within a thin line with a small intermediate gap around the parent object. [*Example*:  *end example*]
threeDEmboss (3D Embossed Line Border)	Specifies a line border consisting of three staged gradient lines around the parent object, getting darker towards the object. [*Example*:  *end example*]
threeDEngrave (3D Engraved Line Border)	Specifies a line border consisting of three staged gradient lines around the parent object, getting darker away from the object. [*Example*:  *end example*]
tornPaper (Torn Paper Art Border)	Specifies an art border consisting of repeating images of stars on the top of the page. If the border is on the left or right, no border is displayed. If the border is on the top or bottom, this image is as follows (shown on top, flipped vertically on bottom): 
tornPaperBlack (Black Torn Paper Art Border)	Specifies an art border consisting of an image with a torn appearance. On the top (bottom border shall be flipped vertically),

Enumeration Value	Description
	this image shall be a stretched version of the following:  On the left (the right side shall be flipped horizontally), this image shall be a stretched version of the following: 
trees (Tree Art Border)	Specifies an art border consisting of a repeated image of a tree, as follows (showing two repetitions): 
triangleParty (Triangle Art Border)	Specifies an art border consisting of a repeated image of triangles, as follows (showing two repetitions):

Enumeration Value	Description
	
triangles (Triangles Art Border)	Specifies an art border consisting of a repeating image of a triangle pattern. If the border is on the left or right, this image is as follows (showing two repetitions on the left, the right shall be rotated 180 degrees):  If the border is on the top or bottom, this image is as follows (showing two repetitions on the top, the bottom shall be rotated 180 degrees):  At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
tribal1 (Tribal Art Border One)	Specifies an art border consisting of a repeated image of a tribal pattern as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image: 
tribal2 (Tribal Art Border Two)	Specifies an art border consisting of a repeated image

Enumeration Value	Description
	of a tribal pattern as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
tribal3 (Tribal Art Border Three)	Specifies an art border consisting of a repeated image of a tribal pattern as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
tribal4 (Tribal Art Border Four)	Specifies an art border consisting of a repeated image of a tribal pattern as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
tribal5 (Tribal Art Border Five)	Specifies an art border consisting of a repeating image

Enumeration Value	Description
	of a tribal pattern. If the border is on the left or right, this image is as follows (showing two repetitions):  If the border is on the top or bottom, this image is as follows (showing two repetitions):  At any corner where two borders of this type intersect, the intersection shall use the following image: 
tribal6 (Tribal Art Border Six)	Specifies an art border consisting of a repeated image of a tribal pattern as follows (showing two repetitions):  At any corner where two borders of this type intersect, the intersection shall use the following image: 
triple (Triple Line Border)	Specifies a line border consisting of a triple line around the parent object. [*Example*: *end example*]
twistedLines1 (Twisted Lines Art Border)	Specifies an art border consisting of a repeated image of twisted lines, as follows (showing two repetitions): 

Enumeration Value	Description
	As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
twistedLines2 (Twisted Lines Art Border)	Specifies an art border consisting of a repeated image of twisted lines, as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
vine (Vine Art Border)	Specifies an art border consisting of a repeating image of a vine pattern. If the border is on the left or right, this image is as follows (showing two repetitions):  If the border is on the top or bottom, this image is as follows (showing two repetitions):

Enumeration Value	Description
	 At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
wave (Wavy Line Border)	Specifies a line border consisting of a wavy line around the parent object. [*Example*: *end example*]
waveline (Wavy Line Art Border)	Specifies an art border consisting of a repeated image of wavy lines, as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
weavingAngles (Weaving Angles Art Border)	Specifies an art border consisting of a repeated image of weaving angles, as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents.

Enumeration Value	Description
	At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
weavingBraid (Weaving Braid Art Border)	Speci Specifies an art border consisting of repeating images of a weaving pattern, as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image: 
weavingRibbon (Weaving Ribbon Art Border)	Specifies an art border consisting of repeating images of a weaving ribbon, as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image, rotated accordingly: 
weavingStrips (Weaving Strips Art Border)	Specifies an art border consisting of repeating images of weaving strips, as follows (showing two repetitions): 

Enumeration Value	Description
	As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image, rotated accordingly: 
whiteFlowers (White Flowers Art Border)	Specifies an art border consisting of a repeating image of white flowers. If the border is on the left or right, this image is as follows (showing two repetitions):  If the border is on the top or bottom, this image is as follows (showing two repetitions):
woodwork (Woodwork Art Border)	Specifies an art border consisting of repeating images of a woodwork pattern, as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image: 

Enumeration Value	Description
xIllusions (Crisscross Art Border)	Specifies an art border consisting of repeating images of a crisscross pattern, as follows (showing two repetitions): 
zanyTriangles (Triangle Art Border)	Specifies an art border consisting of repeating images of a triangle pattern, as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image, rotated appropriately: 
zigZag (Zigzag Art Border)	Specifies an art border consisting of repeating images of a zigzag pattern, as follows (showing two repetitions):  As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image, rotated accordingly: 
zigZagStitch (Zigzag stitch)	Specifies an art border consisting of a repeating image of a zigzag pattern, as follows (showing two repetitions):

Enumeration Value	Description
	 As well, this art border shall be rotated such that the bottom of the image above is always nearest the text extents. At any corner where two borders of this type intersect, the intersection shall use the following image, rotated accordingly: 

Referenced By
bar@val (§2.3.1.4); bdr@val (§2.3.2.3); between@val (§2.3.1.5); bottom@val (§2.6.2); bottom@val (§2.4.3); bottom@val (§2.4.4); bottom@val (§2.15.2.4); bottom@val (§2.3.1.7); insideH@val (§2.4.17); insideH@val (§2.4.18); insideV@val (§2.4.19); insideV@val (§2.4.20); left@val (§2.15.2.21); left@val (§2.4.24); left@val (§2.6.7); left@val (§2.3.1.17); left@val (§2.4.27); right@val (§2.3.1.28); right@val (§2.4.30); right@val (§2.6.15); right@val (§2.15.2.35); right@val (§2.4.32); tl2br@val (§2.4.70); top@val (§2.4.71); top@val (§2.3.1.42); top@val (§2.15.2.42); top@val (§2.4.74); top@val (§2.6.21); tr2bl@val (§2.4.76)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Border">
   <restriction base="xsd:string">
      <enumeration value="nil"/>
      <enumeration value="none"/>
      <enumeration value="single"/>
      <enumeration value="thick"/>
      <enumeration value="double"/>
      <enumeration value="dotted"/>
      <enumeration value="dashed"/>
      <enumeration value="dotDash"/>
      <enumeration value="dotDotDash"/>
      <enumeration value="triple"/>
      <enumeration value="thinThickSmallGap"/>
      <enumeration value="thickThinSmallGap"/>
      <enumeration value="thinThickThinSmallGap"/>
      <enumeration value="thinThickMediumGap"/>
      <enumeration value="thickThinMediumGap"/>
      <enumeration value="thinThickThinMediumGap"/>
      <enumeration value="thinThickLargeGap"/>
      <enumeration value="thickThinLargeGap"/>
      <enumeration value="thinThickThinLargeGap"/>
      <enumeration value="wave"/>
      <enumeration value="doubleWave"/>
      <enumeration value="dashSmallGap"/>
      <enumeration value="dashDotStroked"/>
      <enumeration value="threeDEmboss"/>
      <enumeration value="threeDEngrave"/>
      <enumeration value="outset"/>
      <enumeration value="inset"/>
      <enumeration value="apples"/>
      <enumeration value="archedScallops"/>
      <enumeration value="babyPacifier"/>
      <enumeration value="babyRattle"/>
      <enumeration value="balloons3Colors"/>
      <enumeration value="balloonsHotAir"/>
      <enumeration value="basicBlackDashes"/>
      <enumeration value="basicBlackDots"/>
      <enumeration value="basicBlackSquares"/>
      <enumeration value="basicThinLines"/>
      <enumeration value="basicWhiteDashes"/>
      <enumeration value="basicWhiteDots"/>
      <enumeration value="basicWhiteSquares"/>
      <enumeration value="basicWideInline"/>
      <enumeration value="basicWideMidline"/>
      <enumeration value="basicWideOutline"/>
      <enumeration value="bats"/>
      <enumeration value="birds"/>
      <enumeration value="birdsFlight"/>
      <enumeration value="cabins"/>
      <enumeration value="cakeSlice"/>
      <enumeration value="candyCorn"/>
      <enumeration value="celticKnotwork"/>
```

```
        <enumeration value="certificateBanner"/>
        <enumeration value="chainLink"/>
        <enumeration value="champagneBottle"/>
        <enumeration value="checkedBarBlack"/>
        <enumeration value="checkedBarColor"/>
        <enumeration value="checkered"/>
        <enumeration value="christmasTree"/>
        <enumeration value="circlesLines"/>
        <enumeration value="circlesRectangles"/>
        <enumeration value="classicalWave"/>
        <enumeration value="clocks"/>
        <enumeration value="compass"/>
        <enumeration value="confetti"/>
        <enumeration value="confettiGrays"/>
        <enumeration value="confettiOutline"/>
        <enumeration value="confettiStreamers"/>
        <enumeration value="confettiWhite"/>
        <enumeration value="cornerTriangles"/>
        <enumeration value="couponCutoutDashes"/>
        <enumeration value="couponCutoutDots"/>
        <enumeration value="crazyMaze"/>
        <enumeration value="creaturesButterfly"/>
        <enumeration value="creaturesFish"/>
        <enumeration value="creaturesInsects"/>
        <enumeration value="creaturesLadyBug"/>
        <enumeration value="crossStitch"/>
        <enumeration value="cup"/>
        <enumeration value="decoArch"/>
        <enumeration value="decoArchColor"/>
        <enumeration value="decoBlocks"/>
        <enumeration value="diamondsGray"/>
        <enumeration value="doubleD"/>
        <enumeration value="doubleDiamonds"/>
        <enumeration value="earth1"/>
        <enumeration value="earth2"/>
        <enumeration value="eclipsingSquares1"/>
        <enumeration value="eclipsingSquares2"/>
        <enumeration value="eggsBlack"/>
        <enumeration value="fans"/>
        <enumeration value="film"/>
        <enumeration value="firecrackers"/>
        <enumeration value="flowersBlockPrint"/>
        <enumeration value="flowersDaisies"/>
        <enumeration value="flowersModern1"/>
        <enumeration value="flowersModern2"/>
        <enumeration value="flowersPansy"/>
        <enumeration value="flowersRedRose"/>
        <enumeration value="flowersRoses"/>
        <enumeration value="flowersTeacup"/>
        <enumeration value="flowersTiny"/>
        <enumeration value="gems"/>
        <enumeration value="gingerbreadMan"/>
        <enumeration value="gradient"/>
```

```
        <enumeration value="handmade1"/>
        <enumeration value="handmade2"/>
        <enumeration value="heartBalloon"/>
        <enumeration value="heartGray"/>
        <enumeration value="hearts"/>
        <enumeration value="heebieJeebies"/>
        <enumeration value="holly"/>
        <enumeration value="houseFunky"/>
        <enumeration value="hypnotic"/>
        <enumeration value="iceCreamCones"/>
        <enumeration value="lightBulb"/>
        <enumeration value="lightning1"/>
        <enumeration value="lightning2"/>
        <enumeration value="mapPins"/>
        <enumeration value="mapleLeaf"/>
        <enumeration value="mapleMuffins"/>
        <enumeration value="marquee"/>
        <enumeration value="marqueeToothed"/>
        <enumeration value="moons"/>
        <enumeration value="mosaic"/>
        <enumeration value="musicNotes"/>
        <enumeration value="northwest"/>
        <enumeration value="ovals"/>
        <enumeration value="packages"/>
        <enumeration value="palmsBlack"/>
        <enumeration value="palmsColor"/>
        <enumeration value="paperClips"/>
        <enumeration value="papyrus"/>
        <enumeration value="partyFavor"/>
        <enumeration value="partyGlass"/>
        <enumeration value="pencils"/>
        <enumeration value="people"/>
        <enumeration value="peopleWaving"/>
        <enumeration value="peopleHats"/>
        <enumeration value="poinsettias"/>
        <enumeration value="postageStamp"/>
        <enumeration value="pumpkin1"/>
        <enumeration value="pushPinNote2"/>
        <enumeration value="pushPinNote1"/>
        <enumeration value="pyramids"/>
        <enumeration value="pyramidsAbove"/>
        <enumeration value="quadrants"/>
        <enumeration value="rings"/>
        <enumeration value="safari"/>
        <enumeration value="sawtooth"/>
        <enumeration value="sawtoothGray"/>
        <enumeration value="scaredCat"/>
        <enumeration value="seattle"/>
        <enumeration value="shadowedSquares"/>
        <enumeration value="sharksTeeth"/>
        <enumeration value="shorebirdTracks"/>
        <enumeration value="skyrocket"/>
        <enumeration value="snowflakeFancy"/>
```

```
        <enumeration value="snowflakes"/>
        <enumeration value="sombrero"/>
        <enumeration value="southwest"/>
        <enumeration value="stars"/>
        <enumeration value="starsTop"/>
        <enumeration value="stars3d"/>
        <enumeration value="starsBlack"/>
        <enumeration value="starsShadowed"/>
        <enumeration value="sun"/>
        <enumeration value="swirligig"/>
        <enumeration value="tornPaper"/>
        <enumeration value="tornPaperBlack"/>
        <enumeration value="trees"/>
        <enumeration value="triangleParty"/>
        <enumeration value="triangles"/>
        <enumeration value="tribal1"/>
        <enumeration value="tribal2"/>
        <enumeration value="tribal3"/>
        <enumeration value="tribal4"/>
        <enumeration value="tribal5"/>
        <enumeration value="tribal6"/>
        <enumeration value="twistedLines1"/>
        <enumeration value="twistedLines2"/>
        <enumeration value="vine"/>
        <enumeration value="waveline"/>
        <enumeration value="weavingAngles"/>
        <enumeration value="weavingBraid"/>
        <enumeration value="weavingRibbon"/>
        <enumeration value="weavingStrips"/>
        <enumeration value="whiteFlowers"/>
        <enumeration value="woodwork"/>
        <enumeration value="xIllusions"/>
        <enumeration value="zanyTriangles"/>
        <enumeration value="zigZag"/>
        <enumeration value="zigZagStitch"/>
    </restriction>
</simpleType>
```

2.18.5 ST_BrClear (Line Break Text Wrapping Restart Location)

This simple type specifies the set of possible restart locations which may be used as to determine the next available line when a break's type attribute has a value of `textWrapping`. This property only affects the restart location when the current run is being displayed on a line which does not span the full text extents due to the presence of a floating object (see enumeration values for details).

[*Example*: Consider a text wrapping break character which should force the restart location to the next line which spans the full width of the text extents of the page (there are no floating objects which interrupt the line).

This line break is of type `textWrapping`, since it shall only advance to the next line, but the clear value shall specify that this restart location shall ignore all lines which are not of the full line width by specifying a value of `all`, as follows:

```
<w:br w:type="textWrapping" w:clear="all" />
```

This break shall therefore not use the next available line, but rather the next available line ignoring all lines which do not span the full text width. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
all (Restart On Next Full Line)	Specifies that the text wrapping break shall advance the text to the next line in the WordprocessingML document which spans the full width of the line (i.e. the next line which is not interrupted by any floating objects when those objects are positioned on the page at display time. [*Note*: This setting is typically used to place a single line of text next to a floating object for use as a caption. *end note*]
left (Restart In Next Text Region When In Leftmost Position)	Specifies that the text wrapping break shall behave as follows: If this line is broken into multiple regions (a floating object in the center of the page has text wrapping on both sides: • If this is the leftmost region of text flow on this line, advance the text to the next position on the line • Otherwise, treat this as a text wrapping break of type `all`. If this line is not broken into multiple regions, then treat this break as a text wrapping break of type none. If the parent paragraph is right to left, then these behaviors are also reversed. [*Note*: This break type is used to control the text wrapping on the left side of a floating image without preventing text from appearing on the opposite side. *end note*]
none (Restart On Next Line)	Specifies that the text wrapping break shall advance the text to the next line in the WordprocessingML document, regardless of its position left to right or the presence of any floating objects which intersect with the line,

Enumeration Value	Description
	This is the setting for a typical line break in a document.
right (Restart In Next Text Region When In Rightmost Position)	Specifies that the text wrapping break shall behave as follows: If this line is broken into multiple regions (a floating object in the center of the page has text wrapping on both sides: • If this is the rightmost region of text flow on this line, advance the text to the next position on the next line • Otherwise, treat this as a text wrapping break of type `all`. If this line is not broken into multiple regions, then treat this break as a text wrapping break of type none. If the parent paragraph is right to left, then these behaviors are also reversed. [*Note*: This break type is used to control the text wrapping on the right side of a floating image without preventing text from appearing on the opposite side. *end note*]

Referenced By
br@clear (§2.3.3.1)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_BrClear">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="left"/>
      <enumeration value="right"/>
      <enumeration value="all"/>
   </restriction>
</simpleType>
```

2.18.6 ST_BrType (Break Types)

This simple type specifies the possible types of break characters in a WordprocessingML document. The break type determines the next location where text shall be placed after this manual break is applied to the text contents (see enumeration values for details).

[*Example*: Consider a manual break which shall advance the text to the next text column in the document, rather than just the next available line. This break would therefore be specified as follows:

```
<w:br w:type=”column”/>
```

The type attribute specifies a value of `column`, which means that the break shall force the next character in the document to be restarted on the next line in a new text column in the document. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
column (Column Break)	Specifies that the current break shall restart itself on the next column available on the current page when the document is displayed in page view. If the current section is not divided into columns, or the column break occurs in the last column on the current page when displayed, then the restart location for text shall be the next page in the document.
page (Page Break)	Specifies that the current break shall restart itself on the next page of the document when the document is displayed in page view. Page breaks shall be ignored when present in frames unless the showBreaksInFrames element (§2.15.3.42) is present in the document's compatibility settings.
textWrapping (Line Break)	Specifies that the current break shall restart itself on the next line in the document when the document is displayed in page view. The determine of the next line shall be done subject to the value of the clear attribute on the specified break character.

Referenced By
br@type (§2.3.3.1)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_BrType">
   <restriction base="xsd:string">
      <enumeration value="page"/>
      <enumeration value="column"/>
      <enumeration value="textWrapping"/>
   </restriction>
</simpleType>
```

2.18.7 ST_CalendarType (Calendar Types)

This simple type specifies the possible types of calendars which may be used within the context of a WordprocessingML document.

[*Example*: Consider the following structured document tag properties:

```
<w:sdtPr>
  <w:date w:fullDate="01-01-2006T06:30:00Z">
    <w:calendar w:val="gregorian"/>
  </w:date>
</w:sdtPr>
```

The calendar element specifies that the calendar type for a calendar which may be displayed in the document shall be the Gregorian calendar format (gregorian). *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
gregorian (Gregorian)	Specifies that the Gregorian calendar shall be used. This calendar may be localized into the appropriate language as desired.
gregorianXlitEnglish (Gregorian transliterated English)	Specifies that the Gregorian transliterated English calendar shall be used.
gregorianXlitFrench (Gregorian transliterated French)	Specifies that the Gregorian transliterated French calendar shall be used.
hebrew (Hebrew)	Specifies that the Hebrew lunar calendar shall be used.
hijri (Hijri)	Specifies that the Hijri lunar calendar shall be used.
japan (Japanese Emperor Era)	Specifies that the Japanese Emperor Era calendar shall be used.
korea (Korean Tangun Era)	Specifies that the Korean Tangun Era calendar shall be used.
saka (Saka Era)	Specifies that the Saka Era calendar shall be used.
taiwan (Taiwan)	Specifies that the Taiwanese calendar shall be used.

Enumeration Value	Description
thai (Thai)	Specifies that the Thai calendar shall be used.

Referenced By
calendar@val (§2.5.2.3)

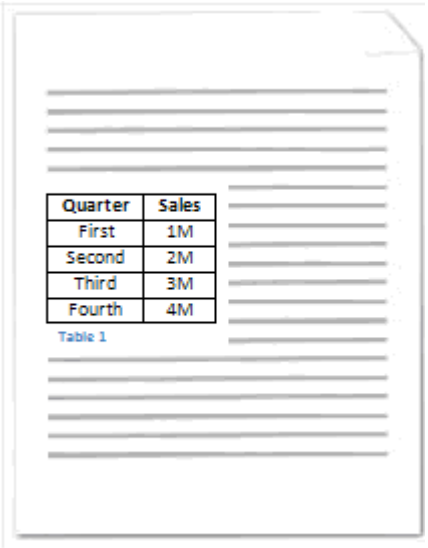
The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_CalendarType">
    <restriction base="xsd:string">
        <enumeration value="gregorian"/>
        <enumeration value="hijri"/>
        <enumeration value="hebrew"/>
        <enumeration value="taiwan"/>
        <enumeration value="japan"/>
        <enumeration value="thai"/>
        <enumeration value="korea"/>
        <enumeration value="saka"/>
        <enumeration value="gregorianXlitEnglish"/>
        <enumeration value="gregorianXlitFrench"/>
    </restriction>
</simpleType>
```

2.18.8 ST_CaptionPos (Automatic Caption Positioning Values)

This simple type specifies the possible values may be used for the position of an automatically inserted caption on an object within this document. These values specify the position a given caption shall be take relative to the object it is used to label.

[*Example*: Consider a WordprocessingML document which should have all automatically inserted captions placed below the objects they are captioning, for example:



This requirement is specified using the following WordprocessingML in the document settings:

```
<w:captions>
  <w:caption w:name="Table" w:pos="below" w:numFmt="decimal" />
</w:captions>
```

The pos attribute has a value of below, specifying that the caption shall be placed below the newly inserted objects. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
above (Position Caption Above Object)	Specifies that an automatically inserted caption shall be positioned above the object that it is used to label.
below (Position Caption Below Object)	Specifies that an automatically inserted caption shall be positioned below the object that it is used to label.
left (Position Caption Left Of Object)	Specifies that an automatically inserted caption shall be positioned to the left of the object that it is used to label (the position where text typed immediately before the object would appear).
right (Position Caption Right Of Object)	Specifies that an automatically inserted caption shall be positioned to the right of the object that it is used to label (the position where text typed immediately after the object would appear).

Referenced By
caption@pos (§2.15.1.16)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_CaptionPos">
   <restriction base="xsd:string">
      <enumeration value="above"/>
      <enumeration value="below"/>
      <enumeration value="left"/>
      <enumeration value="right"/>
   </restriction>
</simpleType>
```

2.18.9 ST_ChapterSep (Chapter Separator Types)

This simple type specifies the character which shall be used to separate the chapter number from the page number for page numbers in a given section, when chapter numbers are being displayed.

[*Example*: Consider a section in a document in which the chapter shall be separated from the page number using a colon character. This constraint would be specified using the following WordprocessingML:

```
<w:pgNumType w:chapSep="colon" chapStyle="1" />
```

The chapSep attribute declares that the chapter and page number shall be separated by a colon (e.g. 1:1 for chapter one, page one). *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
colon (Colon Chapter Separator)	Specifies that a colon character shall be used to separate the chapter number from the page number when page numbers are displayed. [Example: 1:1 for page one, section one. end example]
emDash (Em Dash Chapter Separator)	Specifies that an em dash character shall be used to separate the chapter number from the page number when page numbers are displayed. [*Example*: 1—1 for page one, section one. *end example*]
enDash (En Dash Chapter Separator)	Specifies that an en dash character shall be used to separate the chapter number from the page number when page numbers are displayed. [*Example*: 1–1 for page one, section one. *end example*]
hyphen (Hyphen Chapter Separator)	Specifies that a non-breaking hyphen character shall be used to separate the chapter number from the page number when page numbers are displayed. [*Example*: 1-1 for page one, section one. *end example*]
period (Period Chapter Separator)	Specifies that a period character shall be used to separate the chapter number from the page number when page numbers are displayed. [*Example*: 1.1 for page one, section one. *end example*]

Referenced By
caption@sep (§2.15.1.16); pgNumType@chapSep (§2.6.12)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ChapterSep">
   <restriction base="xsd:string">
      <enumeration value="hyphen"/>
      <enumeration value="period"/>
      <enumeration value="colon"/>
      <enumeration value="emDash"/>
      <enumeration value="enDash"/>
   </restriction>
</simpleType>
```

2.18.10 ST_CharacterSpacing (Character-Level Whitespace Compression Settings)

This simple type specifies the possible ways in which full-width characters in the current WordprocessingML document may be compressed to remove additional whitespace when the contents of this document are displayed, specifically by specifying the set(s) of characters which may be compressed to remove additional whitespace.

[*Example*: Consider the WordprocessingML below:

```
<w:characterSpacingControl w:val="dontCompress" />
```

The characterSpacingControl element has a val attribute value of `dontCompress`, which specifies that no character compression shall be applied to any character when the document is displayed. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
compressPunctuation (Compress Whitespace From Punctuation Characters)	Specifies that only whitespace characters shall have whitespace compression applied to them.
compressPunctuationAndJapaneseKana (Compress Whitespace From Both Japanese Kana And Punctuation Characters)	Specifies that whitespace and Japanese kana characters shall have whitespace compression applied to them.
doNotCompress (Do Not Compress Whitespace)	Specifies that characters shall not have whitespace compression applied to them.

Referenced By
characterSpacingControl@val (§2.15.1.18)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_CharacterSpacing">
   <restriction base="xsd:string">
      <enumeration value="doNotCompress"/>
      <enumeration value="compressPunctuation"/>
      <enumeration value="compressPunctuationAndJapaneseKana"/>
   </restriction>
</simpleType>
```

2.18.11 ST_Cnf (Conditional Formatting Bitmask)

This simple type specifies the format for the set of conditional formatting properties that have been applied to this object.

These properties are expressed using a string serialization of a binary bitmask for each of the following properties (reading from the first character position right):

- First Row - Is this the first row of the table?
- Last Row - Is this the last row of the table?
- First Column - Does this belong to the first column of the table?
- Last Column - Does this belong to the last column of the table?
- Band 1 Vertical - Does this belong to a column which should receive band 1 formatting? This property specifies whether the cell should receive the formatting specified for odd-numbered columns (e.g. 1,3,5,...)
- Band 2 Vertical - Does this belong to a column which should receive band 2 formatting? This property specifies whether the cell should receive the formatting specified for even-numbered columns (e.g. 2,4,6...)
- Band 1 Horizontal - Does this receive band 1 formatting? This property specifies whether the cell should receive the formatting specified for odd-numbered rows (e.g. 1,3,5,...)
- Band 2 Horizontal - Does this receive band 2 formatting? This property specifies whether the cell should receive the formatting specified for even-numbered rows (e.g. 2,4,6...)
- NE Cell - Is this part of the top-right corner of the table?
- NW Cell - Is this part of the top-left corner of the table?
- SE Cell - Is this part of the bottom-right corner of the table?
- SW Cell - Is this part of the bottom-left corner of the table?

For each of these properties, a value of `1` in the specified character position in the string means that the value is true, a value of `0` means false. All values must be specified.

[*Example*: Consider a paragraph in the top right corner of a table with a table style applied. This paragraph would need to specify the following WordprocessingML:

```
<w:p>
  <w:pPr>
    <w:cnfStyle w:val="101000000100" />
```

```
      …
    </w:pPr>
    …
  </w:p>
```

This paragraph specifies that it has the conditional properties from the table style for the first column, first row, and the NW corner of the parent table by setting the appropriate bits in the val attribute. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

This simple type also specifies the following restrictions:

- This simple type's contents must have a length of exactly 12 characters.
- This simple type's contents must match the following regular expression pattern: [01]*.

Referenced By
cnfStyle@val (§2.3.1.8); cnfStyle@val (§2.4.7); cnfStyle@val (§2.4.8)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Cnf">
   <restriction base="xsd:string">
      <length value="12"/>
      <pattern value="[01]*"/>
   </restriction>
</simpleType>
```

2.18.12 ST_ColorSchemeIndex (Theme Color Reference)

This simple type specifies the possible set of theme color stored in the document's Theme part which can be referenced by document content. This reference is used to map the use of the theme colors in the ST_ThemeColor enumeration to the theme colors in the theme part.

[*Example*: Consider a WordprocessingML document that shall have references to the theme color `accent1` mapped to the theme color lt1 as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings:

```
<w:clrSchemeMapping w:accent1="light1" />
```

The accent1 attribute has a value of `light1`, specifying that uses of the theme color value `accent1` shall be mapped to the theme color lt1. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
accent1 (Accent 1 Theme Color Reference)	Specifies a reference to the accent1 theme color in the

Enumeration Value	Description
	document's Theme part.
accent2 (Accent 2 Theme Color Reference)	Specifies a reference to the accent2 theme color in the document's Theme part.
accent3 (Accent 3 Theme Color Reference)	Specifies a reference to the accent3 theme color in the document's Theme part.
accent4 (Accent4 Theme Color Reference)	Specifies a reference to the accent4 theme color in the document's Theme part.
accent5 (Accent5 Theme Color Reference)	Specifies a reference to the accent5 theme color in the document's Theme part.
accent6 (Accent 6 Theme Color Reference)	Specifies a reference to the accent6 theme color in the document's Theme part.
dark1 (Dark 1 Theme Color Reference)	Specifies a reference to the dk1 theme color in the document's Theme part.
dark2 (Dark 2 Theme Color Reference)	Specifies a reference to the dk2 theme color in the document's Theme part.
followedHyperlink (Followed Hyperlink Theme Color Reference)	Specifies a reference to the folHlink theme color in the document's Theme part.
hyperlink (Hyperlink Theme Color Reference)	Specifies a reference to the hlink theme color in the document's Theme part.
light1 (Light 1 Theme Color Reference)	Specifies a reference to the lt1 theme color in the document's Theme part.
light2 (Light 2 Theme Color Reference)	Specifies a reference to the lt2 theme color in the document's Theme part.

Referenced By
clrSchemeMapping@accent1 (§2.15.1.20); clrSchemeMapping@accent2 (§2.15.1.20); clrSchemeMapping@accent3 (§2.15.1.20); clrSchemeMapping@accent4 (§2.15.1.20); clrSchemeMapping@accent5 (§2.15.1.20); clrSchemeMapping@accent6 (§2.15.1.20); clrSchemeMapping@bg1 (§2.15.1.20); clrSchemeMapping@bg2 (§2.15.1.20); clrSchemeMapping@followedHyperlink (§2.15.1.20); clrSchemeMapping@hyperlink (§2.15.1.20); clrSchemeMapping@t1 (§2.15.1.20); clrSchemeMapping@t2 (§2.15.1.20)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ColorSchemeIndex">
   <restriction base="xsd:string">
      <enumeration value="dark1"/>
      <enumeration value="light1"/>
      <enumeration value="dark2"/>
      <enumeration value="light2"/>
      <enumeration value="accent1"/>
      <enumeration value="accent2"/>
      <enumeration value="accent3"/>
      <enumeration value="accent4"/>
      <enumeration value="accent5"/>
      <enumeration value="accent6"/>
      <enumeration value="hyperlink"/>
      <enumeration value="followedHyperlink"/>
   </restriction>
</simpleType>
```

2.18.13 ST_CombineBrackets (Two Lines in One Enclosing Character Type)

This simple type specifies the type of bracket character which shall be used to enclose the two lines in one text within the current run when displayed

[*Example*: Consider a paragraph with the text `two lines in one`, which shall be displayed within a single logical line in the document and enclosed in curly brackets. This constraint would be specified as follows in the WordprocessingML:

```
<w:r>
  <w:rPr>
    <w:eastAsianLayout w:id="1" w:combine="on" w:combineBrackets="curly"/>
  </w:rPr>
  <w:t>two lines in one</w:t>
</w:r>
```

The resulting text would be displayed on two sub lines within the other text on this line and enclosed within `curly` brackets when displayed. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
angle (Angle Brackets)	Specifies that angle bracket characters shall be used to enclose the contents of the current run's two lines in one text. [*Example*: <…> *end example*]
curly (Curly Brackets)	Specifies that curly bracket characters shall be used to

Enumeration Value	Description
	enclose the contents of the current run's two lines in one text. [*Example*: {…} *end example*]
none (No Enclosing Brackets)	Specifies that no characters shall be used to enclose the contents of the current run's two lines in one text.
round (Round Brackets)	Specifies that round bracket characters shall be used to enclose the contents of the current run's two lines in one text. [*Example*: (…) *end example*]
square (Square Brackets)	Specifies that square bracket characters shall be used to enclose the contents of the current run's two lines in one text. [*Example*: […] *end example*]

Referenced By
eastAsianLayout@combineBrackets (§2.3.2.8)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_CombineBrackets">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="round"/>
      <enumeration value="square"/>
      <enumeration value="angle"/>
      <enumeration value="curly"/>
   </restriction>
</simpleType>
```

2.18.14 ST_CryptProv (Cryptographic Provider Types)

This simple type specifies the possible types of cryptographic providers which may be used.

[*Example*: Consider a WordprocessingML document with the following information stored in one of its protection elements:

```
<w:… w:cryptProviderType="rsaAES"
  w:hash="9oN7nWkCAyEZib1RomSJTjmPpCY=" />
```

The cryptProviderType attribute value of rsaAES specifies that the cryptographic provider type shall be an Advanced Encryption Standard provider. *end example*

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
rsaAES (AES Provider)	Specifies that the provider shall support the Advanced Encryption Algorithm standard.
rsaFull (Any Provider)	Specifies that any suitable provider shall be used.

Referenced By
documentProtection@cryptProviderType (§2.15.1.28); writeProtection@cryptProviderType (§2.15.1.94)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_CryptProv">
   <restriction base="xsd:string">
      <enumeration value="rsaAES"/>
      <enumeration value="rsaFull"/>
   </restriction>
</simpleType>
```

2.18.15 ST_DateTime (Standard Date and Time Storage Format)

This simple type specifies that its contents will contain a date in the standard XML Schema xsd:dateTime format, whose contents are interpreted based on the context of the parent XML element.

[*Example*: Consider the following WordprocessingML fragment:

```
<w:date w:realDate="01-01-2006T12:00:00Z">
  …
</w:date>
```

In this case, the date in the realDate attribute is the full date associated with the parent date picker structured document. In every case, the value of this type is interpreted in the context of the parent element or attribute. *end example*]

This simple type's contents are a restriction of the XML Schema dateTime datatype.

Referenced By
cellDel@date (§2.13.5.1); cellIns@date (§2.13.5.2); cellMerge@date (§2.13.5.3); comment@date (§2.13.4.2); customXmlDelRangeStart@date (§2.13.5.5); customXmlInsRangeStart@date (§2.13.5.7); customXmlMoveFromRangeStart@date (§2.13.5.9); customXmlMoveToRangeStart@date (§2.13.5.11); date@fullDate (§2.5.2.7); del@date (§2.13.5.12); del@date (§2.13.5.13); del@date (§2.13.5.14); del@date (§2.13.5.15); ins@date (§2.13.5.16); ins@date (§2.13.5.17); ins@date (§2.13.5.18); ins@date (§2.13.5.19); ins@date (§2.13.5.20); moveFrom@date (§2.13.5.21); moveFrom@date (§2.13.5.22); moveFromRangeStart@date (§2.13.5.24); moveTo@date (§2.13.5.25); moveTo@date (§2.13.5.26); moveToRangeStart@date (§2.13.5.28); numberingChange@date (§2.13.5.29); numberingChange@date (§2.13.5.30); pPrChange@date (§2.13.5.31); rPrChange@date (§2.13.5.32); rPrChange@date (§2.13.5.33);

Referenced By
sectPrChange@date (§2.13.5.34); tblPrChange@date (§2.13.5.36); tblPrExChange@date (§2.13.5.37); tcPrChange@date (§2.13.5.38); trPrChange@date (§2.13.5.39)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DateTime">
   <restriction base="xsd:dateTime"/>
</simpleType>
```

2.18.16 ST_DecimalNumber (Decimal Number Value)

This simple type specifies that its contents will contain a whole decimal number (positive or negative), whose contents are interpreted based on the context of the parent XML element.

[*Example*: Consider the following WordprocessingML fragment:

```
<w:pPr>
  <w:divId w:val="1512645511" />
</w:pPr>
```

The value of the val attribute is the ID of the associated HTML `div`.

However, consider the following fragment:

```
<w:ilvl w:val="1">
  ...
</w:ilvl>
```

In this case, the decimal number in the val attribute is the ID of the associated numbering level. In each case, the value is interpreted in the context of the parent element. *end example*]

This simple type's contents are a restriction of the XML Schema integer datatype.

Referenced By
abstractNum@abstractNumId (§2.9.1); abstractNumId@val (§2.9.2); activeRecord@val (§2.14.2); activeWritingStyle@dllVersion (§2.15.1.1); activeWritingStyle@vendorID (§2.15.1.1); bookFoldPrintingSheets@val (§2.15.1.12); bookmarkEnd@id (§2.13.6.1); bookmarkStart@id (§2.13.6.2); bottom@w (§2.4.2); bottom@w (§2.4.5); caption@heading (§2.15.1.16); cellDel@id (§2.13.5.1); cellIns@id (§2.13.5.2); cellMerge@id (§2.13.5.3); checkErrors@val (§2.14.4); colDelim@val (§2.14.5); cols@num (§2.6.4); column@val (§2.14.6); column@val (§2.14.7); comment@id (§2.13.4.2); commentRangeEnd@id (§2.13.4.3); commentRangeStart@id (§2.13.4.4); commentReference@id (§2.13.4.5); consecutiveHyphenLimit@val (§2.15.1.21); customXmlDelRangeEnd@id (§2.13.5.4); customXmlDelRangeStart@id (§2.13.5.5); customXmlInsRangeEnd@id (§2.13.5.6); customXmlInsRangeStart@id (§2.13.5.7); customXmlMoveFromRangeEnd@id (§2.13.5.8); customXmlMoveFromRangeStart@id (§2.13.5.9); customXmlMoveToRangeEnd@id (§2.13.5.10); customXmlMoveToRangeStart@id (§2.13.5.11); default@val (§2.16.11); del@id (§2.13.5.12); del@id (§2.13.5.13); del@id (§2.13.5.14); del@id (§2.13.5.15); displayHorizontalDrawingGridEvery@val

Referenced By
(§2.15.1.26); displayVerticalDrawingGridEvery@val (§2.15.1.27); div@id (§2.15.2.6); divId@val (§2.3.1.10); divId@val (§2.4.9); docGrid@charSpace (§2.6.5); docGrid@linePitch (§2.6.5); documentProtection@cryptAlgorithmSid (§2.15.1.28); documentProtection@cryptSpinCount (§2.15.1.28); eastAsianLayout@id (§2.3.2.8); endnote@id (§2.11.3); endnote@id (§2.11.2); endnoteReference@id (§2.11.7); fitText@id (§2.3.2.12); footnote@id (§2.11.9); footnote@id (§2.11.10); footnoteReference@id (§2.11.14); framePr@lines (§2.3.1.11); gridAfter@val (§2.4.10); gridBefore@val (§2.4.11); gridSpan@val (§2.4.13); id@val (§2.5.2.18); ilvl@val (§2.9.3); ind@firstLineChars (§2.3.1.12); ind@hangingChars (§2.3.1.12); ind@leftChars (§2.3.1.12); ind@rightChars (§2.3.1.12); ins@id (§2.13.5.16); ins@id (§2.13.5.17); ins@id (§2.13.5.18); ins@id (§2.13.5.19); ins@id (§2.13.5.20); latentStyles@count (§2.7.3.5); latentStyles@defUIPriority (§2.7.3.5); left@w (§2.4.25); left@w (§2.4.26); lnNumType@countBy (§2.6.8); lnNumType@start (§2.6.8); lsdException@uiPriority (§2.7.3.8); lvl@ilvl (§2.9.6); lvl@ilvl (§2.9.7); lvlOverride@ilvl (§2.9.9); lvlPicBulletId@val (§2.9.10); lvlRestart@val (§2.9.11); maxLength@val (§2.16.27); moveFrom@id (§2.13.5.21); moveFrom@id (§2.13.5.22); moveFromRangeEnd@id (§2.13.5.23); moveFromRangeStart@id (§2.13.5.24); moveTo@id (§2.13.5.25); moveTo@id (§2.13.5.26); moveToRangeEnd@id (§2.13.5.27); moveToRangeStart@id (§2.13.5.28); num@numId (§2.9.16); numberingChange@id (§2.13.5.29); numberingChange@id (§2.13.5.30); numId@val (§2.9.19); numIdMacAtCleanup@val (§2.9.20); numPicBullet@numPicBulletId (§2.9.21); numStart@val (§2.11.20); outlineLvl@val (§2.3.1.20); paperSrc@first (§2.6.9); paperSrc@other (§2.6.9); permStart@colFirst (§2.13.7.2); permStart@colLast (§2.13.7.2); pgNumType@chapStyle (§2.6.12); pgNumType@start (§2.6.12); pgSz@code (§2.6.13); pixelsPerInch@val (§2.15.2.33); pPrChange@id (§2.13.5.31); readModeInkLockDown@fontSz (§2.15.1.66); result@val (§2.16.29); right@w (§2.4.29); right@w (§2.4.31); rPrChange@id (§2.13.5.32); rPrChange@id (§2.13.5.33); sectPrChange@id (§2.13.5.34); spacing@afterLines (§2.3.1.33); spacing@beforeLines (§2.3.1.33); start@val (§2.9.27); startOverride@val (§2.9.28); summaryLength@val (§2.15.1.88); tblCellSpacing@w (§2.4.41); tblCellSpacing@w (§2.4.42); tblCellSpacing@w (§2.4.43); tblGridChange@id (§2.13.5.35); tblInd@w (§2.4.47); tblInd@w (§2.4.48); tblPrChange@id (§2.13.5.36); tblPrExChange@id (§2.13.5.37); tblStyleColBandSize@val (§2.7.5.5); tblStyleRowBandSize@val (§2.7.5.7); tblW@w (§2.4.60); tblW@w (§2.4.61); tcPrChange@id (§2.13.5.38); tcW@w (§2.4.68); top@w (§2.4.72); top@w (§2.4.73); trPrChange@id (§2.13.5.39); uiPriority@val (§2.7.3.19); wAfter@w (§2.4.82); wBefore@w (§2.4.83); writeProtection@cryptAlgorithmSid (§2.15.1.94); writeProtection@cryptSpinCount (§2.15.1.94); zoom@percent (§2.15.1.95)

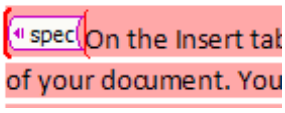
The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DecimalNumber">
   <restriction base="xsd:integer"/>
</simpleType>
```

2.18.17 ST_DisplacedByCustomXml (Location of Custom XML Markup Displacing an Annotation)

This simple type specifies the possible values for the location of a single custom XML element's start and/or end tag relative to the location of an annotation tag in document order. This enumeration shall be used to specify that the parent annotation's placement shall be directly linked with the location of the physical presentation of a custom XML element in the document.

[*Example*: Consider a paragraph with block level custom XML markup and two comment anchor annotations (one before and one after the custom XML element's physical representation), as follows:



Since all three of these items are around the entire paragraph, they are stored outside of the paragraph. However, in order to ensure that their relative positions are stored correctly, any annotation which shall be displaced by the physical custom XML element specifies this information, resulting in the following WordprocessingML:

```
<w:commentRangeStart w:id="0" />
<w:commentRangeStart w:id="1" w:displaced byCustomXml="next" />
<w:customXml w:element="spec" … />
<w:p>
  …
</w:p>
```

The displacedByCustomXml attribute specifies that even though all three of these items are around the paragraph and will be moved inside the paragraph to be represented physically, the comment with ID 0 shall be inside the custom XML, but the comment with ID 1 shall be displaced to stay outside of the relative location of the next custom XML element (the spec element). *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
next (Displaced by Next Custom XML Markup Tag)	Specifies that this annotation anchor shall be displaced by the physical representation of the next element of custom XML markup in the document. If no custom XML markup exists in the same paragraph and after this anchor, then this setting shall be ignored.
prev (Displaced by Previous Custom XML Markup Tag)	Specifies that this annotation anchor shall be displaced by the physical representation of the previous element of custom XML markup in the document. If no custom XML markup exists in the same paragraph and directly before this anchor, then this setting shall be ignored.

Referenced By
bookmarkEnd@displacedByCustomXml (§2.13.6.1); bookmarkStart@displacedByCustomXml (§2.13.6.2); commentRangeEnd@displacedByCustomXml (§2.13.4.3); commentRangeStart@displacedByCustomXml

Referenced By
(§2.13.4.4); moveFromRangeEnd@displacedByCustomXml (§2.13.5.23); moveFromRangeStart@displacedByCustomXml (§2.13.5.24); moveToRangeEnd@displacedByCustomXml (§2.13.5.27); moveToRangeStart@displacedByCustomXml (§2.13.5.28); permEnd@displacedByCustomXml (§2.13.7.1); permStart@displacedByCustomXml (§2.13.7.2)

The following XML Schema fragment defines the contents of this simple type:

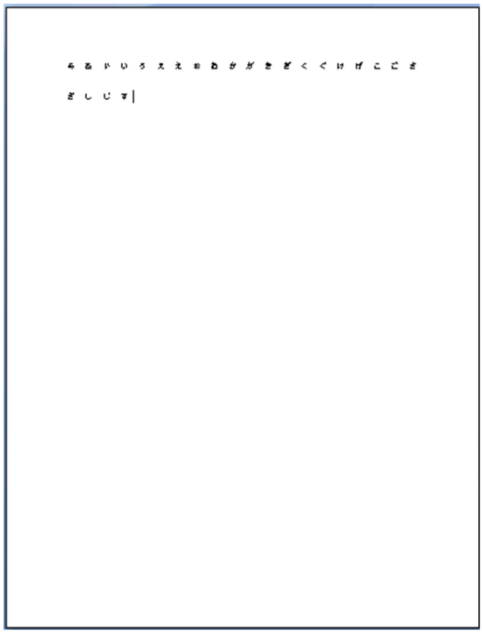
```
<simpleType name="ST_DisplacedByCustomXml">
    <restriction base="xsd:string">
        <enumeration value="next"/>
        <enumeration value="prev"/>
    </restriction>
</simpleType>
```

2.18.18 ST_DocGrid (Document Grid Types)

Specifies the type of the current document grid, which defines the grid behavior.

The grid can define a grid which snaps all East Asian characters to grid positions, but leaves Latin text with its default spacing; a grid which adds the specified character pitch to all characters on each row; or a grid which affects only the line pitch for the current section.

[*Example*: Consider the document discussed above with the document grid defined to allow 20 characters per line, and 20 lines per page by snapping characters to the grid as follows:



This document has a type attribute of type ST_DocGrid and value `snapToChars`, which specifies that the grid shall force East Asian characters to fit 20 to a line. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
default (No Document Grid)	Specifies that no document grid shall be applied to the contents of the current section in the document.
lines (Line Grid Only)	Specifies that the parent section shall have additional line pitch added to each line within it (as specified on the docGrid element (§2.6.5)) in order to maintain the specified number of lines per page.
linesAndChars (Line and Character Grid)	Specifies that the parent section shall have both the additional line pitch and character pitch added to each line and character within it (as specified on the docGrid element (§2.6.5)) in order to maintain a specific number of lines per page and characters per line. When this value is set, the input specified via the user interface may be allowed in exact number of line/character pitch units.
snapToChars (Character Grid Only)	Specifies that the parent section shall have both the additional line pitch and character pitch added to each line and character within it (as specified on the docGrid element (§2.6.5)) in order to maintain a specific number of lines per page and characters per line. When this value is set, the input specified via the user interface may be restricted to the number of lines per page and characters per line, with the consumer or producer translating this information based on the current font data to get the resulting line and character pitch values

Referenced By
docGrid@type (§2.6.5)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DocGrid">
   <restriction base="xsd:string">
      <enumeration value="default"/>
      <enumeration value="lines"/>
      <enumeration value="linesAndChars"/>
      <enumeration value="snapToChars"/>
   </restriction>
</simpleType>
```

2.18.19 ST_DocPartBehavior (Insertion Behavior Types)

This simple type specifies the possible sets of behaviors which may be applied to the contents of a single glossary document entry (§2.12.5) when it is added to the main document story of a WordprocessingML document.

[*Example*: Consider the WordprocessingML fragment for a glossary document entry containing a single run, defined as follows:

```
<w:docPart>
  <w:docPartPr>
    <w:behaviors>
      <w:behavior w:val="p"/>
    </w:behavior>
    …
  </w:docPartPr>
  <w:docPartBody>
    <w:p>
      <w:r>
        <w:t>Sample entry.</w:t>
      </w:r>
    </w:p>
  </w:docPartBody>
</w:docPart>
```

The behavior element of type ST_DocPartBehavior has a value of p, which specifies that the contents of the parent glossary document entry shall be inserted in their own paragraph when they are added to the contents of a document. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
content (Insert Content At Specified Location)	Specifies that when the glossary document entry is inserted into the main document contents of the document, it shall be inserted normally as defined above. This includes ensuring that the final paragraph which is included in the part is not inserted, and its run content is added to the paragraph into which the current part is being inserted.
p (Ensure Entry Is In New Paragraph)	Specifies that the glossary document entry shall be added into its own unique paragraph, by failing to remove the last paragraph from the entry's contents

Enumeration Value	Description
	when they are added to the document.
pg (Ensure Entry Is On New Page)	Specifies that the glossary document entry shall be added into its own new page, by preceding the entry with a blank paragraph whose only content is a page break character.

Referenced By
behavior@val (§2.12.1)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DocPartBehavior">
   <restriction base="xsd:string">
      <enumeration value="content"/>
      <enumeration value="p"/>
      <enumeration value="pg"/>
   </restriction>
</simpleType>
```

2.18.20 ST_DocPartGallery (Entry Gallery Types)

This simple type specifies possible settings for the predefined gallery into which a glossary document part shall be classified. This classification, although its enumeration values may be interpreted to imply semantics around the contents of the parent glossary document entry, shall only be used to classify and sort this entry (via an application or a user interface).

[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:

```
<w:docPartPr>
  <w:category>
    <w:gallery w:val="coverPg" />
    <w:name w:val="Internal Memo Covers" />
  </w:category>
  …
<w:docPartPr>
```

The gallery element with a value of coverPg specifies that the gallery categorization applied to the current entry, for the purposes of classification or user interface sorting, puts this entry into the Cover Pages classification. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
any (All Galleries)	Specifies that this glossary document entry shall be associated with all possible gallery classification values.
autoTxt (AutoText Gallery)	Specifies that this glossary document entry shall be associated with the AutoText gallery classification.
bib (Bibliography Gallery)	Specifies that this glossary document entry shall be associated with the Bibliography gallery classification.
coverPg (Cover Page Gallery)	Specifies that this glossary document entry shall be associated with the Cover Page gallery classification.
custAutoTxt (Custom AutoText Gallery)	Specifies that this glossary document entry shall be associated with the Custom AutoText gallery classification.
custBib (Custom Bibliography Gallery)	Specifies that this glossary document entry shall be associated with the Custom Bibliography gallery classification.
custCoverPg (Custom Cover Page Gallery)	Specifies that this glossary document entry shall be associated with the Custom Cover Page gallery classification.
custEq (Custom Equation Gallery)	Specifies that this glossary document entry shall be associated with the Custom Equation gallery classification.
custFtrs (Custom Footer Gallery)	Specifies that this glossary document entry shall be associated with the Custom Footer gallery classification.
custHdrs (Custom Header Gallery)	Specifies that this glossary document entry shall be associated with the Custom Header gallery classification.
custom1 (Custom 1 Gallery)	Specifies that this glossary document entry shall be associated with the Custom 1 gallery classification.
custom2 (Custom 2 Gallery)	Specifies that this glossary document entry shall be associated with the Custom 2 gallery classification.
custom3 (Custom 3 Gallery)	Specifies that this glossary document entry shall be associated with the Custom 3 gallery classification.
custom4 (Custom 4 Gallery)	Specifies that this glossary document entry shall be associated with the Custom 4 gallery classification.
custom5 (Custom 5 Gallery)	Specifies that this glossary document entry shall be associated with the Custom 5 gallery classification.
custPgNum (Custom Page Number Gallery)	Specifies that this glossary document entry shall be associated with the Custom Page Number gallery classification.

Enumeration Value	Description
custPgNumB (Custom Page Number At Bottom Gallery)	Specifies that this glossary document entry shall be associated with the Custom Page Number At Bottom gallery classification.
custPgNumMargins (Custom Page Number At Margins Gallery)	Specifies that this glossary document entry shall be associated with the Custom Page Number At Margins gallery classification.
custPgNumT (Custom Page Number At Top Gallery)	Specifies that this glossary document entry shall be associated with the Custom Page Number At Top gallery classification.
custQuickParts (Custom Quick Parts Gallery)	Specifies that this glossary document entry shall be associated with the Custom Quick Parts gallery classification.
custTblOfContents (Custom Table of Contents Gallery)	Specifies that this glossary document entry shall be associated with the Custom Table of Contents gallery classification.
custTbls (Custom Table Gallery)	Specifies that this glossary document entry shall be associated with the Custom Tables gallery classification.
custTxtBox (Custom Text Box Gallery)	Specifies that this glossary document entry shall be associated with the Custom Text Box gallery classification.
custWatermarks (Custom Watermark Gallery)	Specifies that this glossary document entry shall be associated with the Custom Watermark gallery classification.
default (No Gallery Classification)	Specifies that this glossary document entry shall not have a gallery classification.
docParts (Document Parts Gallery)	Specifies that this glossary document entry shall be associated with the Document Parts gallery classification.
eq (Equations Gallery)	Specifies that this glossary document entry shall be associated with the Equations gallery classification.
ftrs (Footers Gallery)	Specifies that this glossary document entry shall be associated with the Footers gallery classification.
hdrs (Headers Gallery)	Specifies that this glossary document entry shall be associated with the Headers gallery classification.
pgNum (Page Numbers Gallery)	Specifies that this glossary document entry shall be associated with the Page Numbers gallery classification.
pgNumB (Page Numbers At Bottom Gallery)	Specifies that this glossary document entry shall be associated with the Page Numbers At Bottom gallery classification.

Enumeration Value	Description
pgNumMargins (Page Numbers At Margins Gallery)	Specifies that this glossary document entry shall be associated with the Page Numbers At Margins gallery classification.
pgNumT (Page Numbers At Top Gallery)	Specifies that this glossary document entry shall be associated with the Page Numbers At Top gallery classification.
placeholder (Structured Document Tag Placeholder Text Gallery)	Specifies that this glossary document entry shall be associated with the Structured Document Tag Placeholder Text gallery classification.
tblOfContents (Table of Contents Gallery)	Specifies that this glossary document entry shall be associated with the Table of Contents gallery classification.
tbls (Table Gallery)	Specifies that this glossary document entry shall be associated with the Tables gallery classification.
txtBox (Text Box Gallery)	Specifies that this glossary document entry shall be associated with the Text Box gallery classification.
watermarks (Watermark Gallery)	Specifies that this glossary document entry shall be associated with the Watermark gallery classification.

Referenced By
gallery@val (§2.12.9)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DocPartGallery">
   <restriction base="xsd:string">
      <enumeration value="placeholder"/>
      <enumeration value="any"/>
      <enumeration value="default"/>
      <enumeration value="docParts"/>
      <enumeration value="coverPg"/>
      <enumeration value="eq"/>
      <enumeration value="ftrs"/>
      <enumeration value="hdrs"/>
      <enumeration value="pgNum"/>
      <enumeration value="tbls"/>
      <enumeration value="watermarks"/>
      <enumeration value="autoTxt"/>
      <enumeration value="txtBox"/>
      <enumeration value="pgNumT"/>
      <enumeration value="pgNumB"/>
      <enumeration value="pgNumMargins"/>
      <enumeration value="tblOfContents"/>
      <enumeration value="bib"/>
      <enumeration value="custQuickParts"/>
      <enumeration value="custCoverPg"/>
      <enumeration value="custEq"/>
      <enumeration value="custFtrs"/>
      <enumeration value="custHdrs"/>
      <enumeration value="custPgNum"/>
      <enumeration value="custTbls"/>
      <enumeration value="custWatermarks"/>
      <enumeration value="custAutoTxt"/>
      <enumeration value="custTxtBox"/>
      <enumeration value="custPgNumT"/>
      <enumeration value="custPgNumB"/>
      <enumeration value="custPgNumMargins"/>
      <enumeration value="custTblOfContents"/>
      <enumeration value="custBib"/>
      <enumeration value="custom1"/>
      <enumeration value="custom2"/>
      <enumeration value="custom3"/>
      <enumeration value="custom4"/>
      <enumeration value="custom5"/>
   </restriction>
</simpleType>
```

2.18.21 ST_DocPartType (Entry Types)

This simple type specifies the possible types which may be applied to the properties of a single glossary document entry (§2.12.5). Each of these types may, based on their values, influence the visibility and behavior of the parent glossary document entry.

[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:

```
<w:docPartPr>
  <w:types>
    <w:type w:val="bbPlcHdr" />
  </w:types>
  …
<w:docPartPr>
```

The type element with a value of bbPlcHdr specifies that the parent glossary document entry shall be treated as if it was the placeholder text for one or more structured document tags in the document. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
autoExp (Automatically Replace Name With Content)	Specifies that the type of the current glossary document entry shall allow the entry to be automatically inserted into the document whenever its name is entered into an application.
bbPlcHdr (Structured Document Tag Placeholder Text)	Specifies that the type of the current glossary document entry shall be structured document tag placeholder text.
formFld (Form Field Help Text)	Specifies that the type of the current glossary document entry shall be form field help text.
none (No Type)	Specifies no type information for the current glossary document entry.
normal (Normal)	Specifies that the type of the current glossary document entry shall be normal (i.e. a regular glossary document entry).
speller (AutoCorrect Entry)	Specifies that the type of the current glossary document entry shall be associated with the spelling and grammar tools.
toolbar (AutoText User Interface Entry)	Specifies that the type of the current glossary document entry shall be associated with a special grouping of entries associated with a single piece of user interface.

Referenced By
type@val (§2.12.15)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DocPartType">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="normal"/>
      <enumeration value="autoExp"/>
      <enumeration value="toolbar"/>
      <enumeration value="speller"/>
      <enumeration value="formFld"/>
      <enumeration value="bbPlcHdr"/>
   </restriction>
</simpleType>
```

2.18.22 ST_DocProtect (Document Protection Types)

This simple type specifies the possible set of editing restrictions which may be enforced on a given WordprocessingML document.

[*Example*: Consider a WordprocessingML document that contains the following WordprocessingML specifying that hosting applications shall enforce read-only protection for a given document:

```
<w:documentProtection w:edit="readOnly" w:enforcement="1" />
```

The edit attribute has a value of `readOnly` and an enforcement attribute with a value of `1`, specifying that read-only document protection shall be enforced on the given document. e*nd example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
comments (Allow Editing of Comments)	Specifies that the edits made to this document shall be restricted to: • The insertion and deletion of comments within the document • The editing of regions delimited by range permissions which match the editing rights of the user account which is performing the editing.
forms (Allow Editing of Form Fields)	Specifies that the edits made to this document shall be restricted to: • The editing of form fields in sections where the formProt element (§2.6.6) has a value of `true`. • No restrictions in sections where the formProt element has a value of `false`.
none (No Editing Restrictions)	Specifies that no editing restrictions have been applied to the document.

Enumeration Value	Description
readOnly (Allow No Editing)	Specifies that the edits made to this document shall be restricted to: • The editing of regions delimited by range permissions which match the editing rights of the user account which is performing the editing.
trackedChanges (Allow Editing With Revision Tracking)	Specifies that the edits made to this document shall be tracked as revisions. This value shall imply the presence of the trackRevisions element (§2.15.1.90), and applications shall not allow that element's state to be changed to `false`.

Referenced By
documentProtection@edit (§2.15.1.28)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DocProtect">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="readOnly"/>
      <enumeration value="comments"/>
      <enumeration value="trackedChanges"/>
      <enumeration value="forms"/>
   </restriction>
</simpleType>
```

2.18.23 ST_DocType (Document Classification Values)

This simple type specifies the possible classifications may be used for a WordprocessingML document.

[*Example*: Consider a set of WordprocessingML documents which should be classified as 'letters'. This classification would be specified using the following WordprocessingML in the document settings of these documents:

```
<w:documentType w:val="letter" />
```

The documentType element's val attribute is equal to `letter`, specifying that the hosting application shall apply the behaviors it has specified for letters to the given WordprocessingML document. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
eMail (E-Mail Message)	Specifies that this document shall be classified as an e-

Enumeration Value	Description
	mail message.
letter (Letter)	Specifies that this document shall be classified as a letter.
notSpecified (Default Document)	Specifies that this document shall be classified as a default document.

Referenced By
documentType@val (§2.15.1.29)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DocType">
   <restriction base="xsd:string">
      <enumeration value="notSpecified"/>
      <enumeration value="letter"/>
      <enumeration value="eMail"/>
   </restriction>
</simpleType>
```

2.18.24 ST_DropCap (Text Frame Drop Cap Location)

This simple type specifies the location which shall be used to position a drop cap text frame when the contents of that text frame are displayed in the anchor paragraph at display time.

[*Note*: Although a drop cap is simply a text frame, the values of this simple type are used to determine how the cap should be positioned relative to the following non-frame paragraph in relative terms (see enumeration values), rather than relying on absolute sizing. *end note*]

[*Example*: Consider the following paragraph containing a text frame which should be positioned as a drop cap:

```
<w:p>
  <w:pPr>
    <w:framePr w:dropCap="margin" w:lines="3" w:hSpace="432" w:wrap="around"
w:vAnchor="text" w:hAnchor="page" />
  </w:pPr>
  <w:r>
    <w:t>A</w:t>
  </w:r>
</w:p>
```

The dropCap attribute specifies a value of `margin`, so this drop cap will be placed outside of the text margin before the start of the current text. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
drop (Drop Cap Inside Margin)	Specifies that the drop cap text frame shall be positioned inside the text margin on the anchor paragraph when this text frame is displayed in the document.
margin (Drop Cap Outside Margin)	Specifies that the drop cap text frame shall be positioned outside of the text margin on the anchor paragraph when this text frame is displayed in the document.
none (Not Drop Cap)	Specifies that this text frame is not a drop cap text frame.

Referenced By
framePr@dropCap (§2.3.1.11)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DropCap">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="drop"/>
      <enumeration value="margin"/>
   </restriction>
</simpleType>
```

2.18.25 ST_EdGrp (Range Permision Editing Group)

This simple type specifies the set of possible aliases (or editing groups) which may be used as aliases to determine if the current user shall be allowed to edit a single range defined by a range permission within a document. This mechanism simply provides a set of predefined editing groups which may be associated with user accounts by applications in any desired manner.

[*Example*: Consider a range permission defined as follows:

```
<w:permStart w:id="0" w:edGrp="editors" … />
…
<w:permEnd w:id="0" />
```

The edGrp attribute value of `editors` specifies that only user(s) who the current application associates with the editors group shall be allowed to edit the contents between the start and end markers when document protection is being enforced. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
administrators (Administrator Group)	Specifies that users associated with the Administrators group shall be allowed to edit range permissions using this editing group when document protection is enabled.
contributors (Contributors Group)	Specifies that users associated with the Contributors group shall be allowed to edit range permissions using this editing group when document protection is enabled.
current (Current Group)	Specifies that users associated with the Current group shall be allowed to edit range permissions using this editing group when document protection is enabled.
editors (Editors Group)	Specifies that users associated with the Editors group shall be allowed to edit range permissions using this editing group when document protection is enabled.
everyone (All Users Have Editing Permissions)	Specifies that all users that open the document shall be allowed to edit range permissions using this editing group when document protection is enabled.
none (No Users Have Editing Permissions)	Specifies that none of the users that open the document shall be allowed to edit range permissions using this editing group when document protection is enabled.
owners (Owners Group)	Specifies that users associated with the Owners group shall be allowed to edit range permissions using this editing group when document protection is enabled.

Referenced By
permStart@edGrp (§2.13.7.2)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_EdGrp">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="everyone"/>
      <enumeration value="administrators"/>
      <enumeration value="contributors"/>
      <enumeration value="editors"/>
      <enumeration value="owners"/>
      <enumeration value="current"/>
   </restriction>
</simpleType>
```

2.18.26 ST_EdnPos (Endnote Positioning Location)

This simple type specifies the possible positions of endnotes in a document.

[*Example*: Consider a document in which endnotes shall be positioned at the end of the section. The section properties for this section shall be declared as follows:

```
<w:settings>
  <w:endnotePr>
    <w:pos w:val="endSect" />
  </w:endnotePr>
  …
</w:settings>
```

The val attribute is endSect, therefore the position of endnotes is specified to be at the end of the section. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
docEnd (Endnotes Positioned at End of Document)	Specifies that all endnotes shall be placed at the end of the current document, regardless of which section they are referenced within.
sectEnd (Endnotes Positioned at End of Section)	Specifies that endnotes shall be placed at the end of the section in which they are referenced. An endnote which is never referenced is never displayed.

Referenced By
pos@val (§2.11.22)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_EdnPos">
   <restriction base="xsd:string">
      <enumeration value="sectEnd"/>
      <enumeration value="docEnd"/>
   </restriction>
</simpleType>
```

2.18.27 ST_EighthPointMeasure (Measurement in Eighths of a Point)

This simple type specifies that its contents will contain a positive whole number, whose contents consist of a measurement in eighths of a point (equivalent to 1/576th of an inch).

The contents of this measurement are interpreted based on the context of the parent XML element.

[*Example*: Consider an attribute value of 24 whose type is ST_EighthPointMeasure. This attribute value specifies a size in eighths of a point (24 eighths of a point = 3 points). *end example*]

This simple type's contents are a restriction of the ST_UnsignedDecimalNumber simple type (§2.18.108).

Referenced By
bar@sz (§2.3.1.4); bdr@sz (§2.3.2.3); between@sz (§2.3.1.5); bottom@sz (§2.6.2); bottom@sz (§2.4.3); bottom@sz (§2.4.4); bottom@sz (§2.15.2.4); bottom@sz (§2.3.1.7); insideH@sz (§2.4.17); insideH@sz (§2.4.18); insideV@sz (§2.4.19); insideV@sz (§2.4.20); left@sz (§2.15.2.21); left@sz (§2.4.24); left@sz (§2.6.7); left@sz (§2.3.1.17); left@sz (§2.4.27); right@sz (§2.3.1.28); right@sz (§2.4.30); right@sz (§2.6.15); right@sz (§2.15.2.35); right@sz (§2.4.32); tl2br@sz (§2.4.70); top@sz (§2.4.71); top@sz (§2.3.1.42); top@sz (§2.15.2.42); top@sz (§2.4.74); top@sz (§2.6.21); tr2bl@sz (§2.4.76)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_EighthPointMeasure">
   <restriction base="ST_UnsignedDecimalNumber"/>
</simpleType>
```

2.18.28 ST_Em (Emphasis Mark Type)

This simple type specifies possible types of emphasis marks which may be displayed for each non-space character in a run. This character is rendered above or below the character glyph as specified by enumeration values.

[*Example*: Consider a run of text which shall have a dot underneath each character as an emphasis mark. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:em w:val="dot"/>
</w:rPr>
```

This run explicitly declares that the emphasis mark type is dot, so the contents of this run will have a dot emphasis mark above each character. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
circle (Circle Emphasis Mark Above Characters)	Specifies that the emphasis mark is a circle character which shall be rendered above each character in this run using Unicode character 0x02DA when the language of the text is not Traditional Chinese. For that language Unicode character 0x3002 shall be

Enumeration Value	Description
	used instead, positioned beneath the characters.
comma (Comma Emphasis Mark Above Characters)	Specifies that the emphasis mark is a comma character which shall be rendered above each character in this run, using Unicode character 0x3001.
dot (Dot Emphasis Mark Above Characters)	Specifies that the emphasis mark is a dot character which shall be rendered above each character in this run using Unicode character 0x02D9 whenever the language of the text is not Japanese, Simplified Chinese, or Traditional Chinese. For those three languages, the emphasis mark shall be rendered as follows: • Japanese = Unicode character 0xFF0E (dot beneath characters) • Simplified Chinese = Unicode character 0xFF0E (dot beneath characters) • Traditional Chinese = Unicode character 0x2027
none (No Emphasis Mark)	Specifies that there shall be no emphasis mark for any character in this run.
underDot (Dot Emphasis Mark Below Characters)	Specifies that the emphasis mark is a dot character which shall be rendered below each character in this run using Unicode character 0xFF0E.

Referenced By
em@val (§2.3.2.10)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Em">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="dot"/>
      <enumeration value="comma"/>
      <enumeration value="circle"/>
      <enumeration value="underDot"/>
   </restriction>
</simpleType>
```

2.18.29 ST_FFHelpTextVal (Help Text Value)

This simple type specifies the format of optional help text which may be associated with the parent form field.

[*Example*: Consider the following WordprocessingML fragment for a drop-down list form field:

```
<w:fldSimple w:instr="FORMDROPDOWN">
  <w:ffData>
    <w:helpText w:type="text" w:val="Example help text." />
    <w:ddList>
      …
    </w:ddList>
  </w:ffData>
</w:fldSimple>
```

The helpText element specifies the help text for the parent form field - in this case, literal help text consisting of the string `Example help text.` *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

This simple type also specifies the following restrictions:

- This simple type's contents have a maximum length of 256 characters.

Referenced By
helpText@val (§2.16.23)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FFHelpTextVal">
   <restriction base="xsd:string">
      <maxLength value="256"/>
   </restriction>
</simpleType>
```

2.18.30 ST_FFName (Form Field Name Value)

This simple type specifies the format of the name which may be associated with the parent form field.

[*Example*: Consider the following WordprocessingML fragment for a text box form field:

```
<w:fldSimple w:instr="FORMTEXT">
  <w:ffData>
    <w:name w:val="FirstName"/>
    <w:textInput>
      …
    </w:textInput>
  </w:ffData>
</w:fldSimple>
```

The name element specifies that the name of the current form field is `FirstName`. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

This simple type also specifies the following restrictions:

- This simple type's contents have a maximum length of 65 characters.

Referenced By
name@val (§2.16.28)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FFName">
   <restriction base="xsd:string">
      <maxLength value="65"/>
   </restriction>
</simpleType>
```

2.18.31 ST_FFStatusTextVal (Status Text Value)

This simple type specifies the format of optional status text which may be associated with the parent form field.

[*Example*: Consider the following WordprocessingML fragment for a drop-down list form field:

```
<w:fldSimple w:instr="FORMDROPDOWN">
  <w:ffData>
    <w:statusText w:type="text" w:val="Example status text." />
    <w:ddList>
      …
    </w:ddList>
  </w:ffData>
</w:fldSimple>
```

The statusText element specifies the status text for the parent form field - in this case, literal text consisting of the string `Example status text.` *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

This simple type also specifies the following restrictions:

- This simple type's contents have a maximum length of 140 characters.

Referenced By
statusText@val (§2.16.32)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FFStatusTextVal">
   <restriction base="xsd:string">
      <maxLength value="140"/>
   </restriction>
</simpleType>
```

2.18.32 ST_FFTextType (Text Box Form Field Type Values)

This simple type specifies the possible types of the contents of a text box form field.

[*Example*: Consider the following WordprocessingML fragment for the properties of a text box form field:

```
<w:ffData>
  <w:textInput>
    <w:type w:val="number" />
    <w:maxLength w:val="4" />
    <w:format w:val="0.00" />
  </w:textInput>
</w:ffData>
```

The type element specifies that the contents of this form field should be handled as a `number` by an application. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
calculated (Field Calculation)	Specifies that the contents of this text box form field shall be the result of the field calculation specified by the corresponding default element (§2.16.10). This field should not be directly editable when the editing of form fields is enabled.
currentDate (Current Date Display)	Specifies that the contents of this text box form field shall be the current date when the field is updated.
currentTime (Current Time Display)	Specifies that the contents of this text box form field shall be the current time when the field is updated.
date (Date)	Specifies that the contents of this text box form field shall be treated as a date.
number (Number)	Specifies that the contents of this text box form field shall be treated as a number value.
regular (Text Box)	Specifies that this text form field is a plain text field (no additional content restrictions).

Referenced By
type@val (§2.16.34)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FFTextType">
   <restriction base="xsd:string">
      <enumeration value="regular"/>
      <enumeration value="number"/>
      <enumeration value="date"/>
      <enumeration value="currentTime"/>
      <enumeration value="currentDate"/>
      <enumeration value="calculated"/>
   </restriction>
</simpleType>
```

2.18.33 ST_FldCharType (Complex Field Character Type)

This simple type specifies the possible values for the type of a single complex field character in the document.

[*Example*: Consider the following WordprocessingML for a complex field character:

```
…
<w:fldChar w:type="separate" />
…
```

The type attribute value of `separate` specifies that this is a complex field separator character; therefore it is being used to separate the field codes from the field contents in a complex field. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
begin (Start Character)	Specifies that the character is a start character, which defines the start of a complex field.
end (End Character)	Specifies that the character is an end character, which defines the end of a complex field.
separate (Separator Character)	Specifies that the character is a separator character, which defines the end of the field codes and the start of the field result for a complex field.

Referenced By
fldChar@fldCharType (§2.16.18)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FldCharType">
   <restriction base="xsd:string">
      <enumeration value="begin"/>
      <enumeration value="separate"/>
      <enumeration value="end"/>
   </restriction>
</simpleType>
```

2.18.34 ST_FontFamily (Font Family Value)

This simple type specifies possible values for the font family of a font.

[*Example*: Consider the following information stored for a single font:

```
<w:font w:name="Calibri">
  <w:family w:val="swiss" />
  …
</w:font>
```

The family element specifies via its val attribute value of `swiss` that this font is part of the Swiss family. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
auto (No Font Family)	Specifies that information about a font's font family does not exist.
decorative (Novelty Font)	Specifies the Novelty font family.
modern (Monospace Font)	Specifies a monospace font with or without serifs (monospace fonts are usually modern).
roman (Proportional Font With Serifs)	Specifies a proportional font with serifs.
script (Script Font)	Specifies a script font designed to mimic the appearance of handwriting.
swiss (Proportional Font Without Serifs)	Specifies a proportional font without serifs.

Referenced By
family@val (§2.8.2.9)

The following XML Schema fragment defines the contents of this simple type:

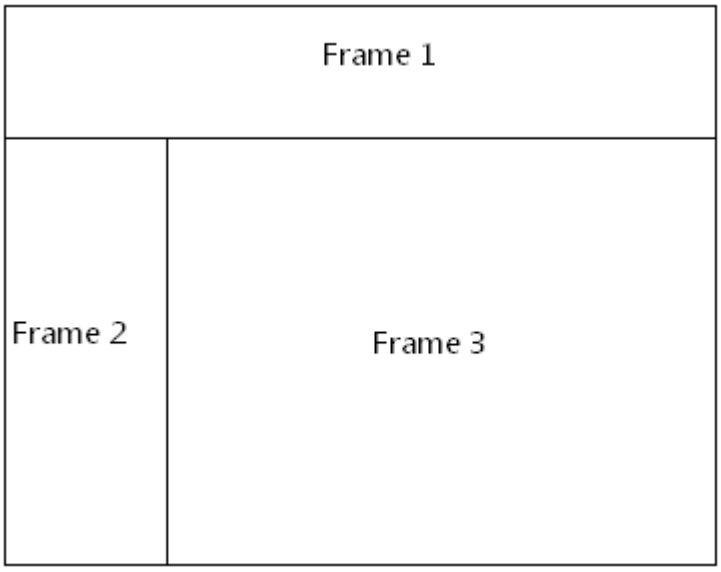
```
<simpleType name="ST_FontFamily">
   <restriction base="xsd:string">
      <enumeration value="decorative"/>
      <enumeration value="modern"/>
      <enumeration value="roman"/>
      <enumeration value="script"/>
      <enumeration value="swiss"/>
      <enumeration value="auto"/>
   </restriction>
</simpleType>
```

2.18.35 ST_FrameLayout (Frameset Layout Order)

This simple type specifies the possible order in which the frames (and nested framesets) in a frameset may be displayed. When a frameset is created, it can only contain frames which are stacked in one direction:

- Vertically (one on top of another)
- Horizontally (one next to another)

[*Example*: Consider a WordprocessingML document which serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  <w:frameLayout w:val="rows" />
  <w:frame>
    …
  </w:frame>
```

```
    <w:frameset>
      <w:frameLayout w:val="cols" />
      <w:frame>
        …
      </w:frame>
      <w:frame>
        …
      </w:frame>
    </w:frameset>
  </w:frameset>
```

The frameLayout element specifies that the outer frameset is consists of the single frame and the child frameset stacked vertically, and an inner nested frameset consisting of two frames stacked horizontally. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
cols (Stack Frames Horizontally)	Specifies that the frames in the frameset shall be stacked horizontally next to each other in left to right order.
none (Do Not Stack Frames)	Specifies that no frames shall be shown in the frameset.
rows (Stack Frames Vertically)	Specifies that the frames in the frameset shall be stacked vertically next to each other in top to bottom order.

Referenced By
frameLayout@val (§2.15.2.17)

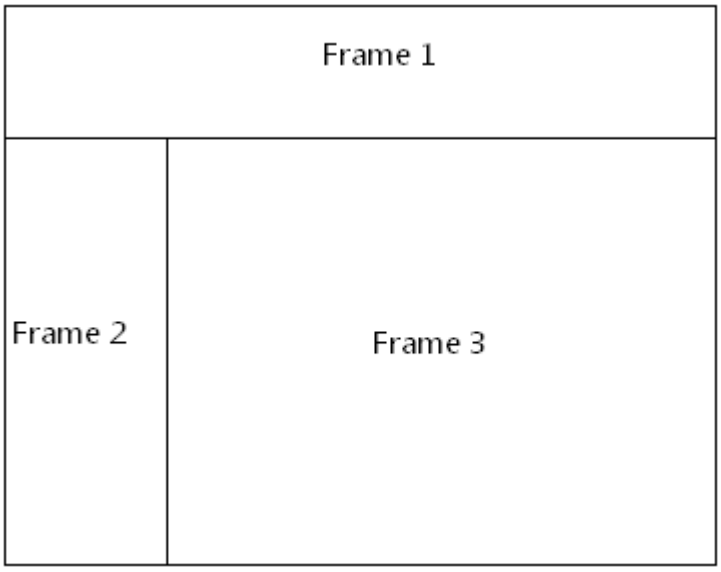
The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FrameLayout">
   <restriction base="xsd:string">
      <enumeration value="rows"/>
      <enumeration value="cols"/>
      <enumeration value="none"/>
   </restriction>
</simpleType>
```

2.18.36 ST_FrameScrollbar (Frame Scrollbar Visibility)

This simple type specifies the possible settings for when a scrollbar shall be visible for the contents of the current frame.

[*Example*: Consider a WordprocessingML document which serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  …
  <w:frameset>
    …
    <w:frame>
      <w:name w:val="Frame 2" />
      <w:scrollbar w:val="auto" />
    </w:frame>
    …
  </w:frameset>
</w:frameset>
```

The scrollbar element has a val attribute of auto, which specifies that the frame shall only display a scrollbar when it is needed to display all of its content. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
auto (Automatically Show Scrollbar As Needed)	Specifies that the scrollbar for a frame shall automatically be hidden and/or displayed as needed based on the length of the contents.

Enumeration Value	Description
off (Never Show Scrollbar)	Specifies that the scrollbar for a frame shall always be hidden.
on (Always Show Scrollbar)	Specifies that the scrollbar for a frame shall always be displayed (even when not needed).

Referenced By
scrollbar@val (§2.15.2.37)

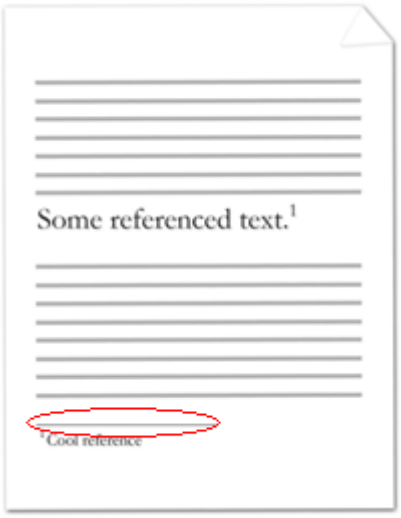
The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FrameScrollbar">
   <restriction base="xsd:string">
      <enumeration value="on"/>
      <enumeration value="off"/>
      <enumeration value="auto"/>
   </restriction>
</simpleType>
```

2.18.37 ST_FtnEdn (Footnote or Endnote Type)

This simple type specifies the possible types of footnotes and endnotes which may be specified in a WordprocessingML document.

[*Example*: Consider a document with a single footnote at the bottom of the first page. This footnote shall be separated from the text by the `separator` footnote (the footnote explicitly used to separate text from the footnote list (circled in red below):



This footnote type would be declared as follows in the WordprocessingML:

```
<w:footnote w:type="separator" w:id="0">
```

```
    …
</w:footnote>
```

In this example, the footnote has a type value of `separator`, specifies when this footnote shall be used. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
continuationNotice (Continuation Notice Separator)	Specifies that this footnote or endnote is a continuation notice footnote or endnote. *Continuation notice footnotes and endnotes* are used when the footnotes or endnotes exceed the length allowed on a single page. When this happens, this footnote or endnote shall be placed on the bottom of each page where the note shall continue to indicate that fact to the reader.
continuationSeparator (Continuation Separator)	Specifies that this footnote or endnote is a continuation separator footnote or endnote. *Continuation separator footnotes and endnotes* are used when the footnotes or endnotes exceed the length allowed on a single page. When this happens, this footnote or endnote shall be placed between the main text contents and the continued footnotes/endnotes on all subsequent pages of the document.
normal (Normal Footnote/Endnote)	Specifies that this footnote or endnote is a normal footnote or endnote, and can be referenced by main document content.
separator (Separator)	Specifies that this footnote or endnote is a separator footnote or endnote. *Separator footnotes and endnotes* are used to indicate the separation between the main document's content and the footnotes or endnotes to indicate that fact to the reader.

Referenced By
endnote@type (§2.11.2); footnote@type (§2.11.10)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FtnEdn">
   <restriction base="xsd:string">
      <enumeration value="normal"/>
      <enumeration value="separator"/>
      <enumeration value="continuationSeparator"/>
      <enumeration value="continuationNotice"/>
   </restriction>
</simpleType>
```

2.18.38 ST_FtnPos (Footnote Positioning Location)

This simple type specifies the position of footnotes in the document.

[*Example*: Consider a document in which footnotes shall be positioned beneath their text. The footnote properties for this document shall be declared as follows:

```
<w:sectPr>
  <w:footnotePr>
    <w:pos w:val="beneathText" />
  </w:footnotePr>
  …
</w:sectPr>
```

The val attribute is `beneathText`, therefore the position of footnotes is specified to be beneath the page's text. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
beneathText (Footnotes Positioned Beneath Text)	Specifies that footnotes shall be displayed immediately after the last line of text on the page on which the note reference mark appears.
docEnd (Footnotes Positioned At End of Document)	Specifies that all footnotes shall be placed at the end of the current document, regardless of which section they are referenced within.
pageBottom (Footnotes Positioned at Page Bottom)	Specifies that footnotes shall be displayed at the bottom margin of the page on which the note reference mark appears.
sectEnd (Footnotes Positioned At End of Section)	Specifies that all footnotes shall be placed at the end of the section in which they are referenced. A footnote which is never referenced is never displayed.

Referenced By
pos@val (§2.11.21)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FtnPos">
   <restriction base="xsd:string">
      <enumeration value="pageBottom"/>
      <enumeration value="beneathText"/>
      <enumeration value="sectEnd"/>
      <enumeration value="docEnd"/>
   </restriction>
</simpleType>
```

2.18.39 ST_Guid (128-Bit GUID)

This simple type specifies that its values shall be a 128-bit globally unique identifier (GUID) value.

[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:

```
<w:docPartPr>
  …
  <w:guid w:val="{00000000-5BD2-4BC8-9F70-7020E1357FB2}" />
  …
<w:docPartPr>
```

The guid element specifies that the unique identifier associated with the parent entry shall be `{00000000-5BD2-4BC8-9F70-7020E1357FB2}`. This value may be used as needed by an application, for example, to uniquely identify a part regardless of its name. *end example*]

This simple type's contents are a restriction of the XML Schema token datatype.

This simple type also specifies the following restrictions:

- This simple type's contents must match the following regular expression pattern: \{[0-9A-F]{8}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{12}\}.

Referenced By
embedBold@fontKey (§2.8.2.3); embedBoldItalic@fontKey (§2.8.2.4); embedItalic@fontKey (§2.8.2.5); embedRegular@fontKey (§2.8.2.6); guid@val (§2.12.11)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Guid">
   <restriction base="xsd:token">
      <pattern value="\{[0-9A-F]{8}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{12}\}"/>
   </restriction>
</simpleType>
```

2.18.40 ST_HAnchor (Horizontal Anchor Location)

This simple type specifies the horizontal position to which the parent object has been anchored in the document. This anchor position shall be used as the base location to determine the final horizontal position of the object in the document.

[*Example*: Consider a text frame which should be positioned one inch to the right of its column in a left-to-right document. This text frame would be specified using the following WordprocessingML:

```
<w:pPr>
  <w:framePr … w:x="1440" w:hAnchor="margin" />
</w:pPr>
```

These frame horizontal anchor properties specify that they are relative to the anchor paragraph's `margin` (the text margin excluding any indents). *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
margin (Relative To Margin)	Specifies that the parent object shall be horizontally anchored to the text margins. This shall be used to specify that any horizontal positioning values shall be calculated with respect to the location of the text margin.
page (Relative to Page)	Specifies that the parent object shall be horizontally anchored to the page edge. This shall be used to specify that any horizontal positioning values shall be calculated with respect to the location of the edge of the page.
text (Relative to Text Extents)	Specifies that the parent object shall be horizontally anchored to the text extents. This shall be used to specify that any horizontal positioning values shall be calculated with respect to the location of the edge of the text in the anchor paragraph (including text indentations on that paragraph within the text margins).

Referenced By
framePr@hAnchor (§2.3.1.11); tblpPr@horzAnchor (§2.4.54)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_HAnchor">
   <restriction base="xsd:string">
      <enumeration value="text"/>
      <enumeration value="margin"/>
      <enumeration value="page"/>
   </restriction>
</simpleType>
```

2.18.41 ST_HdrFtr (Header or Footer Type)

This simple type specifies the possible types of headers and footers which may be specified for a given header or footer reference in a document. This value determines the page(s) on which the current header or footer shall be displayed.

[*Example*: Consider a WordprocessingML section which specifies the following header reference:

```
<w:headerReference r:id="rId10" w:type="first" />
```

The resulting section shall use the specified header part for the `first` page. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
default (Default Header or Footer)	Specifies that this header or footer shall appear on every page in this section which is not overridden with a specific `even` or `first` page header/footer. In a section with all three types specified, this type shall be used on all odd numbered pages (counting from the first page in the section, not the section numbering).
even (Even Numbered Pages Only)	Specifies that this header or footer shall appear on all even numbered pages in this section (counting from the first page in the section, not the section numbering). The appearance of this header or footer is contingent on the setting of the evenAndOddHeaders element (§2.10.1).
first (First Page Only)	Specifies that this header or footer shall appear on the first page in this section. The appearance of this header or footer is contingent on the setting of the titlePg element (§2.10.6).

Referenced By
footerReference@type (§2.10.2); headerReference@type (§2.10.5)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_HdrFtr">
   <restriction base="xsd:string">
      <enumeration value="even"/>
      <enumeration value="default"/>
      <enumeration value="first"/>
   </restriction>
</simpleType>
```

2.18.42 ST_HeightRule (Height Rule)

This simple type specifies the logic which shall be used to calculate the height of the parent object when it is displayed in the document.

[*Example*: Consider the following table row:

```
<w:trPr>
  <w:trHeight w:hRule=”atLeast” w:val=”2189” />
</w:trPr>
```

The val attribute specifies a value of 2189 twentieths of a point, so this table row will be a minimum of 2189 twentieths of a point high regardless of its contents, since its hRule value is set to `atLeast`. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
atLeast (Minimum Height)	Specifies that the height of the parent object shall be at least the value specified, but may be expanded to fit its content as needed.
auto (Determine Height Based On Contents)	Specifies that the height of the parent object shall be automatically determined by the size of its contents, with no predetermined minimum or maximum size.
exact (Exact Height)	Specifies that the height of the parent object shall be exactly the value specified, regardless of the size of the contents of the object. If the contents are too large for the specified height, then they shall be clipped.

Referenced By

Referenced By
framePr@hRule (§2.3.1.11); trHeight@hRule (§2.4.77)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_HeightRule">
   <restriction base="xsd:string">
      <enumeration value="auto"/>
      <enumeration value="exact"/>
      <enumeration value="atLeast"/>
   </restriction>
</simpleType>
```

2.18.43 ST_HexColor (Color Value)

This simple type specifies that its contents will contain one of the following:

- A color values in RRGGBB format (ST_HexColorRGB)
- The enumeration value `auto` (ST_HexColorAuto)

The contents of this measurement are interpreted based on the context of the parent XML element.

[*Example*: Consider a border color with value `auto`, as follows:

```
<w:bottom … w:color="auto"/>
```

This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*]

This simple type is defined as a union of the following types:

- TheST_HexColorAuto simple type (§2.18.44).
- TheST_HexColorRGB simple type (§2.18.45).

Referenced By
background@color (§2.2.1); bar@color (§2.3.1.4); bdr@color (§2.3.2.3); between@color (§2.3.1.5); bottom@color (§2.6.2); bottom@color (§2.4.3); bottom@color (§2.4.4); bottom@color (§2.15.2.4); bottom@color (§2.3.1.7); color@val (§2.3.2.5); color@val (§2.15.2.5); insideH@color (§2.4.17); insideH@color (§2.4.18); insideV@color (§2.4.19); insideV@color (§2.4.20); left@color (§2.15.2.21); left@color (§2.4.24); left@color (§2.6.7); left@color (§2.3.1.17); left@color (§2.4.27); right@color (§2.3.1.28); right@color (§2.4.30); right@color (§2.6.15); right@color (§2.15.2.35); right@color (§2.4.32); shd@color (§2.3.2.30); shd@color (§2.4.33); shd@color (§2.4.34); shd@color (§2.4.35); shd@color (§2.3.1.31); shd@fill (§2.3.2.30); shd@fill (§2.4.33); shd@fill (§2.4.34); shd@fill (§2.4.35); shd@fill (§2.3.1.31); tl2br@color (§2.4.70); top@color (§2.4.71); top@color (§2.3.1.42); top@color (§2.15.2.42); top@color (§2.4.74); top@color (§2.6.21); tr2bl@color (§2.4.76); u@color (§2.3.2.38)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_HexColor">
   <union memberTypes="ST_HexColorAuto ST_HexColorRGB"/>
</simpleType>
```

2.18.44 ST_HexColorAuto ('Automatic' Color Value)

This simple type specifies that its contents will contain the enumeration value auto. This value shall be used to specify an automatically determined color value, the meaning of which is interpreted based on the context of the parent XML element.

[*Example*: Consider a border color with value auto, as follows:

```
<w:bottom … w:color="auto"/>
```

This color therefore may be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
auto (Automatically Determined Color)	Specifies that the color value may automatically be defined when this document is processed, based on the display context.

Referenced By
ST_HexColor (§2.18.43)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_HexColorAuto">
   <restriction base="xsd:string">
      <enumeration value="auto"/>
   </restriction>
</simpleType>
```

2.18.45 ST_HexColorRGB (Hexadecimal Color Value)

This simple type specifies that its contents shall contain a color value in RRGGBB hexadecimal format. This specifies that each of the red, green, and blue color values form 0-255 will be encoded as a two-digit hexadecimal number.

[*Example*: Consider a color defined as follows:

```
Red:      122
Green:    23
```

```
Blue:     209
```

The resulting RRGGBB value would be 7A17D1, as each color is transformed into its hexadecimal equivalent. *end example*]

This simple type's contents are a restriction of the XML Schema hexBinary datatype.

This simple type also specifies the following restrictions:

- This simple type's contents must have a length of exactly 3 characters.

Referenced By
ST_HexColor (§2.18.43)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_HexColorRGB">
   <restriction base="xsd:hexBinary">
      <length value="3" fixed="true"/>
   </restriction>
</simpleType>
```

2.18.46 ST_HighlightColor (Text Highlight Colors)

This simple type specifies the possible values for highlighting colors which may be applied as a background behind the contents of a text run.

[*Example*: Consider a run within a paragraph which has yellow text highlighting using the highlight element. This formatting is specified using the following WordprocessingML:

```
<w:rPr>
  <w:highlight w:val="yellow" />
</w:rPr>
```

The resulting run would have yellow highlighting visible over its contents. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
black (Black Highlighting Color)	Specifies that the text highlighting color for this run shall be black. The hexadecimal RGB value for this setting shall be 000000.
blue (Blue Highlighting Color)	Specifies that the text highlighting color for this run shall be blue.

Enumeration Value	Description
	The hexadecimal RGB value for this setting shall be 0000FF.
cyan (Cyan Highlighting Color)	Specifies that the text highlighting color for this run shall be cyan. The hexadecimal RGB value for this setting shall be 00FFFF.
darkBlue (Dark Blue Highlighting Color)	Specifies that the text highlighting color for this run shall be dark blue. The hexadecimal RGB value for this setting shall be 000080.
darkCyan (Dark Cyan Highlighting Color)	Specifies that the text highlighting color for this run shall be dark cyan. The hexadecimal RGB value for this setting shall be 008080.
darkGray (Dark Gray Highlighting Color)	Specifies that the text highlighting color for this run shall be dark gray. The hexadecimal RGB value for this setting shall be 808080.
darkGreen (Dark Green Highlighting Color)	Specifies that the text highlighting color for this run shall be dark green. The hexadecimal RGB value for this setting shall be 008000.
darkMagenta (Dark Magenta Highlighting Color)	Specifies that the text highlighting color for this run shall be dark magenta. The hexadecimal RGB value for this setting shall be 800080.
darkRed (Dark Red Highlighting Color)	Specifies that the text highlighting color for this run shall be dark red. The hexadecimal RGB value for this setting shall be 800000.
darkYellow (Dark Yellow Highlighting Color)	Specifies that the text highlighting color for this run shall be dark cyan. The hexadecimal RGB value for this setting shall be 808000.

Enumeration Value	Description
green (Green Highlighting Color)	Specifies that the text highlighting color for this run shall be green. The hexadecimal RGB value for this setting shall be `00FF00`.
lightGray (Light Gray Highlighting Color)	Specifies that the text highlighting color for this run shall be light gray. The hexadecimal RGB value for this setting shall be `C0C0C0`.
magenta (Magenta Highlighting Color)	Specifies that the text highlighting color for this run shall be magenta. The hexadecimal RGB value for this setting shall be `FF00FF`.
none (No Text Highlighting)	Specifies that this text run shall have no text highlighting applied to its contents.
red (Red Highlighting Color)	Specifies that the text highlighting color for this run shall be red. The hexadecimal RGB value for this setting shall be `FF0000`.
white (White Highlighting Color)	Specifies that the text highlighting color for this run shall be white. The hexadecimal RGB value for this setting shall be `FFFFFF`.
yellow (Yellow Highlighting Color)	Specifies that the text highlighting color for this run shall be yellow. The hexadecimal RGB value for this setting shall be `FFFF00`.

Referenced By
highlight@val (§2.3.2.13)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_HighlightColor">
   <restriction base="xsd:string">
      <enumeration value="black"/>
      <enumeration value="blue"/>
      <enumeration value="cyan"/>
      <enumeration value="green"/>
      <enumeration value="magenta"/>
      <enumeration value="red"/>
      <enumeration value="yellow"/>
      <enumeration value="white"/>
      <enumeration value="darkBlue"/>
      <enumeration value="darkCyan"/>
      <enumeration value="darkGreen"/>
      <enumeration value="darkMagenta"/>
      <enumeration value="darkRed"/>
      <enumeration value="darkYellow"/>
      <enumeration value="darkGray"/>
      <enumeration value="lightGray"/>
      <enumeration value="none"/>
   </restriction>
</simpleType>
```

2.18.47 ST_Hint (Font Type Hint)

Specifies the font type which shall be used to format any ambiguous characters in the current run.

There are certain characters which are not explicitly stored in the document, and may be mapped into multiple categories of the four mentioned above. This attribute shall be used to arbitrate that conflict, and determine how ambiguities in this run shall be handled. [*Note*: This is primarily used to handle the formatting on the paragraph mark glyph, and other characters that are not stored as text in the WordprocessingML document. *end note*]

[*Example*: Consider the run representing the paragraph mark glyph, which is not stored as a physical character. Since this could therefore be formatted with any of the fonts specified for the run, this ambiguity is resolved using the following WordprocessingML:

```
<w:pPr>
  <w:rPr>
    <w:rFonts w:hint="eastAsia" />
  </w:rPr>
</w:pPr>
```

The `hint` attribute specifies that the run shall use the `eastAsia` font (theme or not, whichever is in use for East Asian text) as applied for this run. *end example*]

This simple type's contents are a restriction of the XML Schema `string` datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
cs (Complex Script Font)	Specifies that the font hint for this text run shall be to use the Complex Script font defined on the run via the style hierarchy.
default (High ANSI Font)	Specifies that the font hint for this text run shall be to use the High ANSI font defined on the run via the style hierarchy.
eastAsia (East Asian Font)	Specifies that the font hint for this text run shall be to use the East Asian font defined on the run via the style hierarchy.

Referenced By
rFonts@hint (§2.3.2.24)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Hint">
   <restriction base="xsd:string">
      <enumeration value="default"/>
      <enumeration value="eastAsia"/>
      <enumeration value="cs"/>
   </restriction>
</simpleType>
```

2.18.48 ST_HpsMeasure (Measurement in Half-Points)

This simple type specifies that its contents will contain a positive whole number, whose contents consist of a measurement in half-points (equivalent to 1/144th of an inch).

The contents of this measurement are interpreted based on the context of the parent XML element.

[*Example*: Consider an attribute value of 72 whose type is ST_HpsMeasure. This attribute value specifies a size of one-half of an inch or 36 points (72 halves of a point = 36 points = 0.5 inches). *end example*]

This simple type's contents are a restriction of the ST_UnsignedDecimalNumber simple type (§2.18.108).

Referenced By
hps@val (§2.3.3.10); hpsBaseText@val (§2.3.3.11); hpsRaise@val (§2.3.3.12); kern@val (§2.3.2.17); size@val (§2.16.30); sz@val (§2.3.2.36); szCs@val (§2.3.2.37)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_HpsMeasure">
   <restriction base="ST_UnsignedDecimalNumber"/>
</simpleType>
```

2.18.49 ST_InfoTextType (Help or Status Text Type)

This simple type specifies the possible values for the type of help or status text which may be associated with a form field.

[*Example*: Consider the following WordprocessingML fragment for a form field:

```
<w:ffData>
  <w:helpText w:type="text" w:val="Example help text." />
</w:ffData>
```

The type attribute has a value of `text`, which specifies that the text in the val attribute is the literal help text for this form field. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
autoText (Glossary Document Entry)	Specifies that the value specified by the parent XML element's val attribute shall be interpreted as the name of a glossary document entry whose contents contain the help or status text.
text (Literal Text)	Specifies that the value specified by the parent XML element's val attribute shall be interpreted as the literal text for the help or status text.

Referenced By
helpText@type (§2.16.23); statusText@type (§2.16.32)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_InfoTextType">
   <restriction base="xsd:string">
      <enumeration value="text"/>
      <enumeration value="autoText"/>
   </restriction>
</simpleType>
```

2.18.50 ST_Jc (Horizontal Alignment Type)

This simple type specifies all types of alignment which are available to be applied to objects in a WordprocessingML document.

[*Example*: Consider a paragraph which is right aligned. This requirement would be specified as follows in the WordprocessingML markup:

```
<w:pPr>
  <w:jc w:val="right" />
</w:pPr>
```

The val attribute's value of `right` specifies that the content shall be right aligned on the page. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
both (Justified)	Specifies that the text shall be justified between both of the text margins in the document. The `lowKashida` setting shall also be applied to Arabic text when this setting is applied. This type of justification shall only affect the inter-word spacing on each line, and not the inter-character spacing between each word when justifying its contents.
center (Align Center)	Specifies that the text shall be centered on the line between both of the text margins in the document.
distribute (Distribute All Characters Equally)	Specifies that the text shall be justified between both of the text margins in the document. This type of justification shall equally affect the inter-word spacing on each line as well as the inter-character spacing between each word when justifying its contents - that is, an equal amount of additional character pitch shall be added to all characters on the line.
highKashida (Widest Kashida Length)	Specifies that the kashida length for text in the current paragraph shall be extended to its widest possible length. This setting only affects *kashidas*, which are special characters used to extend the joiner between two Arabic characters. [*Note*: They are typically used to improve the appearance of justified text by visually lengthening words rather than increasing the spacing between words. *end note*] [*Example*: The following example illustrates each type of kashida:

Enumeration Value	Description
	Low: هذا إمتيحان Medium: هذا إمتيحـــان High: هذا إمتيحـــــان *end example*]
left (Align Left)	Specifies that the text shall be aligned on the left text margin in the document.
lowKashida (Low Kashida Length)	Specifies that the kashida length for text in the current paragraph shall be extended to a slightly longer length. This setting shall also be applied to Arabic text when the both setting is applied. This setting only affects *kashidas*, which are special characters used to extend the joiner between two Arabic characters. [*Note*: They are typically used to improve the appearance of justified text by visually lengthening words rather than increasing the spacing between words. *end note*] [*Example*: The following example illustrates each type of kashida: Low: هذا إمتيحان Medium: هذا إمتيحـــان High: هذا إمتيحـــــان *end example*]
mediumKashida (Medium Kashida Length)	Specifies that the kashida length for text in the current paragraph shall be extended to a medium length determined by the consumer. This setting only affects *kashidas*, which are special characters used to extend the joiner between two Arabic characters. [*Note*: They are typically used to improve the appearance of justified text by visually lengthening words rather than increasing the spacing between words. *end note*] [*Example*: The following example illustrates each type of kashida:

Enumeration Value	Description
	Low: هذا إمتيحان Medium: هذا إمتيحـان High: هذا إمتيحـــان *end example*]
numTab (Align to List Tab)	Specifies that the text shall be aligned to the list tab, which is the tab stop after the numbering for the current paragraph. If the current paragraph has no numbering, this setting has no effect. [*Note*: This justification style is used for backwards compatibility with earlier word processors, and should be deprecated in favor of hanging paragraph indentation. *end note*]
right (Align Right)	Specifies that the text shall be aligned on the right text margin in the document.
thaiDistribute (Thai Language Justification)	Specifies that the text shall be justified with an optimization for Thai. This type of justification shall affect both the inter-word spacing on each line, and the inter-character spacing between each word when justifying its contents, unlike both justification. This difference is created in that the inter-character space is increased slightly in order to ensure that the additional space created by the justification is reduced. [*Note*: This setting is different from justification in that the reduction in inter-character spacing would be inappropriate in Western languages. *end note*]

Referenced By
jc@val (§2.4.21); jc@val (§2.4.22); jc@val (§2.3.1.13); jc@val (§2.4.23); lvlJc@val (§2.9.8)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Jc">
   <restriction base="xsd:string">
      <enumeration value="left"/>
      <enumeration value="center"/>
      <enumeration value="right"/>
      <enumeration value="both"/>
      <enumeration value="mediumKashida"/>
      <enumeration value="distribute"/>
      <enumeration value="numTab"/>
      <enumeration value="highKashida"/>
      <enumeration value="lowKashida"/>
      <enumeration value="thaiDistribute"/>
   </restriction>
</simpleType>
```

2.18.51 ST_Lang (Language Reference)

This simple type specifies that its contents will contain one of the following:

- A hexadecimal language code (ST_LangCode)
- An *ISO 639-1* letter code plus a dash plus an *ISO 3166-1 alpha-2* letter code (ST_String)

The contents of this language are interpreted based on the context of the parent XML element.

[*Example*: Consider a language code defined as follows :

```
<w:lang w:val=”en-CA” />
```

This language is therefore specified as English (en) and Canada (CA), resulting in use of the English (Canada) language setting. *end example*]

This simple type is defined as a union of the following types:

- TheST_LangCode simple type (§2.18.52).
- TheST_String simple type (§2.18.89).

Referenced By
activeWritingStyle@lang (§2.15.1.1); lang@bidi (§2.3.2.18); lang@eastAsia (§2.3.2.18); lang@val (§2.3.2.18); lid@val (§2.3.3.14); lid@val (§2.5.2.19); lid@val (§2.14.17); noLineBreaksAfter@lang (§2.15.1.58); noLineBreaksBefore@lang (§2.15.1.59); themeFontLang@bidi (§2.15.1.89); themeFontLang@eastAsia (§2.15.1.89); themeFontLang@val (§2.15.1.89)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Lang">
   <union memberTypes="ST_LangCode ST_String"/>
</simpleType>
```

2.18.52 ST_LangCode (Two Digit Hexadecimal Language Code)

This simple type specifies that its contents will contain a two digit hexadecimal language code defined as follows:

Language Code	Language - Country/Region
1025	Arabic - Saudi Arabia
1026	Bulgarian
1027	Catalan
1028	Chinese - Taiwan
1029	Czech
1030	Danish
1031	German - Germany
1032	Greek
1033	English - United States
1034	Spanish - Spain (Traditional Sort)
1035	Finnish
1036	French - France
1037	Hebrew
1038	Hungarian
1039	Icelandic
1040	Italian - Italy
1041	Japanese
1042	Korean
1043	Dutch - Netherlands
1044	Norwegian (Bokmål)
1045	Polish
1046	Portuguese - Brazil
1047	Rhaeto-Romanic
1048	Romanian
1049	Russian
1050	Croatian
1051	Slovak
1052	Albanian - Albania
1053	Swedish
1054	Thai
1055	Turkish

Language Code	Language - Country/Region
1056	Urdu
1057	Indonesian
1058	Ukrainian
1059	Belarusian
1060	Slovenian
1061	Estonian
1062	Latvian
1063	Lithuanian
1064	Tajik
1065	Farsi
1066	Vietnamese
1067	Armenian - Armenia
1068	Azeri (Latin)
1069	Basque
1070	Sorbian
1071	FYRO Macedonian
1072	Sutu
1073	Tsonga
1074	Tswana
1075	Venda
1076	Xhosa
1077	Zulu
1078	Afrikaans - South Africa
1079	Georgian
1080	Faroese
1081	Hindi
1082	Maltese
1083	Sami (Lappish)
1084	Gaelic (Scotland)
1085	Yiddish
1086	Malay - Malaysia
1087	Kazakh
1088	Kyrgyz (Cyrillic)
1089	Swahili

Language Code	Language - Country/Region
1090	Turkmen
1091	Uzbek (Latin)
1092	Tatar
1093	Bengali (India)
1094	Punjabi
1095	Gujarati
1096	Oriya
1097	Tamil
1098	Telugu
1099	Kannada
1100	Malayalam
1101	Assamese
1102	Marathi
1103	Sanskrit
1104	Mongolian (Cyrillic)
1105	Tibetan - People's Republic of China
1106	Welsh
1107	Khmer
1108	Lao
1109	Burmese
1110	Galician
1111	Konkani
1112	Manipuri
1113	Sindhi - India
1114	Syriac
1115	Sinhalese - Sri Lanka
1116	Cherokee - United States
1117	Inuktitut
1118	Amharic - Ethiopia
1119	Tamazight (Arabic)
1120	Kashmiri (Arabic)
1121	Nepali
1122	Frisian - Netherlands
1123	Pashto

Language Code	Language - Country/Region
1124	Filipino
1125	Divehi
1126	Edo
1127	Fulfulde - Nigeria
1128	Hausa - Nigeria
1129	Ibibio - Nigeria
1130	Yoruba
1131	Quecha - Bolivia
1132	Sepedi
1136	Igbo - Nigeria
1137	Kanuri - Nigeria
1138	Oromo
1139	Tigrigna - Ethiopia
1140	Guarani - Paraguay
1141	Hawaiian - United States
1142	Latin
1143	Somali
1144	Yi
1145	Papiamentu
1152	Uighur - China
1153	Maori - New Zealand
1279	HID (Human Interface Device)
2049	Arabic - Iraq
2052	Chinese - People's Republic of China
2055	German - Switzerland
2057	English - United Kingdom
2058	Spanish - Mexico
2060	French - Belgium
2064	Italian - Switzerland
2067	Dutch - Belgium
2068	Norwegian (Nynorsk)
2070	Portuguese - Portugal
2072	Romanian - Moldava
2073	Russian - Moldava

Language Code	Language - Country/Region
2074	Serbian (Latin)
2077	Swedish - Finland
2080	Urdu - India
2092	Azeri (Cyrillic)
2108	Gaelic (Ireland)
2110	Malay - Brunei Darussalam
2115	Uzbek (Cyrillic)
2117	Bengali (Bangladesh)
2118	Punjabi (Pakistan)
2128	Mongolian (Mongolian)
2129	Tibetan - Bhutan
2137	Sindhi - Pakistan
2143	Tamazight (Latin)
2144	Kashmiri
2145	Nepali - India
2155	Quecha - Ecuador
2163	Tigrigna - Eritrea
3073	Arabic - Egypt
3076	Chinese - Hong Kong SAR
3079	German - Austria
3081	English - Australia
3082	Spanish - Spain (Modern Sort)
3084	French - Canada
3098	Serbian (Cyrillic)
3179	Quecha - Peru
4097	Arabic - Libya
4100	Chinese - Singapore
4103	German - Luxembourg
4105	English - Canada
4106	Spanish - Guatemala
4108	French - Switzerland
4122	Croatian (Bosnia/Herzegovina)
5121	Arabic - Algeria
5124	Chinese - Macao SAR

Language Code	Language - Country/Region
5127	German - Liechtenstein
5129	English - New Zealand
5130	Spanish - Costa Rica
5132	French - Luxembourg
5146	Bosnian (Bosnia/Herzegovina)
6145	Arabic - Morocco
6153	English - Ireland
6154	Spanish - Panama
6156	French - Monaco
7169	Arabic - Tunisia
7177	English - South Africa
7178	Spanish - Dominican Republic
7180	French - West Indies
8193	Arabic - Oman
8201	English - Jamaica
8202	Spanish - Venezuela
8204	French - Reunion
9217	Arabic - Yemen
9225	English - Caribbean
9226	Spanish - Colombia
9228	French - Democratic Rep. of Congo
10241	Arabic - Syria
10249	English - Belize
10250	Spanish - Peru
10252	French - Senegal
11265	Arabic - Jordan
11273	English - Trinidad
11274	Spanish - Argentina
11276	French - Cameroon
12289	Arabic - Lebanon
12297	English - Zimbabwe
12298	Spanish - Ecuador
12300	French - Cote d'Ivoire
13313	Arabic - Kuwait

Language Code	Language - Country/Region
13321	English - Philippines
13322	Spanish - Chile
13324	French - Mali
14337	Arabic - U.A.E.
14345	English - Indonesia
14346	Spanish - Uruguay
14348	French - Morocco
15361	Arabic - Bahrain
15369	English - Hong Kong SAR
15370	Spanish - Paraguay
15372	French - Haiti
16385	Arabic - Qatar
16393	English - India
16394	Spanish - Bolivia
17417	English - Malaysia
17418	Spanish - El Salvador
18441	English - Singapore
18442	Spanish - Honduras
19466	Spanish - Nicaragua
20490	Spanish - Puerto Rico
21514	Spanish - United States
58378	Spanish - Latin America
58380	French - North Africa
Any other value	Undefined. Shall not be used.

The contents of this language are interpreted based on the context of the parent XML element.

[*Example*: Consider a language code defined as follows :

```
<w:lang w:val="1033" />
```

This language is therefore specified as English (U.S.) which maps to a hexadecimal language setting of 1033. *end example*]

This simple type's contents are a restriction of the XML Schema hexBinary datatype.

This simple type also specifies the following restrictions:

- This simple type's contents must have a length of exactly 2 characters.

Referenced By
ST_Lang (§2.18.51)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_LangCode">
   <restriction base="xsd:hexBinary">
      <length value="2" fixed="true"/>
   </restriction>
</simpleType>
```

2.18.53 ST_LevelSuffix (Content Between Numbering Symbol and Paragraph Text)

This simple type specifies the types of content which shall be possible between a given numbering level's text and the text of every numbered paragraph which references that numbering level.

[*Example:* Consider the numbered paragraph below:

1. Test

In this example, a space exists between the numbering symbol `1.` and the numbered paragraph text `Test`. The space would be specified in WordprocessingML as follows:

```
<w:lvl w:ilvl="0">
  …
  <w:suff w:val="space" />
  …
</w:lvl>
```

The suff element with an attribute value of space specifies that the character between the numbering's level text and the paragraph text shall be a space. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
nothing (Nothing Between Numbering and Text)	Specifies that no character shall be displayed between the numbering level's text and the contents of the paragraph when displaying the numbered paragraph.
space (Space Between Numbering and Text)	Specifies that a space character shall be displayed between the numbering level's text and the contents of the paragraph when displaying the numbered paragraph.
tab (Tab Between Numbering and Text)	Specifies that a tab character shall be displayed between the numbering level's text and the contents of the paragraph when displaying the numbered

Enumeration Value	Description
	paragraph. This tab shall follow normal tab stop rules to determine its length.

Referenced By
suff@val (§2.9.30)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_LevelSuffix">
   <restriction base="xsd:string">
      <enumeration value="tab"/>
      <enumeration value="space"/>
      <enumeration value="nothing"/>
   </restriction>
</simpleType>
```

2.18.54 ST_LineNumberRestart (Line Numbering Restart Position)

This simple type specifies when the line numbering in the parent section shall be reset to its restart value. The line numbering increments for each line (even if the line number itself is not displayed) until it reaches the restart point specified by this element.

[*Example*: Consider the line numbering used on each page of this document, which specifies that line numbering shall restart at the top of each new page. This line numbering setting would be defined using the following WordprocessingML:

```
<w:lnNumType w:restart="newPage" ... />
```

The restart attribute is of type ST_LineNumberRestart, and a value of newPage specifies that the line numbers shall restart at the top of each page to the value specified by the start attribute. In this case, newPage is the default, so this value could have been omitted entirely. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
continuous (Continue Line Numbering From Previous Section)	Specifies that line numbering for the parent section shall continue from the line numbering from the end of the previous section, if any.
newPage (Restart Line Numbering on Each Page)	Specifies that line numbering for the parent section shall restart to the starting value whenever a new page is displayed.

Enumeration Value	Description
newSection (Restart Line Numbering for Each Section)	Specifies that line numbering for the parent section shall restart to the starting value whenever the parent begins.

Referenced By
lnNumType@restart (§2.6.8)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_LineNumberRestart">
   <restriction base="xsd:string">
      <enumeration value="newPage"/>
      <enumeration value="newSection"/>
      <enumeration value="continuous"/>
   </restriction>
</simpleType>
```

2.18.55 ST_LineSpacingRule (Line Spacing Rule)

This simple type specifies the logic which shall be used to calculate the line spacing of the parent object when it is displayed in the document.

[*Example*: Consider the following WordprocessingML paragraph:

```
<w:pPr>
  <w:spacing w:line="276" w:lineRule="auto" />
</w:pPr>
```

This paragraph specifies that the spacing in each line should be automatically calculated using 1.15 times (276 divided by 240) the normal single spacing calculation. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
atLeast (Minimum Line Height)	Specifies that the height of the line shall be at least the value specified, but may be expanded to fit its content as needed.
auto (Automatically Determined Line Height)	Specifies that the line spacing of the parent object shall be automatically determined by the size of its contents, with no predetermined minimum or maximum size.
exact (Exact Line Height)	Specifies that the height of the line shall be exactly the value specified, regardless of the size of the contents

Enumeration Value	Description
	of the contents. If the contents are too large for the specified height, then they shall be clipped as necessary.

Referenced By
spacing@lineRule (§2.3.1.33)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_LineSpacingRule">
   <restriction base="xsd:string">
      <enumeration value="auto"/>
      <enumeration value="exact"/>
      <enumeration value="atLeast"/>
   </restriction>
</simpleType>
```

2.18.56 ST_Lock (Locking Types)

This simple type specifies the possible set of locking behaviors which may be applied to the contents of the parent structured document tag when the contents of this documents are edited by an application (whether through a user interface or directly).

[*Example*: Consider the following plain text structured document tag:

```
<w:sdt>
  <w:sdtPr>
    <w:lock w:val="sdtLocked"/>
    …
    <w:text/>
  </w:sdtPr>
  …
</w:sdt>
```

This plain text structured document tag's properties contain a lock element, specifying locking behaviors for the structured document tag. Since the locking val attribute value is sdtLocked, this locking setting shall specify that the contents of the structured document tag may be edited, but the structured document tag itself shall not be deleted from the document. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description

Enumeration Value	Description
contentLocked (Contents Cannot Be Edited At Runtime)	Specifies that the editing restriction applied to the parent structured document tag shall be as follows: • This structured document tag's contents shall not be editable • This structured document tag may be deleted in its entirety (but only entirely, no sub portion of it may be deleted)
sdtContentLocked (Contents Cannot Be Edited At Runtime And SDT Cannot Be Deleted)	Specifies that the editing restriction applied to the parent structured document tag shall be as follows: • This structured document tag's contents shall not be editable • This structured document tag shall not be deleted in its entirety
sdtLocked (SDT Cannot Be Deleted)	Specifies that the editing restriction applied to the parent structured document tag shall be as follows: • This structured document tag's contents shall be editable • This structured document tag shall not be deleted in its entirety
unlocked (No Locking)	Specifies that no special locking behaviors shall be applied to the parent structured document tag. The default behaviors as specified on the lock element (§2.5.2.22) shall be used.

Referenced By
lock@val (§2.5.2.22)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Lock">
   <restriction base="xsd:string">
      <enumeration value="sdtLocked"/>
      <enumeration value="contentLocked"/>
      <enumeration value="unlocked"/>
      <enumeration value="sdtContentLocked"/>
   </restriction>
</simpleType>
```

2.18.57 ST_LongHexNumber (Four Digit Hexadecimal Number Value)

This simple type specifies a number value specified as a four octet (eight digit) hexadecimal number), whose contents are interpreted based on the context of the parent XML element.

[*Example*: Consider the following value for a node of type ST_LongHexNumber: `00BE2C6C`.

This value is valid, as it contains four hexadecimal octets, each an encoding of an octet of the actual decimal number value. *end example*]

This simple type's contents are a restriction of the XML Schema hexBinary datatype.

This simple type also specifies the following restrictions:

- This simple type's contents must have a length of exactly 4 characters.

Referenced By
documentProtection@algIdExt (§2.15.1.28); documentProtection@cryptProviderTypeExt (§2.15.1.28); lvl@tplc (§2.9.6); lvl@tplc (§2.9.7); nsid@val (§2.9.15); p@rsidDel (§2.3.1.22); p@rsidP (§2.3.1.22); p@rsidR (§2.3.1.22); p@rsidRDefault (§2.3.1.22); p@rsidRPr (§2.3.1.22); r@rsidDel (§2.3.2.23); r@rsidR (§2.3.2.23); r@rsidRPr (§2.3.2.23); rsid@val (§2.7.3.15); rsid@val (§2.15.1.70); rsidRoot@val (§2.15.1.71); sectPr@rsidDel (§2.6.17); sectPr@rsidDel (§2.6.18); sectPr@rsidDel (§2.6.19); sectPr@rsidR (§2.6.17); sectPr@rsidR (§2.6.18); sectPr@rsidR (§2.6.19); sectPr@rsidRPr (§2.6.17); sectPr@rsidRPr (§2.6.18); sectPr@rsidRPr (§2.6.19); sectPr@rsidSect (§2.6.17); sectPr@rsidSect (§2.6.18); sectPr@rsidSect (§2.6.19); sig@csb0 (§2.8.2.16); sig@csb1 (§2.8.2.16); sig@usb0 (§2.8.2.16); sig@usb1 (§2.8.2.16); sig@usb2 (§2.8.2.16); sig@usb3 (§2.8.2.16); tmpl@val (§2.9.31); tr@rsidDel (§2.4.75); tr@rsidR (§2.4.75); tr@rsidRPr (§2.4.75); tr@rsidTr (§2.4.75); writeProtection@algIdExt (§2.15.1.94); writeProtection@cryptProviderTypeExt (§2.15.1.94)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_LongHexNumber">
   <restriction base="xsd:hexBinary">
      <length value="4"/>
   </restriction>
</simpleType>
```

2.18.58 ST_MacroName (Script Subroutine Name Value)

This simple type specifies a subroutine in a scripting language which may be executed based on the context of the parent XML element. The language and location of this subroutine may be determined using any method desired by an application.

[*Example*: Consider the following WordprocessingML fragment for the properties of a form field:

```
<w:ffData>
  <w:exitMacro w:val="TestExitFunction" />
</w:ffData>
```

The exitMacro element specifies that any application which processes this file should attempt to locate and execute a scripting subroutine called `TestExitFunction` when the contents of the form field are exited. If this subroutine cannot be located or executed, then this setting is silently ignored. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

This simple type also specifies the following restrictions:

- This simple type's contents have a maximum length of 33 characters.

Referenced By
entryMacro@val (§2.16.15); exitMacro@val (§2.16.16)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_MacroName">
   <restriction base="xsd:string">
      <maxLength value="33"/>
   </restriction>
</simpleType>
```

2.18.59 ST_MailMergeDataType (Mail Merge Data Source Type Values)

This simple type specifies the possible values for the types of external data sources to be connected to via the Dynamic Data Exchange (DDE) system (such as a spreadsheet or database), or the alternative method of data access if the Dynamic Data Exchange system is not used. This setting is purely a suggestion of the data source access mechanism which shall be used, and may be ignored in favor of an alternative mechanism if one is present.

[*Example*: Consider the following WordprocessingML fragment for a mail merge source or merged document:

```
<w:dataType w:val="odbc" />
```

The dataType element's val attribute is equal to odbc, specifying that the given merged WordprocessingML document has been connected to an external data source via the Open Database Connectivity interface. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
database (Database Data Source)	Specifies that a given merged WordprocessingML document has been connected to a database via the Dynamic Data Exchange (DDE) system.
native (Office Data Source Object Data Source)	Specifies that a given merged WordprocessingML document has been connected to an external data source via the Office Data Source Object (ODSO) interface.
odbc (Open Database Connectivity Data Source)	Specifies that a given merged WordprocessingML document has been connected to an external data source via the Open Database Connectivity interface.
query (Query Data Source)	Specifies that a given merged WordprocessingML document has been connected to an external data source using an external query tool.

Enumeration Value	Description
spreadsheet (Spreadsheet Data Source)	Specifies that a given merged WordprocessingML document has been connected to a database via the Dynamic Data Exchange (DDE) system.
textFile (Text File Data Source)	Specifies that a given merged WordprocessingML document has been connected to a text file via the Dynamic Data Exchange (DDE) system.

Referenced By
dataType@val (§2.14.10)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_MailMergeDataType">
   <restriction base="xsd:string">
      <enumeration value="textFile"/>
      <enumeration value="database"/>
      <enumeration value="spreadsheet"/>
      <enumeration value="query"/>
      <enumeration value="odbc"/>
      <enumeration value="native"/>
   </restriction>
</simpleType>
```

2.18.60 ST_MailMergeDest (Merged Document Destination Types)

This simple type specifies the possible results which may be generated when a mail merge is carried out on a given WordprocessingML source document. In other words, this element is used to specify what is to be done with the merged documents that result from populating the fields within a given merged WordprocessingML document with data from the specified external data source.

[*Example*: Consider a WordprocessingML source document containing the following WordprocessingML:

```
<w:mailMerge>
  <w:destination w:val="newDocument" />
  …
</w:mailMerge>
```

The destination element's val attribute is set to `newDocument`, specifying that when the mail merge is carried out, the source document shall be used to generate a specified number of new documents, which may be handled as appropriate. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description

Enumeration Value	Description
email (Send Merged Documents as E-mail Messages)	Specifies that conforming hosting applications shall generate emails using the documents that result from populating the fields within a given merged WordprocessingML document with data from the specified external data source.
fax (Send Merged Documents as Faxes)	Specifies that conforming hosting applications shall generate faxes using the documents that result from populating the fields within a given merged WordprocessingML document with data from the specified external data source.
newDocument (Send Merged Documents to New Documents)	Specifies that conforming hosting applications shall generate new documents by populating the fields within a given merged WordprocessingML document with data from the specified external data source.
printer (Send Merged Documents to Printer)	Specifies that conforming hosting applications shall print the documents that result from populating the fields within a given merged WordprocessingML document with external data from the specified external data source.

Referenced By
destination@val (§2.14.11)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_MailMergeDest">
   <restriction base="xsd:string">
      <enumeration value="newDocument"/>
      <enumeration value="printer"/>
      <enumeration value="email"/>
      <enumeration value="fax"/>
   </restriction>
</simpleType>
```

2.18.61 ST_MailMergeDocType (Source Document Types)

This simple types specifies the possible types for a given WordprocessingML source document.

[*Example*: Consider the WordprocessingML below:

```
<w:mailMerge>
  <w:mainDocumentType w:val="formLetters" />
  …
</w:mailMerge>
```

In this example, the source document is of the `formLetters` type, as specified by the mainDocumentType element's val attribute being equal to `formLetters`. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
catalog (Catalog Source Document)	Specifies that the mail merge source document is of the catalog type.
email (E-Mail Source Document)	Specifies that the mail merge source document is of the e-mail message type.
envelopes (Envelope Source Document)	Specifies that the mail merge source document is of the envelope type.
fax (Fax Source Document)	Specifies that the mail merge source document is of the fax type.
formLetters (Form Letter Source Document)	Specifies that the mail merge source document is of the form letter type.
mailingLabels (Mailing Label Source Document)	Specifies that the mail merge source document is of the mailing label type.

Referenced By
mainDocumentType@val (§2.14.22)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_MailMergeDocType">
   <restriction base="xsd:string">
      <enumeration value="catalog"/>
      <enumeration value="envelopes"/>
      <enumeration value="mailingLabels"/>
      <enumeration value="formLetters"/>
      <enumeration value="email"/>
      <enumeration value="fax"/>
   </restriction>
</simpleType>
```

2.18.62 ST_MailMergeOdsoFMDFieldType (Merge Field Mapping Types)

This simple types specifies the possible types used to indicate if a given mail merge field has been mapped to a column in the given external data source.

[*Example*: Consider the WordprocessingML below:

```
<w:odso>
  …
  <w:fieldMapData>
    <w:type w:val="dbColumn" />
    <w:name w:val="Country" />
    <w:mappedName w:val="Country or Region" />
    <w:column w:val="9" />
    …
  </w:fieldMapData>
</w:odso>
```

In this example, the `country` column within the given external data source shall be mapped to the mail merge field `Country or Region`, as specified by the type element's val attribute being equal to `dbColumn`. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
dbColumn (Field Mapping to Data Source Column)	Specifies that the mail merge field has been mapped to a column in the given external data source.
null (Field Not Mapped)	Specifies that the mail merge field has not been mapped to a column in the given external data source.

Referenced By
type@val (§2.14.32)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_MailMergeOdsoFMDFieldType">
   <restriction base="xsd:string">
      <enumeration value="null"/>
      <enumeration value="dbColumn"/>
   </restriction>
</simpleType>
```

2.18.63 ST_MailMergeSourceType (Mail Merge ODSO Data Source Types)

This simple type specifies the type of external data source to be connected to via as part of the ODSO connection information for this mail merge. This setting is purely a suggestion of the data source type which is being used for this mail merge, and may be ignored in favor of an alternative mechanism if one is present.

[*Example*: Consider the following WordprocessingML fragment for a mail merge source or merged document:

```
<w:type w:val="database" />
```

The type element's val attribute is equal to database, specifying that the given merged WordprocessingML document has been connected to an external data source via the ODSO settings, and that the resulting data source was a database. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
addressBook (Address Book Data Source)	Specifies that a given merged WordprocesingML document has been connected to an address book of contacts.
database (Database Data Source)	Specifies that a given merged WordprocessingML document has been connected to a database.
document1 (Alternate Document Format Data Source)	Specifies that a given merged WordprocessingML document has been connected to another document format supported by the producing application. The format of this document is application-defined and outside the scope of this Office Open XML Standard.
document2 (Alternate Document Format Data Source Two)	Specifies that a given merged WordprocessingML document has been connected to another document format supported by the producing application. The format of this document is application-defined and outside the scope of this Office Open XML Standard.
email (E-Mail Program Data Source)	Specifies that a given merged WordprocessingML document has been connected to an e-mail application.
legacy (Legacy Document Format Data Source)	Specifies that a given merged WordprocessingML document has been connected to a legacy document format supported by the producing application. The format of this legacy document is application-defined and outside the scope of this Office Open XML Standard.
master (Aggregate Data Source)	Specifies that a given merged WordprocessingML document has been connected to a data source which aggregates other data sources.
native (Native Data Souce)	Specifies that a given merged WordprocessingML document has been connected to another document format native to the producing application. The format of this document is application-defined and outside the scope of this Office Open XML Standard.
text (Text File Data Source)	Specifies that a given merged WordprocessingML document has been connected to a text file.

Referenced By
type@val (§2.14.33)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_MailMergeSourceType">
   <restriction base="xsd:string">
      <enumeration value="database"/>
      <enumeration value="addressBook"/>
      <enumeration value="document1"/>
      <enumeration value="document2"/>
      <enumeration value="text"/>
      <enumeration value="email"/>
      <enumeration value="native"/>
      <enumeration value="legacy"/>
      <enumeration value="master"/>
   </restriction>
</simpleType>
```

2.18.64 ST_Merge (Merged Cell Type)

This element specifies the way in which a cell shall be included in a merged group of cells (horizontally or vertically) within the parent table.

[*Example*: Consider a table with three rows and two columns with the last column completely vertically merged:

The second cell in the first row starts a vertical merge that is completed in the last cell, resulting in the following WordprocessingML:

```
<w:tbl>
  …
  <w:tr>
    <w:tc>
      …
    </w:tc>
    <w:tc>
      …
    </w:tc>
    <w:tc>
      <w:tcPr>
        <w:vmerge w:val="restart"/>
      </w:tcPr>
      …
    </w:tc>
  </w:tr>
  <w:tr>
    <w:tc>
      …
    </w:tc>
    <w:tc>
      …
    </w:tc>
    <w:tc>
      <w:tcPr>
        <w:vmerge w:val="continue"/>
      </w:tcPr>
      …
    </w:tc>
  </w:tr>
```

```
    <w:tr>
      <w:tc>
        …
      </w:tc>
      <w:tc>
        …
      </w:tc>
      <w:tc>
        <w:tcPr>
          <w:vmerge w:val="continue"/>
        </w:tcPr>
        …
      </w:tc>
    </w:tr>
  </w:tbl>
```

The val attribute of type ST_Merge on the vmerge element defines the cells which are vertically merged, and how each cell is merged together. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
continue (Continue Merged Region)	Specifies that the current cell continues a previously existing merged group of cells in the parent table. If the previous cell in the document (horizontally or vertically) does not either begin or continue a set of merged cells, then this value shall be ignored (i.e. a group of merged cells must start with a merge whose ST_Merge value is `restart`).
restart (Start/Restart Merged Region)	Specifies that the current cell starts (or restarts) a group of merged cells in the parent table. After this value, all following cells which have a value of `continue` shall be merged into this merged cell group.

Referenced By
hMerge@val (§2.4.16); vMerge@val (§2.4.81)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Merge">
   <restriction base="xsd:string">
      <enumeration value="continue"/>
      <enumeration value="restart"/>
   </restriction>
</simpleType>
```

2.18.65 ST_MultiLevelType (Numbering Definition Type)

This simple type specifies the possible types of numbering which may be defined by a given abstract numbering type. This information shall only be used by a consumer to determine user interface behaviors for this numbering definition, and shall not be used to limit the behavior of the list (i.e. a list with multiple levels marked as `singleLevel` shall not be prevented from using levels 2 through 9).

[*Example*: Consider the WordprocessingML below:

```
<w:abstractNum w:abstractNumId="8">
  …
  <w:multiLevelType w:val="singleLevel" />
  …
</w:abstractNum>
```

This abstract numbering definition is specified to be of the `singleLevel` type by the multiLevelType element. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
hybridMultilevel (Hybrid Multilevel Numbering Definition)	Specifies that this numbering definition defines a numbering format consisting of a multiple levels, each of a potentially different type (bullets vs. level text).
multilevel (Multilevel Numbering Definition)	Specifies that this numbering definition defines a numbering format consisting of a multiple levels, each of the same type (bullets vs. level text).
singleLevel (Single Level Numbering Definition)	Specifies that this numbering definition defines a numbering format consisting of a single level only.

Referenced By
multiLevelType@val (§2.9.13)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_MultiLevelType">
   <restriction base="xsd:string">
      <enumeration value="singleLevel"/>
      <enumeration value="multilevel"/>
      <enumeration value="hybridMultilevel"/>
   </restriction>
</simpleType>
```

2.18.66 ST_NumberFormat (Numbering Format)

This simple type specifies the numbering format which shall be used for a group of automatically numbered objects,

[*Example*: A value of `lowerLetter` for page numbering indicates that a consumer shall use lowercase letters for each page in this section: a,b,c… *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
aiueo (AIUEO Order Hiragana)	Specifies that the sequence shall consist of hiragana characters in the traditional a-i-u-e-o order. [*Example*: ア, イ, ウ. *end example*]
aiueoFullWidth (Full-Width AIUEO Order Hiragana)	Specifies that the sequence shall consist of full-width hiragana characters in the traditional a-i-u-e-o order. [*Example*: ア, イ, ウ *end example*]
arabicAbjad (Arabic Abjad Numerals)	Specifies that the sequence shall consist of ascending Abjad numerals. [*Example*: أ, ب, ج. *end example*]
arabicAlpha (Arabic Alphabet)	Specifies that the sequence shall consist of characters in the Arabic alphabet. [*Example*: أ, ب, ت. *end example*]
bullet (Bullet)	Specifies that the sequence shall consist of bullet characters. [*Example*: ●. *end example*]
cardinalText (Cardinal Text)	Specifies that the sequence shall consist of cardinal text of the run language. [*Example*: one, two, three. *end example*]

Enumeration Value	Description
chicago (Chicago Manual of Style)	Specifies that the sequence shall consist of characters as defined in the Chicago Manual of Style. [*Example*: *, †, ‡. *end example*]
chineseCounting (Chinese Counting System)	Specifies that the sequence shall consist of ascending numbers from the Chinese counting system. [*Example*: 一,二,三四*end example*]
chineseCountingThousand (Chinese Counting Thousand System)	Specifies that the sequence shall consist of sequential numbers from the Chinese counting thousand system. [*Example*: 一,二,…, 九一〇 *end example*]
chineseLegalSimplified (Chinese Legal Simplified Format)	Specifies that the sequence shall consist of sequential numbers from the Chinese simplified legal format. [*Example*: 壹…, 肆伍*end example*]
chosung (Korean Chosung Numbering)	Specifies that the sequence shall consist of sequential numbers from the Korean Chosung format. [*Example*: ㄱ , ㄴ , … *end example*]
decimal (Decimal Numbers)	Specifies that the sequence shall consist of decimal numbering. [*Example*: 1, 2, 3, … , 9, 10, 11. *end example*]
decimalEnclosedCircle (Decimal Numbers Enclosed in a Circle)	Specifies that the sequence shall consist of decimal numbering enclosed in a circle, using the enclosed alphanumeric glyph character. Once the specified sequence reaches 21, the numbers may be replaced with non-enclosed equivalents. [*Example*: ①②③.. *end example*]
decimalEnclosedCircleChinese (Decimal Numbers Enclosed in a Circle)	Specifies that the sequence shall consist of decimal numbering enclosed in a circle, using the enclosed alphanumeric glyph character. Once the specified sequence reaches 11, the numbers may be replaced with non-enclosed equivalents. [*Example*: ①②③.. *end example*]
decimalEnclosedFullstop (Decimal Numbers Followed by a Period)	Specifies that the sequence shall consist of decimal numbering followed by a period, using the enclosed alphanumeric glyph character.

Enumeration Value	Description
	Once the specified sequence reaches 21, the numbers may be replaced with non-enclosed equivalents. [*Example*: 1, 2, 3, … *end example*]
decimalEnclosedParen (Decimal Numbers Enclosed in Parenthesis)	Specifies that the sequence shall consist of decimal numbering enclosed in parenthesis, using the enclosed alphanumeric glyph character. Once the specified sequence reaches 21, the numbers may be replaced with non-enclosed equivalents. [*Example*: (1), (2), (3), … *end example*]
decimalFullWidth (Double Byte Arabic Numerals)	Specifies that the sequence shall consist of double-byte Arabic numbering. [*Example*: １, ２, ３. *end example*]
decimalFullWidth2 (Double Byte Arabic Numerals Alternate)	Specifies that the sequence shall consist of an alternative set of double-byte Arabic numbering, if one exists in the run font. [*Example*: １, ２, ３. *end example*]
decimalHalfWidth (Single Byte Arabic Numerals)	Specifies that the sequence shall consist of single-byte Arabic numbering. [*Example*: 1, 2, 3. *end example*]
decimalZero (Initial Zero Arabic Numerals)	Specifies that the sequence shall consist of Arabic numbering with a zero added to numbers one through nine. [*Example*: 01, 02, 03, …, 09, 10. *end example*]
ganada (Korean Ganada Numbering)	Specifies that the sequence shall consist of sequential numbers from the Korean Ganada format. [*Example*: 가, 나, … *end example*]
hebrew1 (Hebrew Numerals)	Specifies that the sequence shall consist of Hebrew numerals. [*Example*: א, ב, ג, … , י, יא *end example*]
hebrew2 (Hebrew Alphabet)	Specifies that the sequence shall consist of the Hebrew alphabet.

Enumeration Value	Description
	[*Example*: א, ב, ג, … *end example*]
hex (Hexadecimal Numbering)	Specifies that the sequence shall consist of hexadecimal numbering. [*Example*: 1, 2, 3, … , 9, A, B. *end example*]
hindiConsonants (Hindi Consonants)	Specifies that the sequence shall consist of Hindi consonants. [*Example*: अ, आ, इ, . *end example*]
hindiCounting (Hindi Counting System)	Specifies that the sequence shall consist of sequential numbers from the Hindi counting system. [*Example*: एक, दो, तीन, … *end example*]
hindiNumbers (Hindi Numbers)	Specifies that the sequence shall consist of Hindi numbers. [*Example*: १, २, ३, … *end example*]
hindiVowels (Hindi Vowels)	Specifies that the sequence shall consist of Hindi vowels. [*Example*: क, ख, ग, . *end example*]
ideographDigital (Ideographs)	Specifies that the sequence shall consist of sequential numerical ideographs enclosed in a circle, using the appropriate character. [*Example*: 一,二,三,四*end example*]
ideographEnclosedCircle (Ideographs Enclosed in a Circle)	Specifies that the sequence shall consist of sequential numerical ideographs enclosed in a circle, using the appropriate character. Once the specified sequence reaches 11, the numbers may be replaced with non-enclosed equivalents. [*Example*: ㊀, ㊁, ㊂,… *end example*]
ideographLegalTraditional (Traditional Legal Ideograph Format)	Specifies that the sequence shall consist of sequential numerical traditional legal ideographs. [*Example*: 壹貳參… *end example*]
ideographTraditional (Traditional Ideograph Format)	Specifies that the sequence shall consist of sequential numerical traditional ideographs.

Enumeration Value	Description
	[*Example*: 甲乙丙… *end example*]
ideographZodiac (Zodiac Ideograph Format)	Specifies that the sequence shall consist of sequential zodiac ideographs. [*Example*: 子丑寅… *end example*]
ideographZodiacTraditional (Traditional Zodiac Ideograph Format)	Specifies that the sequence shall consist of sequential traditional zodiac ideographs. [*Example*: 甲子乙丑丙寅… *end example*]
iroha (Iroha Ordered Katakana)	Specifies that the sequence shall consist of the iroha. [*Example*: イ, ロ, ハ, … *end example*]
irohaFullWidth (Full-Width Iroha Ordered Katakana)	Specifies that the sequence shall consist of the full-width forms of the iroha. [*Example*: イ, ロ ハ… *end example*]
japaneseCounting (Japanese Counting System)	Specifies that the sequence shall consist of sequential numbers from the Japanese counting system. [*Example*: 一, 二…, 九十十一 *end example*]
japaneseDigitalTenThousand (Japanese Digital Ten Thousand Counting System)	Specifies that the sequence shall consist of sequential numbers from the Japanese digital ten thousand counting system. [*Example*: 一, 二…, 九, 一〇 *end example*]
japaneseLegal (Japanese Legal Numbering)	Specifies that the sequence shall consist of sequential numbers from the Japanese legal counting system. [*Example*: 壱弐参…. *end example*]
koreanCounting (Korean Counting System)	Specifies that the sequence shall consist of sequential numbers from the Korean counting system. [*Example*: 일, 이, … *end example*]
koreanDigital (Korean Digital Counting System)	Specifies that the sequence shall consist of sequential numbers from the Korean digital counting system. [*Example*: 일, 이, … *end example*]
koreanDigital2 (Korean Digital Counting System Alternate)	Specifies that the sequence shall consist of sequential numbers from the Korean digital counting system. [*Example*: 일, 이, … *end example*]

Enumeration Value	Description
koreanLegal (Korean Legal Numbering)	Specifies that the sequence shall consist of sequential numbers from the Korean legal numbering system. [*Example*: 하나 , 둘 , … *end example*]
lowerLetter (Lowercase Latin Alphabet)	Specifies that the sequence shall consist of the letters of the Latin alphabet in lower case. [*Example*: a, b, c. *end example*]
lowerRoman (Lowercase Roman Numerals)	Specifies that the sequence shall consist of lowercase roman numerals. [*Example*: i, ii, iii. *end example*]
none (No Numbering)	Specifies that the sequence shall not display any numbering.
numberInDash (Number With Dashes)	Specifies that the sequence shall consist of the Arabic numbering surrounded by dash characters. [*Example*: - 1 -, - 2 -, - 3 -. *end example*]
ordinal (Ordinal)	Specifies that the sequence shall consist of ordinals of the run language. [*Example*: 1st, 2nd, 3rd. *end example*]
ordinalText (Ordinal Text)	Specifies that the sequence shall consist of ordinal text of the run language. [*Example*: first, second, third. *end example*]
russianLower (Lowercase Russian Alphabet)	Specifies that the sequence shall consist of the letters of the Russian alphabet in lower case. [*Example*: а, б, в. *end example*]
russianUpper (Uppercase Russian Alphabet)	Specifies that the sequence shall consist of the letters of the Russian alphabet in upper case. [*Example*: А, Б, В. *end example*]
taiwaneseCounting (Taiwanese Counting System)	Specifies that the sequence shall consist of sequential numbers from the Taiwanese counting system. [*Example*: 一,二…, 九十*end example*]
taiwaneseCountingThousand (Taiwanese Counting Thousand System)	Specifies that the sequence shall consist of sequential numbers from the Taiwanese counting thousand system.

Enumeration Value	Description
	[*Example*: 一, 二, …, 九, 〇. *end example*]
taiwaneseDigital (Taiwanese Digital Counting System)	Specifies that the sequence shall consist of sequential numbers from the Taiwanese digital counting system. [*Example*: 一, 二, …, 九, 〇. *end example*]
thaiCounting (Thai Counting System)	Specifies that the sequence shall consist of sequential numbers from the Thai counting system. [*Example*: หนึ่ง, สอง, สาม. *end example*]
thaiLetters (Thai Letters)	Specifies that the sequence shall consist of Thai letters. [*Example*: ก, ข, ค. *end example*]
thaiNumbers (Thai Numerals)	Specifies that the sequence shall consist of Thai numerals. [*Example*: ๒, ๓, ๔. *end example*]
upperLetter (Uppercase Latin Alphabet)	Specifies that the sequence shall consist of the letters of the Latin alphabet in upper case. [*Example*: A, B, C. *end example*]
upperRoman (Uppercase Roman Numerals)	Specifies that the sequence shall consist of uppercase roman numerals. [*Example*: I, II, III. *end example*]
vietnameseCounting (Vietnamese Numerals)	Specifies that the sequence shall consist of Vietnamese numerals. [*Example*: một, hai, ba. *end example*]

Referenced By
caption@numFmt (§2.15.1.16); numFmt@val (§2.9.18); numFmt@val (§2.11.17); numFmt@val (§2.11.18); pgNumType@fmt (§2.6.12)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_NumberFormat">
   <restriction base="xsd:string">
      <enumeration value="decimal"/>
      <enumeration value="upperRoman"/>
      <enumeration value="lowerRoman"/>
      <enumeration value="upperLetter"/>
      <enumeration value="lowerLetter"/>
      <enumeration value="ordinal"/>
      <enumeration value="cardinalText"/>
      <enumeration value="ordinalText"/>
      <enumeration value="hex"/>
      <enumeration value="chicago"/>
      <enumeration value="ideographDigital"/>
      <enumeration value="japaneseCounting"/>
      <enumeration value="aiueo"/>
      <enumeration value="iroha"/>
      <enumeration value="decimalFullWidth"/>
      <enumeration value="decimalHalfWidth"/>
      <enumeration value="japaneseLegal"/>
      <enumeration value="japaneseDigitalTenThousand"/>
      <enumeration value="decimalEnclosedCircle"/>
      <enumeration value="decimalFullWidth2"/>
      <enumeration value="aiueoFullWidth"/>
      <enumeration value="irohaFullWidth"/>
      <enumeration value="decimalZero"/>
      <enumeration value="bullet"/>
      <enumeration value="ganada"/>
      <enumeration value="chosung"/>
      <enumeration value="decimalEnclosedFullstop"/>
      <enumeration value="decimalEnclosedParen"/>
      <enumeration value="decimalEnclosedCircleChinese"/>
      <enumeration value="ideographEnclosedCircle"/>
      <enumeration value="ideographTraditional"/>
      <enumeration value="ideographZodiac"/>
      <enumeration value="ideographZodiacTraditional"/>
      <enumeration value="taiwaneseCounting"/>
      <enumeration value="ideographLegalTraditional"/>
      <enumeration value="taiwaneseCountingThousand"/>
      <enumeration value="taiwaneseDigital"/>
      <enumeration value="chineseCounting"/>
      <enumeration value="chineseLegalSimplified"/>
      <enumeration value="chineseCountingThousand"/>
      <enumeration value="koreanDigital"/>
      <enumeration value="koreanCounting"/>
      <enumeration value="koreanLegal"/>
      <enumeration value="koreanDigital2"/>
      <enumeration value="vietnameseCounting"/>
      <enumeration value="russianLower"/>
      <enumeration value="russianUpper"/>
      <enumeration value="none"/>
      <enumeration value="numberInDash"/>
      <enumeration value="hebrew1"/>
```

```
        <enumeration value="hebrew2"/>
        <enumeration value="arabicAlpha"/>
        <enumeration value="arabicAbjad"/>
        <enumeration value="hindiVowels"/>
        <enumeration value="hindiConsonants"/>
        <enumeration value="hindiNumbers"/>
        <enumeration value="hindiCounting"/>
        <enumeration value="thaiLetters"/>
        <enumeration value="thaiNumbers"/>
        <enumeration value="thaiCounting"/>
    </restriction>
</simpleType>
```

2.18.67 ST_OnOff (On/Off Value)

This simple type specifies a set of values for any binary (on or off) property defined in a WordprocessingML document.

A value of `on`, `1`, or `true` specifies that the property shall be turned on. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted.

A value of `off`, `0`, or `false` specifies that the property shall be explicitly turned off.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
0 (False)	Specifies that the binary state of this property is off (parent property is explicitly not applied).
1 (True)	Specifies that the binary state of this property is on (parent property is explicitly applied).
false (False)	Specifies that the binary state of this property is off (parent property is explicitly not applied).
off (False)	Specifies that the binary state of this property is off (parent property is explicitly not applied).
on (True)	Specifies that the binary state of this property is on (parent property is explicitly applied).
true (True)	Specifies that the binary state of this property is on (parent property is explicitly applied).

Referenced By
active@val (§2.14.1); activeWritingStyle@checkStyle (§2.15.1.1); activeWritingStyle@nlCheck (§2.15.1.1); adjustLineHeightInTable@val (§2.15.3.1); adjustRightInd@val (§2.3.1.1); alignBordersAndEdges@val (§2.15.1.2); alignTablesRowByRow@val (§2.15.3.2); allowPNG@val (§2.15.2.1);

Referenced By
allowSpaceOfSameStyleInTable@val (§2.15.3.3); alwaysMergeEmptyNamespace@val (§2.15.1.3); alwaysShowPlaceholderText@val (§2.15.1.4); applyBreakingRules@val (§2.15.3.4); autofitToFirstFixedWidthCell@val (§2.15.3.5); autoFormatOverride@val (§2.15.1.9); autoHyphenation@val (§2.15.1.10); autoRedefine@val (§2.7.3.2); autoSpaceDE@val (§2.3.1.2); autoSpaceDN@val (§2.3.1.3); autoSpaceLikeWord95@val (§2.15.3.6); b@val (§2.3.2.1); balanceSingleByteDoubleByteWidth@val (§2.15.3.7); bar@frame (§2.3.1.4); bar@shadow (§2.3.1.4); bCs@val (§2.3.2.2); bdr@frame (§2.3.2.3); bdr@shadow (§2.3.2.3); between@frame (§2.3.1.5); between@shadow (§2.3.1.5); bidi@val (§2.6.1); bidi@val (§2.3.1.6); bidiVisual@val (§2.4.1); blockQuote@val (§2.15.2.2); bodyDiv@val (§2.15.2.3); bookFoldPrinting@val (§2.15.1.11); bookFoldRevPrinting@val (§2.15.1.13); bordersDoNotSurroundFooter@val (§2.15.1.14); bordersDoNotSurroundHeader@val (§2.15.1.15); bottom@frame (§2.6.2); bottom@frame (§2.4.3); bottom@frame (§2.4.4); bottom@frame (§2.15.2.4); bottom@frame (§2.3.1.7); bottom@shadow (§2.6.2); bottom@shadow (§2.4.3); bottom@shadow (§2.4.4); bottom@shadow (§2.15.2.4); bottom@shadow (§2.3.1.7); cachedColBalance@val (§2.15.3.8); calcOnExit@val (§2.16.6); cantSplit@val (§2.4.6); caps@val (§2.3.2.4); caption@chapNum (§2.15.1.16); caption@noLabel (§2.15.1.16); checked@val (§2.16.8); cols@equalWidth (§2.6.4); cols@sep (§2.6.4); contextualSpacing@val (§2.3.1.9); convMailMergeEsc@val (§2.15.3.10); cs@val (§2.3.2.6); default@val (§2.16.12); dirty@val (§2.3.3.8); displayBackgroundShape@val (§2.15.1.25); displayHangulFixedWidth@val (§2.15.3.11); docPartUnique@val (§2.5.2.14); documentProtection@enforcement (§2.15.1.28); documentProtection@formatting (§2.15.1.28); doNotAutoCompressPictures@val (§2.15.1.32); doNotAutofitConstrainedTables@val (§2.15.3.12); doNotBreakConstrainedForcedTable@val (§2.15.3.13); doNotBreakWrappedTables@val (§2.15.3.14); doNotDemarcateInvalidXml@val (§2.15.1.33); doNotDisplayPageBoundaries@val (§2.15.1.34); doNotEmbedSmartTags@val (§2.15.1.35); doNotExpandShiftReturn@val (§2.15.3.15); doNotHyphenateCaps@val (§2.15.1.36); doNotIncludeSubdocsInStats@val (§2.15.1.37); doNotLeaveBackslashAlone@val (§2.15.3.16); doNotOrganizeInFolder@val (§2.15.2.10); doNotRelyOnCSS@val (§2.15.2.11); doNotSaveAsSingleFile@val (§2.15.2.12); doNotShadeFormData@val (§2.15.1.38); doNotSnapToGridInCell@val (§2.15.3.17); doNotSuppressBlankLines@val (§2.14.12); doNotSuppressIndentation@val (§2.15.3.18); doNotSuppressParagraphBorders@val (§2.15.3.19); doNotTrackFormatting@val (§2.15.1.39); doNotTrackMoves@val (§2.15.1.40); doNotUseEastAsianBreakRules@val (§2.15.3.20); doNotUseHTMLParagraphAutoSpacing@val (§2.15.3.21); doNotUseIndentAsNumberingTabStop@val (§2.15.3.22); doNotUseLongFileNames@val (§2.15.2.13); doNotUseMarginsForDrawingGridOrigin@val (§2.15.1.41); doNotValidateAgainstSchema@val (§2.15.1.42); doNotVertAlignCellWithSp@val (§2.15.3.23); doNotVertAlignInTxbx@val (§2.15.3.24); doNotWrapTextWithPunct@val (§2.15.3.25); dstrike@val (§2.3.2.7); dynamicAddress@val (§2.14.13); eastAsianLayout@combine (§2.3.2.8); eastAsianLayout@vert (§2.3.2.8); eastAsianLayout@vertCompress (§2.3.2.8); embedBold@subsetted (§2.8.2.3); embedBoldItalic@subsetted (§2.8.2.4); embedItalic@subsetted (§2.8.2.5); embedRegular@subsetted (§2.8.2.6); embedSystemFonts@val (§2.8.2.7); embedTrueTypeFonts@val (§2.8.2.8); emboss@val (§2.3.2.11); enabled@val (§2.16.14); endnoteReference@customMarkFollows (§2.11.7); evenAndOddHeaders@val (§2.10.1); fHdr@val (§2.14.14); flatBorders@val (§2.15.2.15); fldChar@dirty (§2.16.18); fldChar@fldLock (§2.16.18); fldSimple@dirty (§2.16.21); fldSimple@fldLock (§2.16.21); footnoteLayoutLikeWW8@val (§2.15.3.26); footnoteReference@customMarkFollows (§2.11.14); forgetLastTabAlignment@val (§2.15.3.27); formProt@val (§2.6.6); formsDesign@val (§2.15.1.48); framePr@anchorLock (§2.3.1.11); growAutofit@val (§2.15.3.28); gutterAtTop@val (§2.15.1.49); hidden@val (§2.7.3.4); hidden@val (§2.4.14); hideGrammaticalErrors@val (§2.15.1.51); hideMark@val (§2.4.15); hideSpellingErrors@val (§2.15.1.52); hyperlink@history (§2.16.24); i@val (§2.3.2.14); iCs@val (§2.3.2.15); ignoreMixedContent@val (§2.15.1.54); imprint@val (§2.3.2.16); insideH@frame (§2.4.17); insideH@frame

Referenced By
(§2.4.18); insideH@shadow (§2.4.17); insideH@shadow (§2.4.18); insideV@frame (§2.4.19); insideV@frame (§2.4.20); insideV@shadow (§2.4.19); insideV@shadow (§2.4.20); isLgl@val (§2.9.4); keepLines@val (§2.3.1.14); keepNext@val (§2.3.1.15); kinsoku@val (§2.3.1.16); latentStyles@defLockedState (§2.7.3.5); latentStyles@defQFormat (§2.7.3.5); latentStyles@defSemiHidden (§2.7.3.5); latentStyles@defUnhideWhenUsed (§2.7.3.5); layoutRawTableWidth@val (§2.15.3.29); layoutTableRowsApart@val (§2.15.3.30); left@frame (§2.15.2.21); left@frame (§2.4.24); left@frame (§2.6.7); left@frame (§2.3.1.17); left@frame (§2.4.27); left@shadow (§2.15.2.21); left@shadow (§2.4.24); left@shadow (§2.6.7); left@shadow (§2.3.1.17); left@shadow (§2.4.27); legacy@legacy (§2.9.5); lineWrapLikeWord6@val (§2.15.3.31); linkedToFile@val (§2.15.2.22); linkStyles@val (§2.15.1.55); linkToQuery@val (§2.14.18); locked@val (§2.7.3.7); lsdException@locked (§2.7.3.8); lsdException@qFormat (§2.7.3.8); lsdException@semiHidden (§2.7.3.8); lsdException@unhideWhenUsed (§2.7.3.8); lvl@tentative (§2.9.6); lvl@tentative (§2.9.7); lvlText@null (§2.9.12); mailAsAttachment@val (§2.14.19); matchSrc@val (§2.17.3.3); mirrorIndents@val (§2.3.1.18); mirrorMargins@val (§2.15.1.57); mwSmallCaps@val (§2.15.3.32); name@decorated (§2.12.13); noBorder@val (§2.15.2.30); noColumnBalance@val (§2.15.3.33); noEndnote@val (§2.11.16); noExtraLineSpacing@val (§2.15.3.34); noLeading@val (§2.15.3.35); noProof@val (§2.3.2.19); noPunctuationKerning@val (§2.15.1.60); noResizeAllowed@val (§2.15.2.31); noSpaceRaiseLower@val (§2.15.3.36); noTabHangInd@val (§2.15.3.37); notTrueType@val (§2.8.2.12); noWrap@val (§2.4.28); oMath@val (§2.3.2.20); optimizeForBrowser@val (§2.15.2.32); outline@val (§2.3.2.21); overflowPunct@val (§2.3.1.21); pageBreakBefore@val (§2.3.1.23); personal@val (§2.7.3.11); personalCompose@val (§2.7.3.12); personalReply@val (§2.7.3.13); printBodyTextBeforeHeader@val (§2.15.3.38); printColBlack@val (§2.15.3.39); printFormsData@val (§2.15.1.61); printFractionalCharacterWidth@val (§2.15.1.62); printPostScriptOverText@val (§2.15.1.63); printTwoOnOne@val (§2.15.1.64); qFormat@val (§2.7.3.14); readModeInkLockDown@actualPg (§2.15.1.66); relyOnVML@val (§2.15.2.34); removeDateAndTime@val (§2.15.1.67); removePersonalInformation@val (§2.15.1.68); revisionView@comments (§2.15.1.69); revisionView@formatting (§2.15.1.69); revisionView@inkAnnotations (§2.15.1.69); revisionView@insDel (§2.15.1.69); revisionView@markup (§2.15.1.69); right@frame (§2.3.1.28); right@frame (§2.4.30); right@frame (§2.6.15); right@frame (§2.15.2.35); right@frame (§2.4.32); right@shadow (§2.3.1.28); right@shadow (§2.4.30); right@shadow (§2.6.15); right@shadow (§2.15.2.35); right@shadow (§2.4.32); rtl@val (§2.3.2.28); rtlGutter@val (§2.6.16); saveFormsData@val (§2.15.1.73); saveInvalidXml@val (§2.15.1.74); savePreviewPicture@val (§2.15.1.75); saveSmartTagsAsXml@val (§2.15.2.36); saveSubsetFonts@val (§2.8.2.15); saveXmlDataOnly@val (§2.15.1.77); selectFldWithFirstOrLastChar@val (§2.15.3.40); semiHidden@val (§2.7.3.16); shadow@val (§2.3.2.29); shapeLayoutLikeWW8@val (§2.15.3.41); showBreaksInFrames@val (§2.15.3.42); showEnvelope@val (§2.15.1.80); showingPlcHdr@val (§2.5.2.38); showXMLTags@val (§2.15.1.81); sizeAuto@val (§2.16.31); smallCaps@val (§2.3.2.31); snapToGrid@val (§2.3.2.32); snapToGrid@val (§2.3.1.32); spaceForUL@val (§2.15.3.43); spacing@afterAutospacing (§2.3.1.33); spacing@beforeAutospacing (§2.3.1.33); spacingInWholePoints@val (§2.15.3.44); specVanish@val (§2.3.2.34); splitPgBreakAndParaMark@val (§2.15.3.45); strictFirstAndLastChars@val (§2.15.1.83); strike@val (§2.3.2.35); style@customStyle (§2.7.3.17); style@default (§2.7.3.17); styleLockQFSet@val (§2.15.1.84); styleLockTheme@val (§2.15.1.85); subFontBySize@val (§2.15.3.46); suppressAutoHyphens@val (§2.3.1.34); suppressBottomSpacing@val (§2.15.3.47); suppressLineNumbers@val (§2.3.1.35); suppressOverlap@val (§2.3.1.36); suppressSpacingAtTopOfPage@val (§2.15.3.48); suppressSpBfAfterPgBrk@val (§2.15.3.49); suppressTopSpacing@val (§2.15.3.50); suppressTopSpacingWP@val (§2.15.3.51); swapBordersFacingPages@val (§2.15.3.52); tblHeader@val (§2.4.46); tcFitText@val (§2.4.64); temporary@val (§2.5.2.41); text@multiLine (§2.5.2.42); titlePg@val (§2.10.6); tl2br@frame (§2.4.70);

Referenced By
tl2br@shadow (§2.4.70); top@frame (§2.4.71); top@frame (§2.3.1.42); top@frame (§2.15.2.42); top@frame (§2.4.74); top@frame (§2.6.21); top@shadow (§2.4.71); top@shadow (§2.3.1.42); top@shadow (§2.15.2.42); top@shadow (§2.4.74); top@shadow (§2.6.21); topLinePunct@val (§2.3.1.43); tr2bl@frame (§2.4.76); tr2bl@shadow (§2.4.76); trackRevisions@val (§2.15.1.90); truncateFontHeightsLikeWP6@val (§2.15.3.53); types@all (§2.12.16); uiCompat97To2003@val (§2.15.3.54); ulTrailSpace@val (§2.15.3.55); underlineTabInNumList@val (§2.15.3.56); unhideWhenUsed@val (§2.7.3.20); updateFields@val (§2.15.1.91); useAltKinsokuLineBreakRules@val (§2.15.3.57); useAnsiKerningPairs@val (§2.15.3.58); useFELayout@val (§2.15.3.59); useNormalStyleForList@val (§2.15.3.60); usePrinterMetrics@val (§2.15.3.61); useSingleBorderforContiguousCells@val (§2.15.3.62); useWord2002TableStyleRules@val (§2.15.3.63); useWord97LineBreakRules@val (§2.15.3.64); useXSLTWhenSaving@val (§2.15.1.92); vanish@val (§2.3.2.39); viewMergedData@val (§2.14.36); webHidden@val (§2.3.2.42); widowControl@val (§2.3.1.44); wordWrap@val (§2.3.1.45); wpJustification@val (§2.15.3.65); wpSpaceWidth@val (§2.15.3.66); wrapTrailSpaces@val (§2.15.3.67); writeProtection@recommended (§2.15.1.94)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_OnOff">
   <restriction base="xsd:string">
      <enumeration value="true"/>
      <enumeration value="false"/>
      <enumeration value="on"/>
      <enumeration value="off"/>
      <enumeration value="0"/>
      <enumeration value="1"/>
   </restriction>
</simpleType>
```

2.18.68 ST_PageBorderDisplay (Page Border Display Options)

This simple type specifies the pages in the parent section on which the page border shall be printed.

[*Example*: Consider a section in a document for which the page border shall only be printed on the first page. This setting is specified using the following WordprocessingML:

```
<w:pgBorders w:display="firstPage">
  …
</w:pgBorders>
```

The display attribute with a value of `firstPage` specifies that only the first page shall display the page border defined for this section. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
allPages (Display Page Border on All Pages)	Specifies that the page border shall be displayed on all pages in the parent section.

Enumeration Value	Description
firstPage (Display Page Border on First Page)	Specifies that the page border shall be displayed on only the first page in the parent section.
notFirstPage (Display Page Border on All Pages Except First)	Specifies that the page border shall be displayed on only the first page in the parent section.

Referenced By
pgBorders@display (§2.6.10)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PageBorderDisplay">
   <restriction base="xsd:string">
      <enumeration value="allPages"/>
      <enumeration value="firstPage"/>
      <enumeration value="notFirstPage"/>
   </restriction>
</simpleType>
```

2.18.69 ST_PageBorderOffset (Page Border Positioning Base)

This simple type specifies how the relative positioning of the page borders shall be calculated.

If the value of this attribute is `text`, then the space attribute on each page border shall be interpreted as the distance from the text margins that shall be left before the page border.

[*Example*: Consider the following WordprocessingML fragment:

```
<w:pgBorders w:offsetFrom="page">
  <w:top w:val="dashed" w:space="24" />
  <w:left w:val="dashed" w:space="24" />
  <w:bottom w:val="dashed" w:space="24"/>
  <w:right w:val="dashed" w:space="24"/>
</w:pgBorders>
```

This fragment specifies that the page borders shall be indented 24 points from the page extents.

This is distinct from the following fragment with identical space attribute values:

```
<w:pgBorders w:offsetFrom="text">
  <w:top w:val="dashed" w:space="24" />
  <w:left w:val="dashed" w:space="24" />
  <w:bottom w:val="dashed" w:space="24"/>
  <w:right w:val="dashed" w:space="24"/>
</w:pgBorders>
```

In this case, the page borders will be offset by 24 points, but in this case, that offset will be calculated relative to the text margins. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
page (Page Border Is Positioned Relative to Page Edges)	Specifies that the space attribute on each page border shall be interpreted as the distance from the edge of the page that shall be left before the page border.
text (Page Border Is Positioned Relative to Text Extents)	Specifies that the space attribute on each page border shall be interpreted as the distance from the edge of the text extents (text margins) that shall be left before the page border..

Referenced By
pgBorders@offsetFrom (§2.6.10)

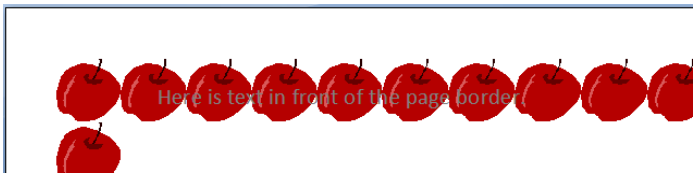
The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PageBorderOffset">
   <restriction base="xsd:string">
      <enumeration value="page"/>
      <enumeration value="text"/>
   </restriction>
</simpleType>
```

2.18.70 ST_PageBorderZOrder (Page Border Z-Order)

This simple type specifies whether the page border is positioned above or below intersecting texts and objects in this document.

[*Example*: Consider a document in which the page border shall be displayed below any intersecting text as follows:



This setting is specified by setting the value of an attribute with to back, which specifies that the page border shall be displayed behind all intersecting text and objects. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
back (Page Border Behind Text)	Specifies that the page border shall be rendered beneath any text or object which intersects it - effectively placing it at the lowest z-order on the page.
front (Page Border Ahead of Text)	Specifies that the page border shall be rendered above any text or object which intersects it - effectively placing it at the highest z-order on the page.

Referenced By
pgBorders@zOrder (§2.6.10)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PageBorderZOrder">
   <restriction base="xsd:string">
      <enumeration value="front"/>
      <enumeration value="back"/>
   </restriction>
</simpleType>
```

2.18.71 ST_PageOrientation (Page Orientation)

This simple type specifies the orientation of all pages in the parent section. This information is used to determine the actual paper size to use when printing the file.

[*Example*: Pages 11" wide by 8.5" long in landscape mode use 8.5"x11" paper, because the width and height are reversed for pages in this landscape section with respect to the printed page. *end example*]

[*Example*: Consider the following WordprocessingML:

```
<w:pgSz w:w="15840" w:h="12240" w:orient="landscape" />
```

Although the page width is 11", and page height is 8.5", according to the w and h attributes, because the orient attribute is set to `landscape`, pages in this section are printed on 8.5x11" paper in landscape mode. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
landscape (Landscape Mode)	Specifies that pages in this section shall be printed in landscape mode, which prints the page contents with a 90 degree rotation with respect to the normal page orientation.
portrait (Portrait Mode)	Specifies that pages in this section shall be printed in

Enumeration Value	Description
	portrait mode.

Referenced By
pgSz@orient (§2.6.13)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PageOrientation">
   <restriction base="xsd:string">
      <enumeration value="portrait"/>
      <enumeration value="landscape"/>
   </restriction>
</simpleType>
```

2.18.72 ST_Panose (Panose-1 Number)

This simple type specifies a number consisting of 10 hexadecimal digits which defines the Panose-1 classification number a font.

[*Example*: Consider the following information stored for a single font:

```
<w:font w:name="Times New Roman">
  <w:panose1 w:val="02020603050405020304" />
  …
</w:font>
```

The panose1 element specifies its Panose-1 number via its val attribute value of `02020603050405020304`. *end example*]

This simple type's contents are a restriction of the XML Schema hexBinary datatype.

This simple type also specifies the following restrictions:

- This simple type's contents must have a length of exactly 10 characters.

Referenced By
panose1@val (§2.8.2.13)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Panose">
   <restriction base="xsd:hexBinary">
      <length value="10"/>
   </restriction>
</simpleType>
```

2.18.73 ST_Pitch (Font Pitch Value)

This simple type specifies the possible values for the font pitch of a font.

[*Example*: Consider the following information stored for a single font:

```
<w:font w:name="Courier New">
  <w:pitch w:val="fixed" />
  …
</w:font>
```

The pitch element specifies via its val attribute value of `fixed` that this is a fixed width font. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
default (Default)	Specifies that no information is available about the pitch of a font.
fixed (Fixed Width)	Specifies that this is a fixed width font.
variable (Proportional Width)	Specifies that this is a proportional width font.

Referenced By
pitch@val (§2.8.2.14)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Pitch">
   <restriction base="xsd:string">
      <enumeration value="fixed"/>
      <enumeration value="variable"/>
      <enumeration value="default"/>
   </restriction>
</simpleType>
```

2.18.74 ST_PixelsMeasure (Measurement in Pixels)

This simple type specifies that its contents will contain a positive whole number, whose contents consist of a measurement in pixels.

The contents of this measurement are interpreted based on the context of the parent XML element.

[*Example*: Consider an attribute value of 96 whose type is ST_PixelsMeasure. This attribute value specifies a size of 96 pixels (one inch on a 96 pixels per inch display). *end example*]

This simple type's contents are a restriction of the ST_UnsignedDecimalNumber simple type (§2.18.108).

Referenced By
marH@val (§2.15.2.24); marW@val (§2.15.2.28); readModeInkLockDown@h (§2.15.1.66); readModeInkLockDown@w (§2.15.1.66)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PixelsMeasure">
   <restriction base="ST_UnsignedDecimalNumber"/>
</simpleType>
```

2.18.75 ST_PointMeasure (Measurement in Points)

This simple type specifies that its contents will contain a positive whole number, whose contents consist of a measurement in points (equivalent to 1/72nd of an inch).

The contents of this measurement are interpreted based on the context of the parent XML element.

[*Example*: Consider an attribute value of 24 whose type is ST_PointMeasure. This attribute value specifies a size in points (24 points = 1/3 of an inch). *end example*]

This simple type's contents are a restriction of the ST_UnsignedDecimalNumber simple type (§2.18.108).

Referenced By
bar@space (§2.3.1.4); bdr@space (§2.3.2.3); between@space (§2.3.1.5); bottom@space (§2.6.2); bottom@space (§2.4.3); bottom@space (§2.4.4); bottom@space (§2.15.2.4); bottom@space (§2.3.1.7); insideH@space (§2.4.17); insideH@space (§2.4.18); insideV@space (§2.4.19); insideV@space (§2.4.20); left@space (§2.15.2.21); left@space (§2.4.24); left@space (§2.6.7); left@space (§2.3.1.17); left@space (§2.4.27); right@space (§2.3.1.28); right@space (§2.4.30); right@space (§2.6.15); right@space (§2.15.2.35); right@space (§2.4.32); tl2br@space (§2.4.70); top@space (§2.4.71); top@space (§2.3.1.42); top@space (§2.15.2.42); top@space (§2.4.74); top@space (§2.6.21); tr2bl@space (§2.4.76)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PointMeasure">
   <restriction base="ST_UnsignedDecimalNumber"/>
</simpleType>
```

2.18.76 ST_Proof (Proofing State Values)

This simple type specifies the values which may be used to indicate the status of a given hosting application's grammar and spell checking when a given WordprocessingML document was last saved.

[*Example*: Consider a WordprocessingML document that is saved by a hosting application whose grammar checking engine had completed checking the grammar in the given WordprocessingML document, but whose spell checking engine had not completed checking the spelling in the given WordprocessingML document . In this instance, the following WordprocessingML shall be written in the document settings:

```
<w:proofState w:spelling="dirty" w:grammar="clean" />
```

The proofState element's attributes spelling and grammar have the ST_Proof simple type enumeration values `dirty` and `clean` respectively, specifying that the hosting application's spell checking engine had not completed checking the spelling of the document, and that the hosting application's grammar checking engine had completed checking the grammar of the document, when the WordprocessingML document was last saved. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
clean (Check Completed)	Specifies that the given proofing engine completed checking the document when it was last saved.
dirty (Check Not Completed)	Specifies that the given proofing engine did not complete checking the document when it was last saved.

Referenced By
proofState@grammar (§2.15.1.65); proofState@spelling (§2.15.1.65)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Proof">
   <restriction base="xsd:string">
      <enumeration value="clean"/>
      <enumeration value="dirty"/>
   </restriction>
</simpleType>
```

2.18.77 ST_ProofErr (Proofing Error Type)

This simple type specifies the possible values for the types of proofing error markers which can appear in the contents of a WordprocessingML document to indicate the last known state of any spell- and grammar-checking performed on the contents of this document.

[*Example*: Consider the following paragraph consisting of two misspelled words, where the second word has been explicitly flagged as not being a spelling error. This paragraph would consist of the following WordprocessingML markup:

```
<w:p>
  <w:proofErr w:val="spellStart"/>
  <w:r>
    <w:t>erqwt</w:t>
  </w:r>
  <w:proofErr w:val="spellEnd"/>
  <w:r>
    <w:t xml:space="preserve"> werewr</w:t>
  </w:r>
</w:p>
```

The proofErr elements with a val attribute value of `spellStart` and `spellEnd`, respectively delimit the start and end the content in this paragraph which is stored as a spelling error. Since the second word is not included in that range, it is not stored as a spelling error. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
gramEnd (End of Region Marked as Grammatical Error)	Specifies that this proofing error marker shall indicate the start of a region to be marked as a grammatical error in the document.
gramStart (Start of Region Marked as Grammatical Error)	Specifies that this proofing error marker shall indicate the end of a region to be marked as a grammatical error in the document.
spellEnd (End of Region Marked as Spelling Error)	Specifies that this proofing error marker shall indicate the end of a region to be marked as a spelling error in the document.
spellStart (Start of Region Marked as Spelling Error)	Specifies that this proofing error marker shall indicate the start of a region to be marked as a spelling error in the document.

Referenced By
proofErr@type (§2.13.8.1)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ProofErr">
   <restriction base="xsd:string">
      <enumeration value="spellStart"/>
      <enumeration value="spellEnd"/>
      <enumeration value="gramStart"/>
      <enumeration value="gramEnd"/>
   </restriction>
</simpleType>
```

2.18.78 ST_PTabAlignment (Absolute Position Tab Alignment)

This simple type specifies the alignment of an absolutely positioned tab character in a document. This alignment value determines the position on the line to which this absolute tab shall advance, as well as the alignment of the text entered after the alignment tab character position.

[*Example*: Consider a positional tab stop in a WordprocessingML document who shall move to the left edge of the text margins and whose subsequent text should be left aligned. This positional tab stop would be defined as follows:

```
<w:ptab w:alignment="left" w:relativeTo=”margin” … />
```

The alignment attribute has a value of `left`, which specifies that this custom tab stop shall align on the left edge of the line relative to the text margin. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
center (Center)	Specifies that the positional tab should be center aligned on the line relative to the specified base (the text margins with or without indents), and that the text at that location shall be center aligned.
left (Left)	Specifies that the positional tab should be left aligned on the line relative to the specified base (the text margins with or without indents), and that the text at that location shall be left aligned.
right (Right)	Specifies that the positional tab should be right aligned on the line relative to the specified base (the text margins with or without indents), and that the text at that location shall be right aligned.

Referenced By
ptab@alignment (§2.3.3.22)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PTabAlignment">
   <restriction base="xsd:string">
      <enumeration value="left"/>
      <enumeration value="center"/>
      <enumeration value="right"/>
   </restriction>
</simpleType>
```

2.18.79 ST_PTabLeader (Absolute Position Tab Leader Character)

This simple type specifies the characters which may be used to fill in the space created by a positional tab. This character shall be repeated as required to completely fill the tab spacing generated by the positional tab character.

[*Example*: Consider a positional tab stop which should be preceded by a sequence of underscore characters, as follows:

_____Text at the positional tab stop

This tab stop would have a leader attribute value of underscore, indicating that the tab stop shall be preceded by underscore characters as needed to fill the tab spacing. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
dot (Dot Leader Character)	Specifies that the leader character for this positional tab stop shall be a dot. [*Example*: ..... Text at absolute tab. *end example*]
hyphen (Hyphen Leader Character)	Specifies that the leader character for this positional tab stop shall be a hyphen. [*Example*: ---------- Text at absolute tab. *end example*]
middleDot (Centered Dot Leader Character)	Specifies that the leader character for this positional tab stop shall be a centered dot.

Enumeration Value	Description
	[*Example*: ····· Text at absolute tab. *end example*]
none (No Leader Character)	Specifies that there shall be no leader character for this positional tab. [*Example*: Text at absolute tab. *end example*]
underscore (Underscore Leader Character)	Specifies that the leader character for this positional tab stop shall be an underscore. [*Example*: _____ Text at absolute tab. *end example*]

Referenced By
ptab@leader (§2.3.3.22)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PTabLeader">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="dot"/>
      <enumeration value="hyphen"/>
      <enumeration value="underscore"/>
      <enumeration value="middleDot"/>
   </restriction>
</simpleType>
```

2.18.80 ST_PTabRelativeTo (Absolute Position Tab Positioning Base)

Specifies the possible extents which may be used to calculate the absolute positioning of this positional tab character.

[*Example*: Consider a positional tab stop in a WordprocessingML document that should have a resulting position that is centered on the text margins, ignoring both any custom tab stops and any text indents on the paragraph. This positional tab stop would be defined as follows:

```
<w:ptab w:relativeTo="margin" … />
```

The relativeTo attribute specifies that this absolute position tab stop shall be relative to the `margin`. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
indent (Relative To Indents)	Specifies that the absolute positioning of the tab shall be relative to the indents.
margin (Relative To Text Margins)	Specifies that the absolute positioning of the tab shall be relative to the margins.

Referenced By
ptab@relativeTo (§2.3.3.22)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PTabRelativeTo">
   <restriction base="xsd:string">
      <enumeration value="margin"/>
      <enumeration value="indent"/>
   </restriction>
</simpleType>
```

2.18.81 ST_RestartNumber (Footnote/Endnote Numbering Restart Locations)

This simple type specifies the possible values for when the automatic numbering of footnotes or endnotes shall be restarted.

[*Example*: Consider a WordprocessingML document where the numbering for its endnotes shall be restarted after each section shall be restarted after each page. This setting is represented by the following WordprocessingML:

```
<w:footnotePr>
  …
  <w:numRestart w:val="eachSect" />
  …
</w:footnotePr>
```

The val attribute value of `eachSect` specifies that numbering shall be restarted after each section. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
continuous (Continue Numbering From Previous Section)	Specifies that the numbering of footnotes or endnotes shall continue from the previous section in the document.
eachPage (Restart Numbering On Each Page)	Specifies that the numbering of footnotes or endnotes shall be restarted to its starting value for each unique page in the document.
eachSect (Restart Numbering For Each Section)	Specifies that the numbering of footnotes or endnotes shall be restarted to its starting value for each unique section in the document.

Referenced By
numRestart@val (§2.11.19)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_RestartNumber">
   <restriction base="xsd:string">
      <enumeration value="continuous"/>
      <enumeration value="eachSect"/>
      <enumeration value="eachPage"/>
   </restriction>
</simpleType>
```

2.18.82 ST_RubyAlign (Phonetic Guide Text Alignment)

This simple type specifies the possible alignment settings which may be used to determine the placement of phonetic guide text with respect to the base text when this phonetic guide is displayed.

[*Example*: Consider a run of phonetic guide text which shall have the ruby text positioned to the far left of the base text. This constraint is specified using the following WordprocessingML:

```
<w:rubyPr>
  …
  <w:rubyAlign w:val="left"/>
  …
</w:rubyPr>
```

The rubyAlign property is `left` for the phonetic guide, so the ruby text will be displayed on the left side of the base text. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description

Enumeration Value	Description
center (Center)	Specifies that the phonetic guide text shall be centered with respect to the base text in this document. [*Example*: guide text this is a test center *end example*]
distributeLetter (Distribute All Characters)	Specifies that the phonetic guide text shall be distributed with respect to the base text in this document. This type of justification shall equally affect the inter-word spacing on each line as well as the inter-character spacing between each word when justifying its contents - that is, an equal amount of additional character pitch shall be added to all characters on the line. [*Example*: g u i d e t e x t a test distribute letter *end example*]
distributeSpace (Distribute all Characters w/ Additional Space On Either Side)	Specifies that the phonetic guide text shall be distributed with respect to the base text in this document, with additional space added to the guide text to ensure it is indented with respect to the base text. This type of justification shall equally affect the inter-word spacing on each line as well as the inter-character spacing between each word when justifying its contents - that is, an equal amount of additional character pitch shall be added to all characters on the line. As well, an additional space is added before and after the guide text to ensure it is indented with respect to the base text. [*Example*: g u i d e t e x t a test distribute space *end example*]

Enumeration Value	Description
left (Left Aligned)	Specifies that the phonetic guide text shall be left aligned with respect to the base text in this document. [*Example*: guide text this is a test left *end example*]
right (Right Aligned)	Specifies that the phonetic guide text shall be right aligned with respect to the base text in this document. [*Example*: guide text this is a test right *end example*]
rightVertical (Vertically Aligned to Right of Base Text)	Specifies that the phonetic guide text shall be right aligned with respect to the base text in this document, and shall always be displayed vertically and to the right of the base text, regardless of the alignment of the base text. [*Example*: this is a test right vertical guide text *end example*]

Referenced By
rubyAlign@val (§2.3.3.25)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_RubyAlign">
   <restriction base="xsd:string">
      <enumeration value="center"/>
      <enumeration value="distributeLetter"/>
      <enumeration value="distributeSpace"/>
      <enumeration value="left"/>
      <enumeration value="right"/>
      <enumeration value="rightVertical"/>
   </restriction>
</simpleType>
```

2.18.83 ST_SdtDateMappingType (Date Storage Format Types)

This simple type specifies then possible types of translations which may be performed on the displayed date in a date picker structured document tag when the current contents are saved into the associated custom XML data via the dataBinding element (§2.5.2.6).

[*Example*: Consider the following date picker structured document tag:

```
<w:sdt>
  <w:sdtPr>
    <w:date w:fullDate="01-01-2006T06:30:00Z">
      <w:storeMappedDateAs w:val="text"/>
      …
    </w:date>
  </w:sdtPr>
  <w:sdtContent>
    <w:r>
      <w:t>January 1</w:t>
    </w:r>
  </w:sdtContent>
</w:sdt>
```

The value of the storeMappedDateAs element's attribute value is `text`, therefore the current run contents shall be sent to the mapped XML element without any translation (in this case, the value shall be `January 1`). *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
date (XML Schema Date Format)	Specifies that the date specified in the parent date picker structured document tag shall be converted to the xsd:date format when stored in a mapped XML element.
dateTime (XML Schema DateTime Format)	Specifies that the date specified in the parent date picker structured document tag shall be converted to the xsd:dateTime format when stored in a mapped XML element.
text (Same As Display)	Specifies that no translation shall be performed on the displayed date when stored in a mapped XML element - the mapped contents shall be the same as the displayed contents.

Referenced By
storeMappedDataAs@val (§2.5.2.39)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SdtDateMappingType">
   <restriction base="xsd:string">
      <enumeration value="text"/>
      <enumeration value="date"/>
      <enumeration value="dateTime"/>
   </restriction>
</simpleType>
```

2.18.84 ST_SectionMark (Section Type)

Specifies the type of the current section.

[*Example*: Consider a section that shall start on the next page in the document. The WordprocessingML specifying this would look like:

```
<w:sectPr>
  …
  <w:type w:val="nextPage"/>
</w:sectPr>
```

The nextPage value specifies that this section starts on the next page. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
continuous (Continuous Section Break)	Specifies a continuous section break, which begin the new section on the following paragraph. This means that continuous section breaks might not specify certain page-level section properties, since they must be inherited from the following section. These breaks, however, can specify other section properties, such as line numbering and footnote/endnote settings.
evenPage (Even Page Section Break)	Specifies an even page section break, which begins the new section on the next even-numbered page, leaving the next odd page blank if necessary.
nextColumn (Column Section Break)	Specifies a column section break, which begins the new section on the following column on the page.
nextPage (Next Page Section Break)	Specifies a next page section break, which begins the new section on the following page.
oddPage (Odd Page Section Break)	Specifies an odd page section break, which begins the

Enumeration Value	Description
	new section on the next odd-numbered page, leaving the next even page blank if necessary.

Referenced By
type@val (§2.6.22)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SectionMark">
   <restriction base="xsd:string">
      <enumeration value="nextPage"/>
      <enumeration value="nextColumn"/>
      <enumeration value="continuous"/>
      <enumeration value="evenPage"/>
      <enumeration value="oddPage"/>
   </restriction>
</simpleType>
```

2.18.85 ST_Shd (Shading Patterns)

This simple type specifies the pattern which shall be used to lay the pattern color over the background color for a shading.

This pattern consists of a mask which is applied over the background shading color to get the locations where the pattern color should be shown. Each of these possible masks are shown in the enumeration values located below. In each example, black has been used as the fill color, and white has been used as the pattern color.

[*Example*: Consider a shaded paragraph which uses a 10 percent foreground fill, resulting in the following WordprocessingML:

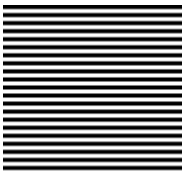
```
<w:shd w:val="pct10" .../>
```

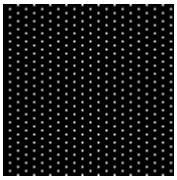
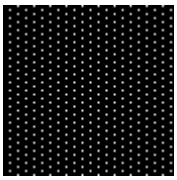
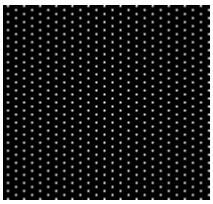
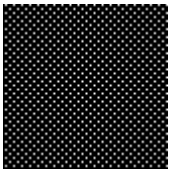
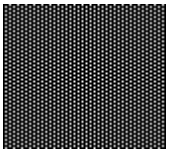
This shading val is `pct10`, indicating that the border style is a 10 percent foreground fill mask. *end example*]

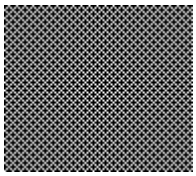
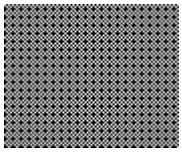
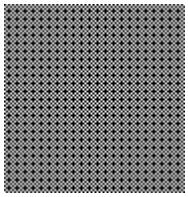
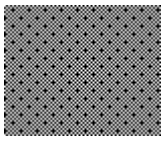
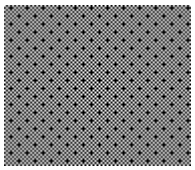
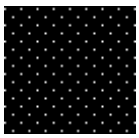
This simple type's contents are a restriction of the XML Schema string datatype.

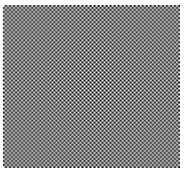
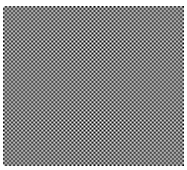
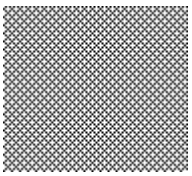
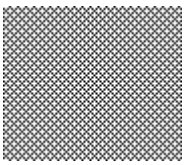
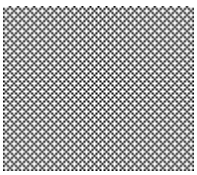
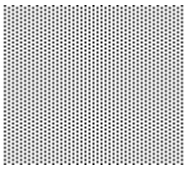
The following are possible enumeration values for this type:

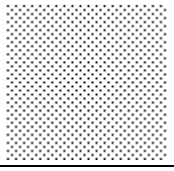
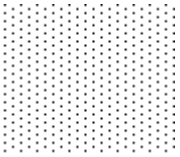
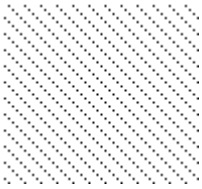
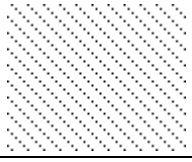
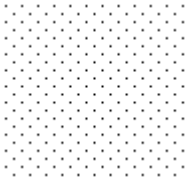
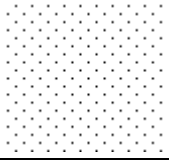
Enumeration Value	Description
clear (No Pattern)	Specifies that there shall be no pattern used on the current shaded region (i.e. the pattern shall be a complete fill with the background color), as follows:

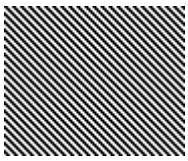
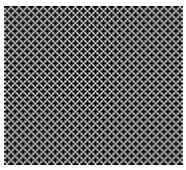
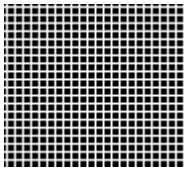
Enumeration Value	Description
	
diagCross (Diagonal Cross Pattern)	Specifies that the pattern used on the current shaded region shall be a series of diagonal crosses, as follows: 
diagStripe (Diagonal Stripe Pattern)	Specifies that the pattern used on the current shaded region shall be a series of diagonal stripes, as follows: 
horzCross (Horizontal Cross Pattern)	Specifies that the pattern used on the current shaded region shall be a series of horizontal crosses, as follows: 
horzStripe (Horizontal Stripe Pattern)	Specifies that the pattern used on the current shaded region shall be a series of horizontal stripes, as follows: 
nil (No Pattern)	Specifies that there shall be no pattern used on the current shaded region (i.e. the pattern shall be a complete fill with the background color), as follows: 
pct10 (10% Fill Pattern)	Specifies that the pattern used for the current shaded

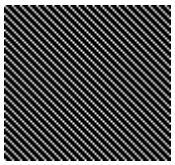
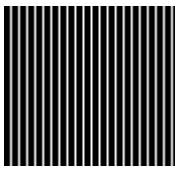
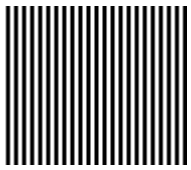
Enumeration Value	Description
	region shall be a 10% fill pattern, as follows: 
pct12 (12.5% Fill Pattern)	Specifies that the pattern used for the current shaded region shall be a 12.5% fill pattern, as follows: 
pct15 (15% Fill Pattern)	Specifies that the pattern used for the current shaded region shall be a 15% fill pattern, as follows: 
pct20 (20% Fill Pattern)	Specifies that the pattern used for the current shaded region shall be a 20% fill pattern, as follows: 
pct25 (25% Fill Pattern)	Specifies that the pattern used for the current shaded region shall be a 25% fill pattern, as follows: 
pct30 (30% Fill Pattern)	Specifies that the pattern used for the current shaded region shall be a 30% fill pattern, as follows:

Enumeration Value	Description
	
pct35 (35% Fill Pattern)	Specifies that the pattern used for the current shaded region shall be a 35% fill pattern, as follows: 
pct37 (37.5% Fill Pattern)	Specifies that the pattern used for the current shaded region shall be a 37.5% fill pattern, as follows: 
pct40 (40% Fill Pattern)	Specifies that the pattern used for the current shaded region shall be a 40% fill pattern, as follows: 
pct45 (45% Fill Pattern)	Specifies that the pattern used for the current shaded region shall be a 45% fill pattern, as follows: 
pct5 (5% Fill Pattern)	Specifies that the pattern used for the current shaded region shall be a 5% fill pattern, as follows: 
pct50 (50% Fill Pattern)	Specifies that the pattern used for the current shaded region shall be a 50% fill pattern, as follows:

Enumeration Value	Description
	
pct55 (55% Fill Pattern)	Specifies that the pattern used for the current shaded region shall be a 55% fill pattern, as follows: 
pct60 (60% Fill Pattern)	Specifies that the pattern used for the current shaded region shall be a 60% fill pattern, as follows: 
pct62 (62.5% Fill Pattern)	Specifies that the pattern used for the current shaded region shall be a 62.5% fill pattern, as follows: 
pct65 (65% Fill Pattern)	Specifies that the pattern used for the current shaded region shall be a 65% fill pattern, as follows: 
pct70 (70% Fill Pattern)	Specifies that the pattern used for the current shaded region shall be a 70% fill pattern, as follows: 
pct75 (75% Fill Pattern)	Specifies that the pattern used for the current shaded region shall be a 75% fill pattern, as follows:

Enumeration Value	Description
	
pct80 (80% Fill Pattern)	Specifies that the pattern used for the current shaded region shall be a 80% fill pattern, as follows: 
pct85 (85% Fill Pattern)	Specifies that the pattern used for the current shaded region shall be a 85% fill pattern, as follows: 
pct87 (87.5% Fill Pattern)	Specifies that the pattern used for the current shaded region shall be a 87.5% fill pattern, as follows: 
pct90 (90% Fill Pattern)	Specifies that the pattern used for the current shaded region shall be a 90% fill pattern, as follows: 
pct95 (95% Fill Pattern)	Specifies that the pattern used for the current shaded region shall be a 95% fill pattern, as follows: 
reverseDiagStripe (Reverse Diagonal Stripe Pattern)	Specifies that the pattern used on the current shaded

Enumeration Value	Description
	region shall be a series of reverse diagonal stripes, as follows: 
solid (100% Fill Pattern)	Specifies that the pattern used for the current shaded region shall be a 100% fill pattern, as follows:
thinDiagCross (Thin Diagonal Cross Pattern)	Specifies that the pattern used on the current shaded region shall be a series of thin diagonal crosses, as follows: 
thinDiagStripe (Thin Diagonal Stripe Pattern)	Specifies that the pattern used on the current shaded region shall be a series of thin diagonal stripes, as follows: 
thinHorzCross (Thin Horizontal Cross Pattern)	Specifies that the pattern used on the current shaded region shall be a series of thin horizontal crosses, as follows: 
thinHorzStripe (Thin Horizontal Stripe Pattern)	Specifies that the pattern used on the current shaded region shall be a series of thin horizontal stripes, as follows:

Enumeration Value	Description
	
thinReverseDiagStripe (Thin Reverse Diagonal Stripe Pattern)	Specifies that the pattern used on the current shaded region shall be a series of thin reverse diagonal stripes, as follows: 
thinVertStripe (Thin Vertical Stripe Pattern)	Specifies that the pattern used on the current shaded region shall be a series of thin vertical stripes, as follows: 
vertStripe (Vertical Stripe Pattern)	Specifies that the pattern used on the current shaded region shall be a series of vertical stripes, as follows: 

Referenced By
shd@val (§2.3.2.30); shd@val (§2.4.33); shd@val (§2.4.34); shd@val (§2.4.35); shd@val (§2.3.1.31)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Shd">
   <restriction base="xsd:string">
      <enumeration value="nil"/>
      <enumeration value="clear"/>
      <enumeration value="solid"/>
      <enumeration value="horzStripe"/>
      <enumeration value="vertStripe"/>
      <enumeration value="reverseDiagStripe"/>
      <enumeration value="diagStripe"/>
      <enumeration value="horzCross"/>
      <enumeration value="diagCross"/>
      <enumeration value="thinHorzStripe"/>
      <enumeration value="thinVertStripe"/>
      <enumeration value="thinReverseDiagStripe"/>
      <enumeration value="thinDiagStripe"/>
      <enumeration value="thinHorzCross"/>
      <enumeration value="thinDiagCross"/>
      <enumeration value="pct5"/>
      <enumeration value="pct10"/>
      <enumeration value="pct12"/>
      <enumeration value="pct15"/>
      <enumeration value="pct20"/>
      <enumeration value="pct25"/>
      <enumeration value="pct30"/>
      <enumeration value="pct35"/>
      <enumeration value="pct37"/>
      <enumeration value="pct40"/>
      <enumeration value="pct45"/>
      <enumeration value="pct50"/>
      <enumeration value="pct55"/>
      <enumeration value="pct60"/>
      <enumeration value="pct62"/>
      <enumeration value="pct65"/>
      <enumeration value="pct70"/>
      <enumeration value="pct75"/>
      <enumeration value="pct80"/>
      <enumeration value="pct85"/>
      <enumeration value="pct87"/>
      <enumeration value="pct90"/>
      <enumeration value="pct95"/>
   </restriction>
</simpleType>
```

2.18.86 ST_ShortHexNumber (Two Digit Hexadecimal Number Value)

This simple type specifies a number value specified as a two octet hexadecimal number), whose contents are interpreted based on the context of the parent XML element.

[*Example*: Consider the following value for a node of type ST_ShortHexNumber: `2F6C`.

This value is valid, as it contains two hexadecimal octets, each an encoding of an octet of the actual decimal number value. *end example*]

This simple type's contents are a restriction of the XML Schema hexBinary datatype.

This simple type also specifies the following restrictions:

- This simple type's contents must have a length of exactly 2 characters.

Referenced By
stylePaneFormatFilter@val (§2.15.1.86); stylePaneSortMethod@val (§2.15.1.87); sym@char (§2.3.3.29); tblLook@val (§2.4.51); tblLook@val (§2.4.52)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ShortHexNumber">
   <restriction base="xsd:hexBinary">
      <length value="2"/>
   </restriction>
</simpleType>
```

2.18.87 ST_SignedHpsMeasure (Signed Measurement in Half-Points)

This simple type specifies that its contents will contain a positive or negative whole number, whose contents consist of a measurement in half-points (equivalent to 1/144th of an inch).

The contents of this measurement are interpreted based on the context of the parent XML element.

[*Example*: Consider an attribute value of -72 whose type is ST_HpsMeasure. This attribute value specifies a size of negative one-half of an inch or -36 points (-72 halves of a point = -36 points = -0.5 inches). *end example*]

This simple type's contents are a restriction of the XML Schema integer datatype.

Referenced By
position@val (§2.3.2.22)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SignedHpsMeasure">
   <restriction base="xsd:integer"/>
</simpleType>
```

2.18.88 ST_SignedTwipsMeasure (Signed Measurement in Twentieths of a Point)

This simple type specifies that its contents will contain a positive or negative whole number, whose contents consist of a measurement in twentieths of a point (equivalent to 1/1440th of an inch).

The contents of this measurement are interpreted based on the context of the parent XML element.

[*Example*: Consider an attribute value of `720` whose type is ST_EighthPointMeasure. This attribute value specifies a size of one-half of an inch or 36 points (`720` twentieths of a point = 36 points = 0.5 inches). *end example*]

This simple type's contents are a restriction of the XML Schema integer datatype.

Referenced By
framePr@x (§2.3.1.11); framePr@y (§2.3.1.11); ind@left (§2.3.1.12); ind@right (§2.3.1.12); legacy@legacyIndent (§2.9.5); marBottom@val (§2.15.2.23); marLeft@val (§2.15.2.25); marRight@val (§2.15.2.26); marTop@val (§2.15.2.27); pgMar@bottom (§2.6.11); pgMar@top (§2.6.11); spacing@line (§2.3.1.33); spacing@val (§2.3.2.33); tab@pos (§2.3.1.37); tblpPr@tblpX (§2.4.54); tblpPr@tblpY (§2.4.54)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SignedTwipsMeasure">
   <restriction base="xsd:integer"/>
</simpleType>
```

2.18.89 ST_String (String)

This simple type specifies that its contents will contain a string. The contents of this string are interpreted based on the context of the parent XML element.

[*Example*: Consider the following WordprocessingML fragment:

```
<w:pPr>
  <w:pStyle w:val="heading1" />
</w:pPr>
```

The value of the val attribute is the ID of the associated paragraph style's `styleId`. However, consider the following fragment:

```
<w:sdtPr>
  <w:alias w:val="SDT Title Example" />
  ...
</w:sdtPr>
```

In this case, the decimal number in the val attribute is the caption of the parent structured document tag. In each case, the value is of type ST_String, and therefore must be interpreted in the context of the parent element. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

Referenced By
activeWritingStyle@appName (§2.15.1.1); addressFieldName@val (§2.14.3); alias@val (§2.5.2.1); aliases@val (§2.7.3.1); altName@val (§2.8.2.1); attachedSchema@val (§2.15.1.5); attr@name (§2.5.1.1); attr@name (§2.5.1.2); attr@uri (§2.5.1.1); attr@uri (§2.5.1.2); attr@val (§2.5.1.1); attr@val (§2.5.1.2); autoCaption@caption (§2.15.1.7); autoCaption@name (§2.15.1.7); basedOn@val (§2.7.3.3);

Referenced By
bookmarkStart@name (§2.13.6.2); caption@name (§2.15.1.16); cellDel@author (§2.13.5.1); cellIns@author (§2.13.5.2); cellMerge@author (§2.13.5.3); clickAndTypeStyle@val (§2.15.1.19); comboBox@lastValue (§2.5.2.5); comment@author (§2.13.4.2); comment@initials (§2.13.4.2); connectString@val (§2.14.8); control@name (§2.3.3.2); control@name (§2.3.3.3); control@shapeid (§2.3.3.2); control@shapeid (§2.3.3.3); customXml@element (§2.5.1.6); customXml@element (§2.5.1.4); customXml@element (§2.5.1.3); customXml@element (§2.5.1.5); customXml@uri (§2.5.1.4); customXml@uri (§2.5.1.3); customXml@uri (§2.5.1.5); customXml@uri (§2.5.1.6); customXmlDelRangeStart@author (§2.13.5.5); customXmlInsRangeStart@author (§2.13.5.7); customXmlMoveFromRangeStart@author (§2.13.5.9); customXmlMoveToRangeStart@author (§2.13.5.11); dataBinding@prefixMappings (§2.5.2.6); dataBinding@storeItemID (§2.5.2.6); dataBinding@xpath (§2.5.2.6); dateFormat@val (§2.5.2.8); decimalSymbol@val (§2.15.1.22); default@val (§2.16.10); defaultTableStyle@val (§2.15.1.23); del@author (§2.13.5.12); del@author (§2.13.5.13); del@author (§2.13.5.14); del@author (§2.13.5.15); description@val (§2.12.4); docPart@val (§2.5.2.9); docPartCategory@val (§2.5.2.10); docPartGallery@val (§2.5.2.11); documentProtection@algIdExtSource (§2.15.1.28); documentProtection@cryptProvider (§2.15.1.28); documentProtection@cryptProviderTypeExtSource (§2.15.1.28); docVar@name (§2.15.1.30); docVar@val (§2.15.1.30); dropDownList@lastValue (§2.5.2.15); encoding@val (§2.15.2.14); fldSimple@instr (§2.16.21); font@name (§2.8.2.10); format@val (§2.16.22); hyperlink@anchor (§2.16.24); hyperlink@docLocation (§2.16.24); hyperlink@tgtFrame (§2.16.24); hyperlink@tooltip (§2.16.24); ins@author (§2.13.5.16); ins@author (§2.13.5.17); ins@author (§2.13.5.18); ins@author (§2.13.5.19); ins@author (§2.13.5.20); link@val (§2.7.3.6); listEntry@val (§2.16.26); listItem@displayText (§2.5.2.20); listItem@displayText (§2.5.2.21); listItem@value (§2.5.2.20); listItem@value (§2.5.2.21); listSeparator@val (§2.15.1.56); lsdException@name (§2.7.3.8); lvlText@val (§2.9.12); mailSubject@val (§2.14.21); mappedName@val (§2.14.23); moveFrom@author (§2.13.5.21); moveFrom@author (§2.13.5.22); moveFromRangeStart@author (§2.13.5.24); moveFromRangeStart@name (§2.13.5.24); moveTo@author (§2.13.5.25); moveTo@author (§2.13.5.26); moveToRangeStart@author (§2.13.5.28); moveToRangeStart@name (§2.13.5.28); name@val (§2.12.13); name@val (§2.7.3.9); name@val (§2.9.14); name@val (§2.14.24); name@val (§2.15.2.29); name@val (§2.12.12); next@val (§2.7.3.10); noLineBreaksAfter@val (§2.15.1.58); noLineBreaksBefore@val (§2.15.1.59); numberingChange@author (§2.13.5.29); numberingChange@author (§2.13.5.30); numberingChange@original (§2.13.5.29); numberingChange@original (§2.13.5.30); numStyleLink@val (§2.9.22); permEnd@id (§2.13.7.1); permStart@ed (§2.13.7.2); permStart@id (§2.13.7.2); placeholder@val (§2.5.1.8); pPrChange@author (§2.13.5.31); pStyle@val (§2.3.1.27); pStyle@val (§2.9.25); query@val (§2.14.26); rFonts@ascii (§2.3.2.24); rFonts@cs (§2.3.2.24); rFonts@eastAsia (§2.3.2.24); rFonts@hAnsi (§2.3.2.24); rPrChange@author (§2.13.5.32); rPrChange@author (§2.13.5.33); rStyle@val (§2.3.2.27); saveThroughXslt@solutionID (§2.15.1.76); sectPrChange@author (§2.13.5.34); smartTag@element (§2.5.1.9); smartTag@uri (§2.5.1.9); smartTagType@name (§2.15.1.82); smartTagType@namespaceuri (§2.15.1.82); smartTagType@url (§2.15.1.82); ST_Lang (§2.18.51); style@styleId (§2.7.3.17); style@val (§2.12.14); styleLink@val (§2.9.29); sym@font (§2.3.3.29); sz@val (§2.15.2.39); sz@val (§2.15.2.40); table@val (§2.14.31); tag@val (§2.5.2.40); tblPrChange@author (§2.13.5.36); tblPrExChange@author (§2.13.5.37); tblStyle@val (§2.4.59); tcPrChange@author (§2.13.5.38); trPrChange@author (§2.13.5.39); udl@val (§2.14.34); writeProtection@algIdExtSource (§2.15.1.94); writeProtection@cryptProvider (§2.15.1.94); writeProtection@cryptProviderTypeExtSource (§2.15.1.94)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_String">
   <restriction base="xsd:string"/>
</simpleType>
```

2.18.90 ST_StyleType (Style Types)

This simple type specifies the possible values for the types of style definitions defined within a WordprocessingML document. WordprocessingML supports six types of style definitions:

- Paragraph styles
- Character styles
- Table styles
- Numbering styles
- Linked styles (paragraph + character)
- Default paragraph + character properties

Each of the first four types corresponds to a different value below, and therefore defines the type of the current style. [*Note*: The last two types are unique in that they are not simply a style type: a linked style is a pairing of a character and paragraph style via the link element (§2.7.3.6); and the document default properties are defined via the docDefaults element (§2.7.4.1). *end note*]

[*Example*: Consider a style defined as follows:

```
<w:style w:type="paragraph" … >
  <w:name w:val="My Paragraph Style"/>
  <w:rPr>
    <w:b/>
  </w:rPr>
</w:style>
```

The type attribute is of type ST_StyleType, and its value of paragraph specifies that this style definition creates a paragraph style. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
character (Character Style)	Specifies that the parent style definition is a character style.
numbering (Numbering Style)	Specifies that the parent style definition is a numbering style.
paragraph (Paragraph Style)	Specifies that the parent style definition is a paragraph style.

Enumeration Value	Description
table (Table Style)	Specifies that the parent style definition is a table style.

Referenced By
style@type (§2.7.3.17)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_StyleType">
   <restriction base="xsd:string">
      <enumeration value="paragraph"/>
      <enumeration value="character"/>
      <enumeration value="table"/>
      <enumeration value="numbering"/>
   </restriction>
</simpleType>
```

2.18.91 ST_TabJc (Custom Tab Stop Type)

This simple type specifies the available types of custom tab stop, which determines the behavior of the tab stop and the alignment which shall be applied to text entered at the current custom tab stop.

[*Example*: Consider a custom tab stops at 1.5" in a WordprocessingML document. This tab stop would be contained within a tab element defining the tab stop as follows:

```
<w:tab w:val="left" w:pos="2160" />
```

The val attribute specifies that this custom tab stop shall align all text entered at its location to its left. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
bar (Bar Tab)	Specifies that the current tab is a bar tab. A *bar tab* is a tab which does not result in a custom tab stop in the parent paragraph (this tab stop location shall be skipped when positioning custom tab characters), but instead shall be used to draw a vertical line (or bar) at this location in the parent paragraph.
center (Centered Tab)	Specifies that the current tab stop shall result in a location in the document where all following text is centered (i.e. all text runs following this tab stop and preceding the next tab stop shall be centered around the tab stop location).

Enumeration Value	Description
clear (No Tab Stop)	Specifies that the current tab stop is cleared and shall be removed and ignored when processing the contents of this document.
decimal (Decimal Tab)	Specifies that the current tab stop shall result in a location in the document where all following text is aligned around the first decimal character in the following text runs. All text runs before the first decimal character shall be before the tab stop, all text runs after it shall be after the tab stop location.
left (Left Tab)	Specifies that the current tab stop shall result in a location in the document where all following text is left aligned (i.e. all text runs following this tab stop and preceding the next tab stop shall be left aligned with respect to the tab stop location).
num (List Tab)	Specifies that the current tab is a list tab, which is the tab stop between the numbering and the paragraph contents in a numbered paragraph. [*Note*: This justification style is used for backwards compatibility with earlier word processors, and should be deprecated in favor of hanging paragraph indentation. *end note*]
right (Right Tab)	Specifies that the current tab stop shall result in a location in the document where all following text is right aligned (i.e. all text runs following this tab stop and preceding the next tab stop shall be right aligned with respect to the tab stop location).

Referenced By
tab@val (§2.3.1.37)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TabJc">
   <restriction base="xsd:string">
      <enumeration value="clear"/>
      <enumeration value="left"/>
      <enumeration value="center"/>
      <enumeration value="right"/>
      <enumeration value="decimal"/>
      <enumeration value="bar"/>
      <enumeration value="num"/>
   </restriction>
</simpleType>
```

2.18.92 ST_TabTlc (Custom Tab Stop Leader Character)

This simple type specifies the characters which may be used to fill in the space created by a tab which ends at this custom tab stop. The chosen character shall be repeated as required to completely fill the tab spacing generated by the tab character.

[*Example*: Consider a tab stop which should be preceded by a sequence of underscore characters, as follows:

_____Text at the tab stop

This tab stop would have a leader attribute value of underscore, indicating that the tab stop shall be preceded by underscore characters as needed to fill the tab spacing. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
dot (Dotted leader line)	Specifies that the leader character for this custom tab stop shall be a dot. [*Example*: .....Text at tab stop. *end example*]
heavy (Heavy solid leader line)	Specifies that the leader character for this custom tab stop shall be a heavy solid line, or an underscore. [Note: This setting is used for backwards compatibility with earlier word processors, and should be deprecated in favor of other leader characters. It may be displayed using underscores if desired. *end note*] [*Example*:

Enumeration Value	Description
	_____Text at tab stop. *end example*]
hyphen (Dashed tab stop leader line)	Specifies that the leader character for this custom tab stop shall be a hyphen. [*Example*: ---------- Text at tab stop. *end example*]
middleDot (Middle dot leader line)	Specifies that the leader character for this custom tab stop shall be a centered dot. [*Example*: ·····Text at tab stop. *end example*]
none (No tab stop leader)	Specifies that there shall be no leader character for this custom tab. [*Example*: Text at tab stop. *end example*]
underscore (Solid leader line)	Specifies that the leader character for this custom tab stop shall be an underscore. [*Example*: _____Text at tab stop. *end example*]

Referenced By
tab@leader (§2.3.1.37)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TabTlc">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="dot"/>
      <enumeration value="hyphen"/>
      <enumeration value="underscore"/>
      <enumeration value="heavy"/>
      <enumeration value="middleDot"/>
   </restriction>
</simpleType>
```

2.18.93 ST_TargetScreenSz (Target Screen Sizes for Generated Web Pages)

This simple type specifies possible ideal minimum target screen sizes (width by height, specified in pixels) for which web pages generated may be optimized when saving this document as a web page.

[*Example*: Consider a WordprocessingML document which contains the following content within the web settings part:

```
<w:webSettings>
  <w:targetScreenSz w:val="1600x1200" />
</w:webSettings>
```

The targetScreenSz element's val attribute has a value of `1600x1200`, which specifies that a target screen size of 1600 by 1200 pixels shall be assumed when saving this document as a web page. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
1024x768 (Optimize for 1024x768)	Specifies that web pages produced from this document should be optimized for a screen size of 1024x768.
1152x882 (Optimize for 1152x882)	Specifies that web pages produced from this document should be optimized for a screen size of 1152x882.
1152x900 (Optimize for 1152x900)	Specifies that web pages produced from this document should be optimized for a screen size of 1152x900.
1280x1024 (Optimize for 1280x1024)	Specifies that web pages produced from this document should be optimized for a screen size of 1280x1024.
1600x1200 (Optimize for 1600x1200)	Specifies that web pages produced from this document should be optimized for a screen size of 1600x1200.

Enumeration Value	Description
1800x1440 (Optimize for 1800x1440)	Specifies that web pages produced from this document should be optimized for a screen size of 1800x1440.
1920x1200 (Optimize for 1920x1200)	Specifies that web pages produced from this document should be optimized for a screen size of 1920x1200.
544x376 (Optimize for 544x376)	Specifies that web pages produced from this document should be optimized for a screen size of 544x376.
640x480 (Optimize for 640x480)	Specifies that web pages produced from this document should be optimized for a screen size of 640x480.
720x512 (Optimize for 720x512)	Specifies that web pages produced from this document should be optimized for a screen size of 720x512.
800x600 (Optimize for 800x600)	Specifies that web pages produced from this document should be optimized for a screen size of 800x600.

Referenced By
targetScreenSz@val (§2.15.2.41)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TargetScreenSz">
   <restriction base="xsd:string">
      <enumeration value="544x376"/>
      <enumeration value="640x480"/>
      <enumeration value="720x512"/>
      <enumeration value="800x600"/>
      <enumeration value="1024x768"/>
      <enumeration value="1152x882"/>
      <enumeration value="1152x900"/>
      <enumeration value="1280x1024"/>
      <enumeration value="1600x1200"/>
      <enumeration value="1800x1440"/>
      <enumeration value="1920x1200"/>
   </restriction>
</simpleType>
```

2.18.94 ST_TblLayoutType (Table Layout Type)

This simple type defines the possible types of layout algorithms which may be used to lay out a table within a WordprocessingML document.

These algorithms are defined in the following paragraphs (noting, of course, that implementations are free to implement more efficient versions of each).

Fixed Width Table Layout - This method of table layout uses the preferred widths on the table items to generate the final sizing of the table, but does not change that size regardless of the contents of each table cell, hence the table is fixed width.

[*Guidance*: Although an application may choose to use a different process, this layout could be performed as follows:

- The table grid is used to create the set of shared columns in the table and their initial widths as defined in the tblGrid element (§2.4.44)
- The table's total width is defined based on the tblW property (§2.4.61) – if it is set to `auto` or `nil`, then the width is not yet determined and will be specified using the row and cell information.
- The first table row is read and the initial number of grid units before the row starts is skipped. The width of the skipped grid columns is set using the wBefore property (§2.4.83).
- The first cell is placed on the grid, and the width of the specified grid column span set by gridSpan (§2.4.13) is set based on the tcW property (§2.4.68).
- Each additional cell is placed on the grid.
- If at any stage, the preferred width requested for the cells exceeds the preferred width of the table, then each grid column is proportionally reduced in size to fit the table width.
- If the grid is exceeded (e.g. tblGrid specifies three grid columns, but the second cell has a gridSpan of three), the grid is dynamically increased with a default width for the new grid column.
- For each subsequent row, cells are placed on the grid, and each grid column is adjusted to be the maximum value of the requested widths (if the widths do not agree) by adding width to the last cell that ends with that grid column. Again, if at any point, the space requested for the cells exceeds the width of the table, then each grid column is proportionally reduced in size to fit the table width.

end guidance]

The resulting table shall be displayed regardless of its contents to the size requested.

AutoFit Table Layout - This method of table layout uses the preferred widths on the table items to generate the final sizing of the table, but then uses the contents of each cell to determine final column widths.

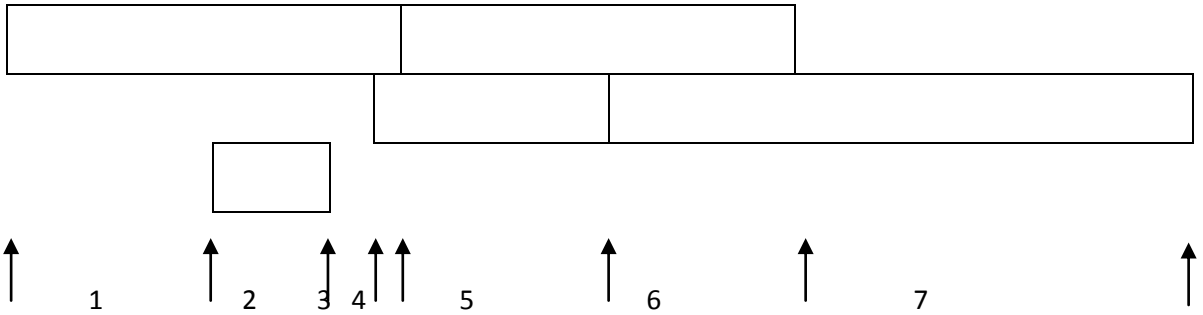
[*Guidance*: This layout may be performed in any manner available to an application, but one algorithm as follows may be used:

- Perform the steps above to lay out the fixed width version of the table.
- Calculate the minimum content width - the width of the cell's contents including all possible line breaking locations (or the cell's width, if the width of the content is smaller), and the maximum content width -the width of the cell's contents (assuming no line breaking not generated by explicit line breaks).
- The minimum and maximum content width of all cells that span a single grid column is the minimum and maximum content width of that column.

- For cells which span multiple grid columns, enlarge all cells which it spans as needed to meet that cell's minimum width.
- If any cell in a grid column has a preferred width, the first such width overrides the maximum width of the column's contents.
- Place the text in the cells in the table, respecting the minimum content width of each cell's content. If a cell's minimum content width exceeds the cell's current width, preferences are overridden as follows:
- First, override the column widths by making all other grid columns proportionally smaller until each it at its minimum width. This cell may then grow to any width between its own minimum and maximum width.
- Next, override the preferred table width until the table reaches the page width.
- Finally, force a line break in each cell's contents as needed

end guidance]

[*Example*: Consider the following fixed width table, which makes extensive use of resized and merged cells on what is actually just a seven-column grid. (The arrows point to each (invisible) vertical line of the grid and the numbers refer to the grid columns):



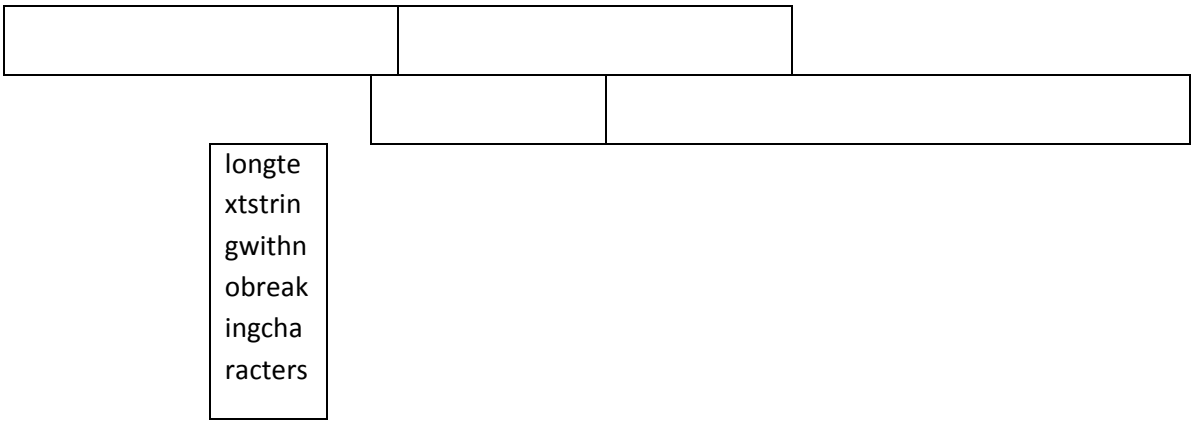
Although the table is visually complex, the standard rules apply: the first cell in the table is simply a cell which spans four grid units horizontally, as specified in the gridSpan element, and whose preferred with is 2952 twentieths of a point, specified in the tcW element:

```
<w:tc>
  <w:tcPr>
    <w:tcW w:w="2952" w:type="dxa"/>
    <w:gridSpan w:val="4"/>
  </w:tcPr>
  <w:p/>
</w:tc>
```

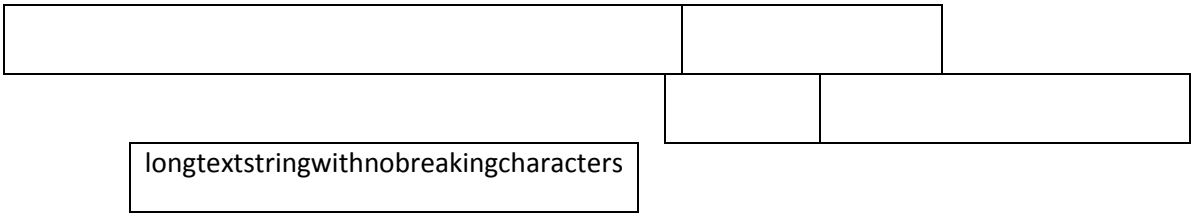
Similarly, all cells indented from the stand and end of the grid specify that indent using the gridBefore and gridAfter elements. For example, the XML for the second row in the table shows that that row starts three grid units into the table:

```
<w:tr>
  <w:trPr>
    <w:gridBefore w:val="3"/>
    <w:wBefore w:w="2748" w:type="dxa"/>
  </w:trPr>
  …
</w:tr>
```

If we take this fixed width table and introduce a long string into the single cell in row three, we see that the presence of this text does not affect cell widths:



If we now turn on the AutoFit property, we see that the algorithm for this AutoFit table causes grid column two to increase in size, proportionally decreasing the other grid columns' size to accommodate the long non-breaking string in the last cell:



Each of the other grid columns was reduced, but since all columns are not at their minimum size, the table width is not increased even though the table is not yet at the page width. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
autofit (AutoFit Table Layout)	Specifies that this table shall use an AutoFit table layout algorithm.

Enumeration Value	Description
fixed (Fixed Width Table Layout)	Specifies that this table shall use the fixed width table layout algorithm described above.

Referenced By
tblLayout@type (§2.4.49); tblLayout@type (§2.4.50)

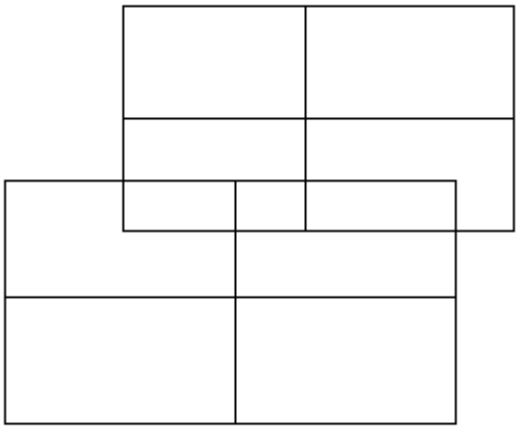
The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TblLayoutType">
   <restriction base="xsd:string">
      <enumeration value="fixed"/>
      <enumeration value="autofit"/>
   </restriction>
</simpleType>
```

2.18.95 ST_TblOverlap (Table Overlap Setting)

This simple type contains the possible settings for a floating table which shall be used to determine if the table can overlap with other floating tables when displayed in the document.

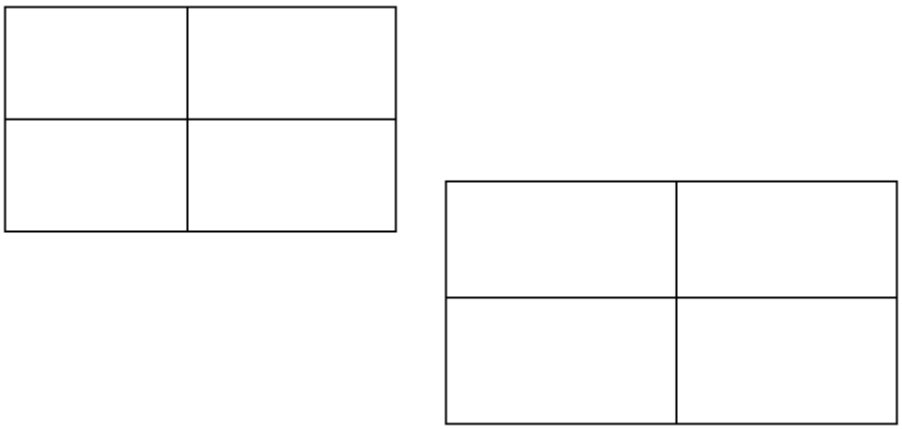
[*Example*: Consider two floating tables in a WordprocessingML document which overlap when displayed, as follows:



If either of these tables specifies that it shall not allow overlapping, using the following WordprocessingML:

```
<w:tblPr>
  <w:tblOverlap w:val="never"/>
</w:tblPr>
```

The resulting tables shall not overlap, and must be adjusted at display time to prevent any overlapping, for example:



The value of never specifies that the specified table cannot overlap with other floating tables in the document. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
never (Floating Table Cannot Overlap)	Specifies that the parent table, if floating, shall never be displayed in a state where it would be overlapping another floating table in the document. If two floating tables intersect and this option is set on either of them, then one or both tables shall be adjusted as needed to ensure that the table whose value is never is not overlapped when displayed.
overlap (Floating Table Can Overlap)	Specifies that the parent table, if floating, may be displayed in a state where it would be overlapping another floating table in the document.

Referenced By
tblOverlap@val (§2.4.53)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TblOverlap">
   <restriction base="xsd:string">
      <enumeration value="never"/>
      <enumeration value="overlap"/>
   </restriction>
</simpleType>
```

2.18.96 ST_TblStyleOverrideType (Conditional Table Style Formatting Types)

This simple type specifies possible values for the sections of the table to which the current conditional formatting properties shall be applied when this table style is used.

[*Example*: Consider a table style which contains conditional formatting, defined as follows:

```
<w:style w:type="table" …>
  …
  <w:tblStylePr w:type="lastRow">
    …
  </w:tblStylePr>
</w:style>
```

The type attribute value of `lastRow` specifies that this set of conditional formatting properties shall be applied to the last row of the table only. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
band1Horz (Banded Row Conditional Formatting)	Specifies that the table formatting applies to odd numbered groupings of rows.
band1Vert (Banded Column Conditional Formatting)	Specifies that the table formatting applies to odd numbered groupings of columns.
band2Horz (Even Row Stripe Conditional Formatting)	Specifies that the table formatting applies to even numbered groupings of rows.
band2Vert (Even Column Stripe Conditional Formatting)	Specifies that the table formatting applies to even numbered groupings of columns.
firstCol (First Column Conditional Formatting)	Specifies that the table formatting applies to the first column.
firstRow (First Row Conditional Formatting)	Specifies that the table formatting applies to the first row. Any subsequent row which has the tblHeader element present (§2.4.46) shall also use this conditional format.
lastCol (Last table column formatting)	Specifies that the table formatting applies to the last column.
lastRow (Last table row formatting)	Specifies that the table formatting applies to the last row.
neCell (Top right table cell formatting)	Specifies that the table formatting applies to the top right cell.
nwCell (Top left table cell formatting)	Specifies that the table formatting applies to the top

Enumeration Value	Description
	left cell.
seCell (Bottom right table cell formatting)	Specifies that the table formatting applies to the bottom right cell.
swCell (Bottom left table cell formatting)	Specifies that the table formatting applies to the bottom left cell.
wholeTable (Whole table formatting)	Specifies that the conditional formatting applies to the whole table.

Referenced By
tblStylePr@type (§2.7.5.6)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TblStyleOverrideType">
   <restriction base="xsd:string">
      <enumeration value="wholeTable"/>
      <enumeration value="firstRow"/>
      <enumeration value="lastRow"/>
      <enumeration value="firstCol"/>
      <enumeration value="lastCol"/>
      <enumeration value="band1Vert"/>
      <enumeration value="band2Vert"/>
      <enumeration value="band1Horz"/>
      <enumeration value="band2Horz"/>
      <enumeration value="neCell"/>
      <enumeration value="nwCell"/>
      <enumeration value="seCell"/>
      <enumeration value="swCell"/>
   </restriction>
</simpleType>
```

2.18.97 ST_TblWidth (Table Width Units)

This simple type specifies the possible values for the units of the width property being defined by a specific table width property. These properties are used to define various properties of a table, including: cell spacing, preferred width, and table margins.

[*Example*: Consider a table with a table cell bottom cell spacing with a type of dxa, as follows:

```
<w:bottom ... w:type="dxa" />
```

This type shall therefore be used to interpret the width specified in the w attribute as a value in twentieths of a point. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
auto (Automatically Determined Width)	Specifies that the value for the measurement of the current table width property in the parent table shall be automatically determined by the table layout algorithm when the table is displayed (this width can be adjusted as appropriate). If this value is inappropriate for the current measurement (i.e. this measurement is not affected by that algorithm), then this type and the associated value may be ignored.
dxa (Width in Twentieths of a Point)	Specifies that the value for the measurement of the current table width property in the parent table shall be interpreted as twentieths of a point (1/1440 of an inch).
nil (No Width)	Specifies that the current width is zero, regardless of any width value specified on the parent element.
pct (Width in Fiftieths of a Percent)	Specifies that the value for the measurement of the current table width property in the parent table shall be interpreted as fiftieths of a percent. [*Example*: 4975 = 99.5% *end example*] These percentages shall be calculated relative to the extents specified by the parent XML element. If this value is inappropriate for the current measurement (i.e. this measurement is not part of the width of the table), then this type and the associated value may be ignored.

Referenced By
bottom@type (§2.4.2); bottom@type (§2.4.5); left@type (§2.4.25); left@type (§2.4.26); right@type (§2.4.29); right@type (§2.4.31); tblCellSpacing@type (§2.4.41); tblCellSpacing@type (§2.4.42); tblCellSpacing@type (§2.4.43); tblInd@type (§2.4.47); tblInd@type (§2.4.48); tblW@type (§2.4.60); tblW@type (§2.4.61); tcW@type (§2.4.68); top@type (§2.4.72); top@type (§2.4.73); wAfter@type (§2.4.82); wBefore@type (§2.4.83)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TblWidth">
   <restriction base="xsd:string">
      <enumeration value="nil"/>
      <enumeration value="pct"/>
      <enumeration value="dxa"/>
      <enumeration value="auto"/>
   </restriction>
</simpleType>
```

2.18.98 ST_TextAlignment (Vertical Text Alignment Types)

This simple type specifies the type of vertical alignment which shall be used to align the characters on each line in the parent object.

[*Example*: Consider a paragraph of text of different font sizes, as follows:

If the text on this paragraph shall be aligned based on the top point of the maximum character height, that requirement would be specified as follows in the WordprocessingML:

```
<w:pPr>
  <w:textAlignment w:val="top" />
</w:pPr>
```

The resulting text would be top aligned, as follows:

The characters are all aligned to the maximum character extent on the line. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
auto (Automatically Determine Alignment)	Specifies that all text in the parent object shall be aligned automatically when displayed.
baseline (Align Text at Baseline)	Specifies that all text in the parent object shall be aligned to the baseline of each character when displayed.

Enumeration Value	Description
bottom (Align Text at Bottom)	Specifies that all text in the parent object shall be aligned to the bottom of each character when displayed.
center (Align Text at Center)	Specifies that all text in the parent object shall be aligned to the center of each character when displayed.
top (Align Text at Top)	Specifies that all text in the parent object shall be aligned to the top of each character when displayed.

Referenced By
textAlignment@val (§2.3.1.39)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextAlignment">
   <restriction base="xsd:string">
      <enumeration value="top"/>
      <enumeration value="center"/>
      <enumeration value="baseline"/>
      <enumeration value="bottom"/>
      <enumeration value="auto"/>
   </restriction>
</simpleType>
```

2.18.99 ST_TextboxTightWrap (Lines To Tight Wrap Within Text Box)

This simple type specifies the lines in the parent paragraph which shall allow the text to be tight wrapped to the paragraph (and not the containing text box) extents when displaying the document.

[*Example*: Consider a paragraph in a text box which meets the criteria specified above which shall allow wrapping to the text extents on its first line only. That requirement would be specified using the following WordprocessingML:

```
<w:pPr>
  <w:textboxTightWrap w:val="firstLineOnly" />
</w:pPr>
```

The resulting paragraph would allow text to tightly wrap to the contents of its first line only. All other lines would wrap to the text box's extents. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description

Enumeration Value	Description
allLines (Tight Wrap All Lines)	Specifies that all lines in the paragraph shall allow surrounding text to be tight wrapped to their extents and not the containing text box's extents.
firstAndLastLine (Tight Wrap First and Last Lines)	Specifies that only the first and last lines in the paragraph shall allow surrounding text to be tight wrapped to their extents and not the containing text box's extents.
firstLineOnly (Tight Wrap First Line)	Specifies that only the first line in the paragraph shall allow surrounding text to be tight wrapped to their extents and not the containing text box's extents.
lastLineOnly (Tight Wrap Last Line)	Specifies that only the last line in the paragraph shall allow surrounding text to be tight wrapped to their extents and not the containing text box's extents.
none (Do Not Tight Wrap)	Specifies that no lines in the paragraph shall allow surrounding text to be tight wrapped to their extents and not the containing text box's extents.

Referenced By
textboxTightWrap@val (§2.3.1.40)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextboxTightWrap">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="allLines"/>
      <enumeration value="firstAndLastLine"/>
      <enumeration value="firstLineOnly"/>
      <enumeration value="lastLineOnly"/>
   </restriction>
</simpleType>
```

2.18.100 ST_TextDirection (Text Flow Direction)

This simple type specifies the direction of the text flow for the parent object.

[*Example*: Consider an object in which text shall flow bottom to top vertically, and left to right horizontally. The `btLr` value in an element of type ST_TextDirection specifies that the text flow shall go bottom to top, and left to right. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description

Enumeration Value	Description
btLr (Bottom to Top, Left to Right)	Specifies that text in the parent object shall flow from bottom to top vertically, then from left to right horizontally on the page. This means that vertical lines are filled before the text expands horizontally.
lrTb (Left to Right, Top to Bottom)	Specifies that text in the parent object shall flow from left to right horizontally, then top to bottom vertically on the page. This means that horizontal lines are filled before the text expands vertically.
lrTbV (Left to Right, Top to Bottom Rotated)	Specifies that text in the parent object shall flow from left to right horizontally, then top to bottom vertically on the page. This means that horizontal lines are filled before the text expands vertically. This flow is also rotated such that any East Asian text shall be rotated 270 degrees when displayed on a page.
tbLrV (Top to Bottom, Left to Right Rotated)	Specifies that text in the parent object shall flow from top to bottom vertically, then left to right horizontally on the page. This means that vertical lines are filled before the text expands horizontally. This flow is also rotated such that all text is rotated 90 degrees when displayed on a page.
tbRl (Top to Bottom, Right to Left)	Specifies that text in the parent object shall flow from right to left horizontally, then top to bottom vertically on the page. This means that horizontal lines are filled before the text expands vertically.
tbRlV (Top to Bottom, Right to Left Rotated)	Specifies that text in the parent object shall flow from top to bottom vertically, then right to left horizontally on the page. This means that vertical lines are filled before the text expands horizontally.

Enumeration Value	Description
	This flow is also rotated such that all text is rotated 90 degrees when displayed on a page.

Referenced By
textDirection@val (§2.6.20); textDirection@val (§2.4.69); textDirection@val (§2.3.1.41)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextDirection">
   <restriction base="xsd:string">
      <enumeration value="lrTb"/>
      <enumeration value="tbRl"/>
      <enumeration value="btLr"/>
      <enumeration value="lrTbV"/>
      <enumeration value="tbRlV"/>
      <enumeration value="tbLrV"/>
   </restriction>
</simpleType>
```

2.18.101 ST_TextEffect (Animated Text Effects)

This simple type specifies the possible types of animated text effect which may be applied to a text run when it is displayed..

[*Example*: Consider a run of text which shall have an animated text effect consisting of multiple colored flashing lights. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:effect w:val="lights"/>
</w:rPr>
```

This run explicitly declares a type of text effect, using the val property, of `lights`, so the contents of this run will have the animated lights text effect. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
antsBlack (Black Dashed Line Animation)	Specifies that this text shall be surrounded by an animated black dashed line border.
antsRed (Marching Red Ants)	Specifies that this text shall be surrounded by an animated red dashed line border.
blinkBackground (Blinking Background Animation)	Specifies that this text shall be surrounded by a background color which alternates between black and

Enumeration Value	Description
	white.
lights (Colored Lights Animation)	Specifies that this text shall be surrounded by a border consisting of a series of colored lights, which constantly change colors in sequence.
none (No Animation)	Specifies that this text shall have no animated text effect.
shimmer (Shimmer Animation)	Specifies that this text shall be animated by alternating between normal and blurry states.
sparkle (Sparkling Lights Animation)	Specifies that this text shall have a background consisting of a random pattern of colored lights, which constantly change colors in sequence.

Referenced By
effect@val (§2.3.2.9)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextEffect">
   <restriction base="xsd:string">
      <enumeration value="blinkBackground"/>
      <enumeration value="lights"/>
      <enumeration value="antsBlack"/>
      <enumeration value="antsRed"/>
      <enumeration value="shimmer"/>
      <enumeration value="sparkle"/>
      <enumeration value="none"/>
   </restriction>
</simpleType>
```

2.18.102 ST_TextScale (Text Expansion/Compression Percentage)

This simple type specifies that the percentage by which the contents of a run shall be expanded or compressed with respect to its normal (100%) character width, with a minimum width of 1% and maximum width of 600%.

[*Example*: Consider a run of text which shall be compressed to 200% when displaying each character within the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:w w:val="50"/>
</w:rPr>
```

This run explicitly declares that the w value is 50, so the contents of this run will appear at 50% of their normal character width by compressing the width of each character. *end example*]

This simple type's contents are a restriction of the XML Schema integer datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 0.
- This simple type has a maximum value of less than or equal to 600.

Referenced By
w@val (§2.3.2.41)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextScale">
   <restriction base="xsd:integer">
      <minInclusive value="0"/>
      <maxInclusive value="600"/>
   </restriction>
</simpleType>
```

2.18.103 ST_Theme (Theme Font)

This simple type specifies a theme font type which may be referenced as a theme font within the parent run properties. This theme font is a reference to one of the predefined theme fonts, located in the document's Theme part,which allows for font information to be set centrally in the document.

[*Example*: Consider a run of ASCII text which shall be displayed using the `majorASCII` theme font. This requirement would be specified as follows in the resulting WordprocessingML:

```
<w:rPr>
  <w:rFonts w:asciiTheme=”majorAscii” />
</w:rPr>
```

The ascii attribute specifies that the run shall use the `majorAscii` theme font as defined in the document's themes part for all text in the ASCII range. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
majorAscii (Major ASCII Theme Font)	Specifies that the current font is a reference to the major theme font for the ASCII range.
majorBidi (Major Complex Script Theme Font)	Specifies that the current font is a reference to the major theme font for the Complex Script range.
majorEastAsia (Major East Asian Theme Font)	Specifies that the current font is a reference to the major theme font for the East Asian range.
majorHAnsi (Major High ANSI Theme Font)	Specifies that the current font is a reference to the major theme font for the High ANSI range.
minorAscii (Minor ASCII Theme Font)	Specifies that the current font is a reference to the

Enumeration Value	Description
	minor theme font for the ASCII range.
minorBidi (Minor Complex Script Theme Font)	Specifies that the current font is a reference to the minor theme font for the Complex Script range.
minorEastAsia (Minor East Asian Theme Font)	Specifies that the current font is a reference to the minor theme font for the East Asian range.
minorHAnsi (Minor High ANSI Theme Font)	Specifies that the current font is a reference to the minor theme font for the High ANSI range.

Referenced By
rFonts@asciiTheme (§2.3.2.24); rFonts@cstheme (§2.3.2.24); rFonts@eastAsiaTheme (§2.3.2.24); rFonts@hAnsiTheme (§2.3.2.24)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Theme">
   <restriction base="xsd:string">
      <enumeration value="majorEastAsia"/>
      <enumeration value="majorBidi"/>
      <enumeration value="majorAscii"/>
      <enumeration value="majorHAnsi"/>
      <enumeration value="minorEastAsia"/>
      <enumeration value="minorBidi"/>
      <enumeration value="minorAscii"/>
      <enumeration value="minorHAnsi"/>
   </restriction>
</simpleType>
```

2.18.104 ST_ThemeColor (Theme Color)

This simple type specifies a theme color to be applied to the current object. The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part, which allows color information to be set centrally in the document.

[*Example*: Consider a set of borders configured to use the `accent2` theme color, resulting in the following WordprocessingML markup:

```
<w:top … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />
<w:bottom … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />
<w:left … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />
<w:right … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />
```

The borders have a themeColor attribute of type ST_ThemeColor that when specified, imports the `accent2` theme color specified for this document. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
accent1 (Accent 1 Theme Color)	Specifies that the color to be used shall be the accent1 theme color.
accent2 (Accent 2 Theme Color)	Specifies that the color to be used shall be the accent2 theme color.
accent3 (Accent 3 Theme Color)	Specifies that the color to be used shall be the accent3 theme color.
accent4 (Accent 4 Theme Color)	Specifies that the color to be used shall be the accent4 theme color.
accent5 (Accent 5 Theme Color)	Specifies that the color to be used shall be the accent5 theme color.
accent6 (Accent 6 Theme Color)	Specifies that the color to be used shall be the accent6 theme color.
background1 (Background 1 Theme Color)	Specifies that the color to be used shall be the background1 theme color.
background2 (Background 2 Theme Color)	Specifies that the color to be used shall be the background2 theme color.
dark1 (Dark 1 Theme Color)	Specifies that the color to be used shall be the dark1 theme color.
dark2 (Dark 2 Theme Color)	Specifies that the color to be used shall be the dark2 theme color.
followedHyperlink (Followed Hyperlink Theme Color)	Specifies that the color to be used shall be the followedHyperlink theme color.
hyperlink (Hyperlink Theme Color)	Specifies that the color to be used shall be the hyperlink theme color.
light1 (Light 1 Theme Color)	Specifies that the color to be used shall be the light1 theme color.
light2 (Light 2 Theme Color)	Specifies that the color to be used shall be the light1 theme color.
none (No Theme Color)	Specifies that no theme color shall be applied to the current object.
text1 (Text 1 Theme Color)	Specifies that the color to be used shall be the text1 theme color.
text2 (Text 2 Theme Color)	Specifies that the color to be used shall be the text2 theme color.

Referenced By
background@themeColor (§2.2.1); bar@themeColor (§2.3.1.4); bdr@themeColor (§2.3.2.3);

Referenced By
between@themeColor (§2.3.1.5); bottom@themeColor (§2.6.2); bottom@themeColor (§2.4.3); bottom@themeColor (§2.4.4); bottom@themeColor (§2.15.2.4); bottom@themeColor (§2.3.1.7); color@themeColor (§2.3.2.5); color@themeColor (§2.15.2.5); insideH@themeColor (§2.4.17); insideH@themeColor (§2.4.18); insideV@themeColor (§2.4.19); insideV@themeColor (§2.4.20); left@themeColor (§2.15.2.21); left@themeColor (§2.4.24); left@themeColor (§2.6.7); left@themeColor (§2.3.1.17); left@themeColor (§2.4.27); right@themeColor (§2.3.1.28); right@themeColor (§2.4.30); right@themeColor (§2.6.15); right@themeColor (§2.15.2.35); right@themeColor (§2.4.32); shd@themeColor (§2.3.2.30); shd@themeColor (§2.4.33); shd@themeColor (§2.4.34); shd@themeColor (§2.4.35); shd@themeColor (§2.3.1.31); shd@themeFill (§2.3.2.30); shd@themeFill (§2.4.33); shd@themeFill (§2.4.34); shd@themeFill (§2.4.35); shd@themeFill (§2.3.1.31); tl2br@themeColor (§2.4.70); top@themeColor (§2.4.71); top@themeColor (§2.3.1.42); top@themeColor (§2.15.2.42); top@themeColor (§2.4.74); top@themeColor (§2.6.21); tr2bl@themeColor (§2.4.76); u@themeColor (§2.3.2.38)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ThemeColor">
   <restriction base="xsd:string">
      <enumeration value="dark1"/>
      <enumeration value="light1"/>
      <enumeration value="dark2"/>
      <enumeration value="light2"/>
      <enumeration value="accent1"/>
      <enumeration value="accent2"/>
      <enumeration value="accent3"/>
      <enumeration value="accent4"/>
      <enumeration value="accent5"/>
      <enumeration value="accent6"/>
      <enumeration value="hyperlink"/>
      <enumeration value="followedHyperlink"/>
      <enumeration value="none"/>
      <enumeration value="background1"/>
      <enumeration value="text1"/>
      <enumeration value="background2"/>
      <enumeration value="text2"/>
   </restriction>
</simpleType>
```

2.18.105 ST_TwipsMeasure (Measurement in Twentieths of a Point)

This simple type specifies that its contents will contain a positive whole number, whose contents consist of a measurement in twentieths of a point (equivalent to 1/1440th of an inch).

The contents of this measurement are interpreted based on the context of the parent XML element.

[*Example*: Consider an attribute value of `720` whose type is ST_TwipsMeasure. This attribute value specifies a size of one-half of an inch or 36 points (`720` twentieths of a point = 36 points = 0.5 inches). *end example*]

This simple type's contents are a restriction of the ST_UnsignedDecimalNumber simple type (§2.18.108).

Referenced By
col@space (§2.6.3); col@w (§2.6.3); cols@space (§2.6.4); defaultTabStop@val (§2.15.1.24); drawingGridHorizontalOrigin@val (§2.15.1.43); drawingGridHorizontalSpacing@val (§2.15.1.44); drawingGridVerticalOrigin@val (§2.15.1.45); drawingGridVerticalSpacing@val (§2.15.1.46); fitText@val (§2.3.2.12); framePr@h (§2.3.1.11); framePr@hSpace (§2.3.1.11); framePr@vSpace (§2.3.1.11); framePr@w (§2.3.1.11); gridCol@w (§2.4.12); hyphenationZone@val (§2.15.1.53); ind@firstLine (§2.3.1.12); ind@hanging (§2.3.1.12); legacy@legacySpace (§2.9.5); lnNumType@distance (§2.6.8); object@dxaOrig (§2.3.3.19); object@dyaOrig (§2.3.3.19); pgMar@footer (§2.6.11); pgMar@gutter (§2.6.11); pgMar@header (§2.6.11); pgMar@left (§2.6.11); pgMar@right (§2.6.11); pgSz@h (§2.6.13); pgSz@w (§2.6.13); spacing@after (§2.3.1.33); spacing@before (§2.3.1.33); tblpPr@bottomFromText (§2.4.54); tblpPr@leftFromText (§2.4.54); tblpPr@rightFromText (§2.4.54); tblpPr@topFromText (§2.4.54); trHeight@val (§2.4.77); w@val (§2.15.2.43)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TwipsMeasure">
   <restriction base="ST_UnsignedDecimalNumber"/>
</simpleType>
```

2.18.106 ST_UcharHexNumber (Two Digit Hexadecimal Number Value)

This simple type specifies a number value specified as a two digit (one octet) hexadecimal number), whose contents are interpreted based on the context of the parent XML element.

[*Example*: Consider the following value for a node of type ST_LongHexNumber: `BE`.

This value is valid, as it contains two hexadecimal digits, as an encoding of an octet of the actual decimal number value. *end example*]

This simple type's contents are a restriction of the XML Schema hexBinary datatype.

This simple type also specifies the following restrictions:

- This simple type's contents must have a length of exactly 1 characters.

Referenced By
background@themeShade (§2.2.1); background@themeTint (§2.2.1); bar@themeShade (§2.3.1.4); bar@themeTint (§2.3.1.4); bdr@themeShade (§2.3.2.3); bdr@themeTint (§2.3.2.3); between@themeShade (§2.3.1.5); between@themeTint (§2.3.1.5); bottom@themeShade (§2.6.2); bottom@themeShade (§2.4.3); bottom@themeShade (§2.4.4); bottom@themeShade (§2.15.2.4); bottom@themeShade (§2.3.1.7); bottom@themeTint (§2.6.2); bottom@themeTint (§2.4.3); bottom@themeTint (§2.4.4); bottom@themeTint (§2.15.2.4); bottom@themeTint (§2.3.1.7); charset@val (§2.8.2.2); color@themeShade (§2.3.2.5); color@themeShade (§2.15.2.5); color@themeTint (§2.3.2.5); color@themeTint (§2.15.2.5); insideH@themeShade (§2.4.17); insideH@themeShade (§2.4.18); insideH@themeTint (§2.4.17); insideH@themeTint (§2.4.18); insideV@themeShade (§2.4.19); insideV@themeShade (§2.4.20); insideV@themeTint (§2.4.19); insideV@themeTint (§2.4.20); left@themeShade (§2.15.2.21); left@themeShade (§2.4.24); left@themeShade (§2.6.7); left@themeShade (§2.3.1.17); left@themeShade (§2.4.27); left@themeTint (§2.15.2.21); left@themeTint (§2.4.24); left@themeTint (§2.6.7); left@themeTint

Referenced By
(§2.3.1.17); left@themeTint (§2.4.27); right@themeShade (§2.3.1.28); right@themeShade (§2.4.30); right@themeShade (§2.6.15); right@themeShade (§2.15.2.35); right@themeShade (§2.4.32); right@themeTint (§2.3.1.28); right@themeTint (§2.4.30); right@themeTint (§2.6.15); right@themeTint (§2.15.2.35); right@themeTint (§2.4.32); shd@themeFillShade (§2.3.2.30); shd@themeFillShade (§2.4.33); shd@themeFillShade (§2.4.34); shd@themeFillShade (§2.4.35); shd@themeFillShade (§2.3.1.31); shd@themeFillTint (§2.3.2.30); shd@themeFillTint (§2.4.33); shd@themeFillTint (§2.4.34); shd@themeFillTint (§2.4.35); shd@themeFillTint (§2.3.1.31); shd@themeShade (§2.3.2.30); shd@themeShade (§2.4.33); shd@themeShade (§2.4.34); shd@themeShade (§2.4.35); shd@themeShade (§2.3.1.31); shd@themeTint (§2.3.2.30); shd@themeTint (§2.4.33); shd@themeTint (§2.4.34); shd@themeTint (§2.4.35); shd@themeTint (§2.3.1.31); tl2br@themeShade (§2.4.70); tl2br@themeTint (§2.4.70); top@themeShade (§2.4.71); top@themeShade (§2.3.1.42); top@themeShade (§2.15.2.42); top@themeShade (§2.4.74); top@themeShade (§2.6.21); top@themeTint (§2.4.71); top@themeTint (§2.3.1.42); top@themeTint (§2.15.2.42); top@themeTint (§2.4.74); top@themeTint (§2.6.21); tr2bl@themeShade (§2.4.76); tr2bl@themeTint (§2.4.76); u@themeShade (§2.3.2.38); u@themeTint (§2.3.2.38)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_UcharHexNumber">
   <restriction base="xsd:hexBinary">
      <length value="1"/>
   </restriction>
</simpleType>
```

2.18.107 ST_Underline (Underline Patterns)

This simple type specifies the types of patterns which may be used to create the underline applied beneath the text in a run.

[*Example*: Consider a run of text which shall have a double underline explicitly turned on for the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:u w:val="double"/>
</w:rPr>
```

The val of the underline on this run is `double`, so the style of the underline on this run shall be a double line. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
dash (Dashed Underline)	Specifies an underline consisting of a dashed line beneath all characters in this run.

Enumeration Value	Description
	[*Example*: Underline text. *end example*]
dashDotDotHeavy (Thick Dash-Dot-Dot Underline)	Specifies an underline consisting of a series of thick dash, dot, dot characters beneath all characters in this run. [*Example*: Underline text. *end example*]
dashDotHeavy (Thick Dash-Dot Underline)	Specifies an underline consisting of a series of thick dash, dot characters beneath all characters in this run. [*Example*: Underline text. *end example*]
dashedHeavy (Thick Dashed Underline)	Specifies an underline consisting of a series of thick dashes beneath all characters in this run. [*Example*: Underline text. *end example*]
dashLong (Long Dashed Underline)	Specifies an underline consisting of long dashed characters beneath all characters in this run. [*Example*: Underline text. *end example*]
dashLongHeavy (Thick Long Dashed Underline)	Specifies an underline consisting of thick long dashed characters beneath all characters in this run. [*Example*:

Enumeration Value	Description
	Underline text. *end example*]
dotDash (Dash-Dot Underline)	Specifies an underline consisting of a series of dash, dot characters beneath all characters in this run. [*Example*: Underline text. *end example*]
dotDotDash (Dash-Dot-Dot Underline)	Specifies an underline consisting of a series of dash, dot, dot characters beneath all characters in this run. [*Example*: Underline text. *end example*]
dotted (Dotted Underline)	Specifies an underline consisting of a series of dot characters beneath all characters in this run. [*Example*: Underline text. *end example*]
dottedHeavy (Thick Dotted Underline)	Specifies an underline consisting of a series of thick dot characters beneath all characters in this run. [*Example*: Underline text. *end example*]
double (Double Underline)	Specifies an underline consisting of two lines beneath all characters in this run. [*Example*: Underline text. *end example*]

Enumeration Value	Description
none (No Underline)	Specifies no underline beneath this run. [*Example*: Underline text. *end example*]
single (Single Underline)	Specifies an underline consisting of a single line beneath all characters in this run. [*Example*: Underline text. *end example*]
thick (Thick Underline)	Specifies an underline consisting of a single thick line beneath all characters in this run. [*Example*: Underline text. *end example*]
wave (Wave Underline)	Specifies an underline consisting of a single wavy line beneath all characters in this run. [*Example*: Underline text. *end example*]
wavyDouble (Double Wave Underline)	Specifies an underline consisting of a pair of wavy lines beneath all characters in this run. [*Example*: Underline text. *end example*]
wavyHeavy (Heavy Wave Underline)	Specifies an underline consisting of a single thick wavy line beneath all characters in this run.

Enumeration Value	Description
	[*Example*: Underline text. *end example*]
words (Underline Non-Space Characters Only)	Specifies an underline consisting of a single line beneath all non-space characters in the run. There shall be no underline beneath any space character (breaking or non-breaking). [*Example*: Underline text. *end example*]

Referenced By
u@val (§2.3.2.38)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Underline">
   <restriction base="xsd:string">
      <enumeration value="single"/>
      <enumeration value="words"/>
      <enumeration value="double"/>
      <enumeration value="thick"/>
      <enumeration value="dotted"/>
      <enumeration value="dottedHeavy"/>
      <enumeration value="dash"/>
      <enumeration value="dashedHeavy"/>
      <enumeration value="dashLong"/>
      <enumeration value="dashLongHeavy"/>
      <enumeration value="dotDash"/>
      <enumeration value="dashDotHeavy"/>
      <enumeration value="dotDotDash"/>
      <enumeration value="dashDotDotHeavy"/>
      <enumeration value="wave"/>
      <enumeration value="wavyHeavy"/>
      <enumeration value="wavyDouble"/>
      <enumeration value="none"/>
   </restriction>
</simpleType>
```

2.18.108 ST_UnsignedDecimalNumber (Unsigned Decimal Number Value)

This simple type specifies that its contents will contain a positive whole decimal number, whose contents are interpreted based on the context of the parent XML element.

[*Example*: Consider the following WordprocessingML fragment:

```
<w:pPr>
  <w:divId w:val="1512645511" />
</w:pPr>
```

The value of the val attribute is the ID of the associated HTML `div`.

However, consider the following fragment:

```
<w:ilvl w:val="1">
  ...
</w:ilvl>
```

In this case, the decimal number in the val attribute is the ID of the associated numbering level. In each case, the decimal number value is interpreted in the context of the parent element. *end example*]

This simple type's contents are a restriction of the XML Schema unsignedLong datatype.

Referenced By
ST_EighthPointMeasure (§2.18.27); ST_HpsMeasure (§2.18.48); ST_PixelsMeasure (§2.18.74); ST_PointMeasure (§2.18.75); ST_TwipsMeasure (§2.18.105)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_UnsignedDecimalNumber">
   <restriction base="xsd:unsignedLong"/>
</simpleType>
```

2.18.109 ST_VAnchor (Vertical Anchor Location)

This simple type specifies the vertical position to which the parent object has been anchored in the document. This anchor position shall be used as the base location to determine the final vertical position of the object in the document.

[*Example*: Consider a text frame which should be positioned one inch to the right of its column in a left-to-right document. This text frame would be specified using the following WordprocessingML:

```
<w:pPr>
  <w:framePr … w:y="1440" w:vAnchor="page" />
</w:pPr>
```

These frame vertical anchor properties specify that they are relative to the anchor paragraph's page. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
margin (Relative To Margin)	Specifies that the parent object shall be vertically anchored to the text margins. This shall be used to specify that any vertical positioning values shall be calculated with respect to the location of the text margin.
page (Relative To Page)	Specifies that the parent object shall be vertically anchored to the page edge. This shall be used to specify that any vertical positioning values shall be calculated with respect to the location of the edge of the page.
text (Relative To Vertical Text Extents)	Specifies that the parent object shall be vertically anchored to the text extents. This shall be used to specify that any vertical positioning values shall be calculated with respect to the location of the top edge of the text in the anchor paragraph.

Referenced By
framePr@vAnchor (§2.3.1.11); tblpPr@vertAnchor (§2.4.54)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_VAnchor">
   <restriction base="xsd:string">
      <enumeration value="text"/>
      <enumeration value="margin"/>
      <enumeration value="page"/>
   </restriction>
</simpleType>
```

2.18.110 ST_VerticalAlignRun (Vertical Positioning Location)

This simple type specifies possible values for the alignment of the contents of this run in relation to the default appearance of the run's text. This allows the text to be repositioned as subscript or superscript without altering the font size of the run properties.

[*Example*: Consider a run which shall be positioning as superscript when displaying its contents. This requirement would be specified using the following WordprocessingML:

```
<w:rPr>
  <w:vertAlign w:val="superscript" />
</w:rPr>
```

The resulting run is positioned as `superscript`, therefore it is rendered in a smaller size above the default baseline location for the contents of the run. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
baseline (Regular Vertical Positioning)	Specifies that the text in the parent run shall be located at the baseline and presented in the same size as surrounding text.
subscript (Subscript)	Specifies that this text should be subscript. This setting shall lower the text in this run below the baseline and change it to a smaller size, if a smaller size is available.
superscript (Superscript)	Specifies that this text should be superscript. This setting shall raise the text in this run above the baseline and change it to a smaller size, if a smaller size is available.

Referenced By
vertAlign@val (§2.3.2.40)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_VerticalAlignRun">
   <restriction base="xsd:string">
      <enumeration value="baseline"/>
      <enumeration value="superscript"/>
      <enumeration value="subscript"/>
   </restriction>
</simpleType>
```

2.18.111 ST_VerticalJc (Vertical Alignment Type)

This simple type specifies the vertical alignment for text between the top and bottom margins of the parent container (page or table cell).

[*Example*: Consider a region where the text shall be vertically centered in the parent element. This would require a val value of `center`, in order to specify that all justification vertically shall be centered relative to the parent. For a section, this setting would be specified as follows:

```
<w:vAlign w:val="center" />
```

The val attribute of `center` specifies that the content is centered relative to its container . *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
both (Vertical Justification)	Specifies that the text shall be vertically justified between the top and bottom margins of the parent object, by adding additional line spacing to each paragraph as required. This setting is only applied for the content of the section which is displayed on full pages. If the content does not use the full page (e.g. another section begins on the same page, or the document ends mid-page), then the value shall be ignored when rendering that page (returning to the default value of `top`) This value is only valid for page justification settings, and shall be ignored when specified on a table cell (returning to the default value of `top`).
bottom (Align Bottom)	Specifies that the text shall be vertically aligned to the bottom margin of the parent object, by moving all text to the bottom text extent within the parent object as required.
center (Align Center)	Specifies that the text shall be vertically aligned to the center of the parent object..
top (Align Top)	Specifies that the text shall be vertically aligned to the top margin of the parent object, by moving all text to the top text extent within the parent object as required.

Referenced By
vAlign@val (§2.4.80); vAlign@val (§2.6.23)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_VerticalJc">
   <restriction base="xsd:string">
      <enumeration value="top"/>
      <enumeration value="center"/>
      <enumeration value="both"/>
      <enumeration value="bottom"/>
   </restriction>
</simpleType>
```

2.18.112 ST_View (Document View Values)

This simple type defines the possible views which may be used to determine how WordprocessingML documents may be rendered when displayed by an application.

[*Example*: Consider a WordprocessingML document that shall be displayed on the screen in the same form as it will be printed. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:view w:val="print" />
```

The view element's val attribute is equal to `print` specifying that the given WordprocessingML document shall be rendered as it will be printed. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
masterPages (Master Document View)	Specifies that a given WordprocessingML document shall be rendered in a view optimized for outlining or creating long documents. [*Note*: This setting may be interpreted as functionally equivalent to the `outline` setting, as it only remains separate to support legacy applications. *end note*]
none (Default View)	Specifies that a given WordprocessingML document shall be rendered in the default view of the application.
normal (Draft View)	Specifies that a given WordprocessingML document shall be rendered in a view optimized for outlining or creating long documents.
outline (Outline View)	Specifies that a given WordprocessingML document shall be rendered in a view optimized for outlining or creating long documents.
print (Print Layout View)	Specifies that this document shall be opened in a view that displays the document as it will print.

Enumeration Value	Description
web (Web Page View)	Specifies that a given WordprocessingML document shall be rendered in a view mimicking the way this document would be displayed in a web page.

Referenced By
view@val (§2.15.1.93)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_View">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="print"/>
      <enumeration value="outline"/>
      <enumeration value="masterPages"/>
      <enumeration value="normal"/>
      <enumeration value="web"/>
   </restriction>
</simpleType>
```

2.18.113 ST_Wrap (Text Wrapping around Text Frame Type)

This simple type specifies the type of text wrapping which shall be allowed around a text frame within a document.

[*Example*: Consider the following WordprocessingML fragment specifying a text frame:

```
<w:p>
  <w:pPr>
    <w:framePr w:w="2419" w:h="2189" w:hRule="atLeast" w:hSpace="187"
w:wrap="around" w:vAnchor="text" w:hAnchor="page" w:x="1643" w:y="73" />
  </w:pPr>
  <w:r>
    <w:t>Text Frame Content.</w:t>
  </w:r>
</w:p>
```

This wrap attribute on this text frame specifies that when the frame is rendered on the page, any non-text frame paragraphs which would normally flow onto the same lines shall be allowed to wrap around it. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description

Enumeration Value	Description
around (Allow Text Wrapping Around Frame)	Specifies that text shall be allowed to wrap around the remaining space on each line around this text frame in the document.
auto (Default Text Wrapping Around Frame)	Specifies that text shall have the default application-defined behavior of the application displaying the WordprocessingML document with regard to the text wrapping displayed around the frame.
none (No Text Wrapping Around Frame)	Specifies that text shall not be allowed to wrap around the remaining space on each lines around this text frame. Any text content shall therefore be placed on the next line following this text frame which does not intersect with the frame's extents.
notBeside (No Text Wrapping Beside Frame)	Specifies that text shall not be allowed to wrap around the remaining space on each lines around this text frame. Any text content shall therefore be placed on the next line following this text frame which does not intersect with the frame's extents.
through (Through Text Wrapping Around Frame)	Specifies that text shall be allowed to wrap around the remaining space on each line around this text frame in the document.
tight (Tight Text Wrapping Around Frame)	Specifies that text shall be allowed to tightly wrap around the remaining space on each line around this text frame in the document.

Referenced By
framePr@wrap (§2.3.1.11)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Wrap">
   <restriction base="xsd:string">
      <enumeration value="auto"/>
      <enumeration value="notBeside"/>
      <enumeration value="around"/>
      <enumeration value="tight"/>
      <enumeration value="through"/>
      <enumeration value="none"/>
   </restriction>
</simpleType>
```

2.18.114 ST_XAlign (Horizontal Alignment Location)

This simple type specifies the set of possible relative horizontal positions for the parent floating object. This relative position is specified relative to the horizontal anchor specified by the parent object.

[*Example*: Consider the following WordprocessingML fragment specifying a text frame:

```
<w:p>
  <w:pPr>
    <w:framePr w:w="2419" w:h="2189" w:hRule="atLeast" w:hSpace="187"
w:wrap="around" w:vAnchor="text" w:hAnchor="page" w:xAlign="left" w:y="73" />
  </w:pPr>
  <w:r>
    <w:t>Text Frame Content.</w:t>
  </w:r>
</w:p>
```

This text frame specifies by the presence of the xAlign attribute to align the frame on the `left` side of the anchor object, in this case, the page. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
center (Centered Horizontally)	Specifies that the parent object shall be centered with respect to the anchor settings. [*Example*: Centered on the page horizontally. *end example*]
inside (Inside)	Specifies that the parent object shall be inside of the anchor object. [*Example*: Inside the text margin horizontally. *end example*]
left (Left Aligned Horizontally)	Specifies that the parent object shall be left aligned with respect to the anchor settings. [*Example*: Left aligned on the page horizontally. *end example*]
outside (Outside)	Specifies that the parent object shall be outside of the anchor object. [*Example*: Outside the text margin horizontally. *end example*]
right (Right Aligned Horizontally)	Specifies that the parent object shall be right aligned

Enumeration Value	Description
	with respect to the anchor settings. [*Example*: Right aligned on the page horizontally. *end example*]

Referenced By
framePr@xAlign (§2.3.1.11); tblpPr@tblpXSpec (§2.4.54)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_XAlign">
   <restriction base="xsd:string">
      <enumeration value="left"/>
      <enumeration value="center"/>
      <enumeration value="right"/>
      <enumeration value="inside"/>
      <enumeration value="outside"/>
   </restriction>
</simpleType>
```

2.18.115 ST_YAlign (Vertical Alignment Location)

This simple type specifies the set of possible relative vertical positions for the parent floating object. This relative position is specified relative to the vertical anchor specified by the parent object.

[*Example*: Consider the following WordprocessingML fragment specifying a text frame:

```
<w:p>
  <w:pPr>
    <w:framePr w:w="2419" w:h="2189" w:hRule="atLeast" w:hSpace="187"
w:wrap="around" w:vAnchor="margin" w:hAnchor="page" w:x="1643" w:y="73"
w:yAlign="center" />
  </w:pPr>
  <w:r>
    <w:t>Text Frame Content.</w:t>
  </w:r>
</w:p>
```

This text frame specifies by the presence of the yAlign attribute to vertically align the frame in the `center` of the anchor object, in this case, the `margin`. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description

Enumeration Value	Description
bottom (Bottom)	Specifies that the parent object shall be vertically aligned to the bottom edge of the anchor object . [*Example*: At the bottom of the current paragraph. *end example*]
center (Centered Vertically)	Specifies that the parent object shall be vertically centered with respect to the anchor object. [*Example*: Centered on the page vertically. *end example*]
inline (In line With Text)	Specifies that the parent object shall be vertically aligned in line with the surrounding text (i.e. shall not allow any text wrapping around it when positioned in the document.
inside (Inside Anchor Extents)	Specifies that the parent object shall be vertically aligned to the edge of the anchor object, and positioned inside that object. [*Example*: Inside the text margins vertically. *end example*]
outside (Outside Anchor Extents)	Specifies that the parent object shall be vertically aligned to the edge of the anchor object, and positioned outside that object. [*Example*: Outside the text margins vertically. *end example*]
top (Top)	Specifies that the parent object shall be vertically aligned to the top edge of the anchor object . [*Example*: At the top of the current paragraph. *end example*]

Referenced By
framePr@yAlign (§2.3.1.11); tblpPr@tblpYSpec (§2.4.54)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_YAlign">
   <restriction base="xsd:string">
      <enumeration value="inline"/>
      <enumeration value="top"/>
      <enumeration value="center"/>
      <enumeration value="bottom"/>
      <enumeration value="inside"/>
      <enumeration value="outside"/>
   </restriction>
</simpleType>
```

2.18.116 ST_Zoom (Magnification Preset Values)

This simple type specifies the type of magnification settings which may be applied to a given document on open.

[*Example*: Consider a WordprocessingML document that should be visible without any horizontal scrolling when it is displayed. This requirement would be specified using the following WordprocessingML:

```
<w:zoom w:val="bestFit" w:percent="90" />
```

The val attribute is equal to the value `bestFit` specifying that an application shall dynamically calculate the magnification needed such that the given document shall be visible on the horizontal plane of the document with no horizontal scrolling required to see any part of the WordprocessingML document's pages. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
bestFit (Display Page Width)	Specifies that the magnification setting shall be adjusted to ensure the width of the current page matches the available window width.
fullPage (Display One Full Page)	Specifies that the magnification setting shall be adjusted to ensure that one full page can be seen at a time.
none (No Preset Magnification)	Specifies that no preset magnification is present, and the last known cached setting shall be used.
textFit (Display Text Width)	Specifies that the magnification setting shall be adjusted to ensure the width of the text extents on the current page matches the available window width.

Referenced By
zoom@val (§2.15.1.95)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Zoom">
    <restriction base="xsd:string">
        <enumeration value="none"/>
        <enumeration value="fullPage"/>
        <enumeration value="bestFit"/>
        <enumeration value="textFit"/>
    </restriction>
</simpleType>
```

3. SpreadsheetML Reference Material

The subordinate subclauses specify the semantics for the XML markup comprising a SpreadsheetML document, as defined by Part 1 of this Office Open XML Standard.

3.1 Table of Contents

This subclause is informative.

End of informative text.

3.2 Workbook

This subclause describes the elements and simple types that comprise the workbook main definition.

A workbook is composed of book-level properties and a collection of 1 or more sheets. The sheets are the central working surface for a spreadsheet application. The workbook part and corresponding properties comprise data used to set application and workbook-level operational state. The workbook also serves to bind all the sheets and child objects into an organized single file. The workbook properties include information about what application last saved the file, where and how the windows of the workbook were positioned, and an enumeration of the worksheets in the workbook.

It is important for the sake of simplicity to minimize the *required* set of workbook properties that must be present to compose a valid workbook. Therefore these are the required data points for the smallest possible (blank) workbook:

```
<workbook>
  <sheets>
    <sheet name="Sheet1" sheetId="1" r:id="rId1"/>
  </sheets>
</workbook>
```

Note that this workbook has a single sheet, named Sheet1. An Id for the sheet is required, and a relationship Id pointing to the location of the sheet definition is also required. Sheet1 itself will be discussed in a separate section.

3.2.1 bookViews (Workbook Views)

This element specifies the collection of workbook views. Each view can specifies a window position, filter options, and other configurations. There is no limit on the number of views that can be defined for a workbook.

[*Example*:

```
<bookViews>
  <workbookView showHorizontalScroll="0" showVerticalScroll="0"
    showSheetTabs="0" xWindow="120" yWindow="45" windowWidth="15135"
    windowHeight="8130" activeTab="2" autoFilterDateGrouping="0"/>
</bookViews>
```

end example]

Parent Elements
workbook (§3.2.27)

Child Elements	Subclause
workbookView (Workbook View)	§3.2.30

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BookViews">
   <sequence>
      <element name="workbookView" type="CT_BookView" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.2.2 calcPr (Calculation Properties)

This element defines the collection of properties the application uses to record calculation status and details. Calculation is the process of computing formulas and then displaying the results as values in the cells that contain the formulas.

[*Example*:

```
<calcPr calcId="122211" calcMode="auto" refMode="R1C1" iterate="1"
  fullPrecision="0"/>
```

end example]

Parent Elements
workbook (§3.2.27)

Attributes	Description

Attributes	Description
calcCompleted (Calc Completed)	Specifies a boolean value that determines whether workbook data was recalculated before the workbook was saved. A value of on, 1, or true indicates recalculation was completed before save. A value of off, 0, or false indicates that recalculation was not completed before save. The default value for this attribute is true. The possible values for this attribute are defined by the XML Schema boolean datatype.
calcId (Calculation Id)	Specifies the version of the calculation engine used to calculate values in the workbook. When you open a workbook created in the current version, the application recalculates only the formulas that depend on cells that have changed. When you use open a workbook that was created in a earlier version of the application, all the formulas in the workbook— those that depend on cells that have changed and those that do not— are recalculated. This ensures that the workbook is fully optimized for the current application version. The value for `calcID` depends on the application. SpreadsheetML defaults form [version][build], where [version] refers to the version of the application, and [build] refers to the build of the application when the calculation engine changed. [*Example*: `<calcPr calcId="122211"/>` *end example*] The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
calcMode (Calculation Mode)	Specifies when the application should calculate formulas in the workbook. The default value for this attribute is "auto." The possible values for this attribute are defined by the ST_CalcMode simple type (§3.18.4).
calcOnSave (Calculate On Save)	Specifies a boolean value that indicates whether the application will recalculate values when the workbook is saved. A value of on, 1, or true indicates recalculation will be performed when the workbook is saved. A value of off, 0, or false indicates recalculation will not be performed when the workbook is saved.

Attributes	Description
	The default value for this attribute is true. The possible values for this attribute are defined by the XML Schema boolean datatype.
concurrentCalc (Concurrent Calculations)	Specifies a boolean value that indicates whether concurrent calculation processes are enabled for this workbook. A value of on, 1, or true indicates concurrent calculations are enabled in this workbook. A value of off, 0, or false indicates concurrent calculations are not enabled. The default value for this attribute is true. The possible values for this attribute are defined by the XML Schema boolean datatype.
concurrentManualCount (Concurrent Thread Manual Count)	Specifies the count of concurrent calculation processes manually set by the user. If omitted, the count is set automatically by the application. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
forceFullCalc (Force Full Calculation)	Specifies a boolean value that indicates whether the application will perform a full recalculation when one was not indicated by other calculation properties. This attribute allows the application to expose mechanisms in the user interface that give users the ability to trigger when full recalculations take place. A value of on, 1, or true indicates the application will perform a full recalculation of workbook. A value of off, 0, or false indicates the application will not perform a full recalculation when the workbook. The possible values for this attribute are defined by the XML Schema boolean datatype.
fullCalcOnLoad (Full Calculation On Load)	Specifies a boolean value that indicates whether the application shall perform a full recalculation when the workbook is opened. After load and successful calculation, the application should set this value to false. The application should set this value to true when cell formulas or values are modified by another process while the application has the workbook opened. A value of on, 1, or true indicates the application will perform a full recalculation of workbook values when the workbook is opened. A value of off, 0, or false indicates the application will not perform a full recalculation when the workbook is opened. Note: If manual calcMode is `true`, then a full recalculation will not be performed on load, even when this attribute is set.

Attributes	Description
	The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.
fullPrecision (Full Precision Calculation)	Specifies a boolean that indicates the precision the application will use when performing calculations in the workbook. Full precision means that the application uses the entire value(s) stored in cells referenced by the formula to perform the calculation. For example, if two cells each contain the value 10.005 and the cells are formatted to display values in currency format, the value $10.01 is displayed in each cell. If you add the two cells together, the result is $20.01 because the application adds the stored values 10.005 and 10.005, not the displayed values. You can change the precision of calculations so that the application uses the displayed value instead of the stored value when it recalculates formulas. For the above example, if fullPrecision is `false`, then the result shall be $20.02, because each cell shows $10.01, so those are the values to be added. Furthermore, when fullPrecision is `false`, the calculated value as displayed shall be saved to file. A value of on, 1, or true indicates the application uses the stored values of the referenced cells when performing calculations. A value of off, 0, or false indicates the application uses the display values of the referenced cells when performing calculations. The possible values for this attribute are defined by the XML Schema boolean datatype.
iterate (Calculation Iteration)	Specifies a boolean value that indicates whether the application should attempt to calculate formulas that contain circular references. A circular reference is a formula that refers to the cell— either directly or indirectly— that contains the formula. If a formula refers back to one of its own cells, you must determine how many times the formula should recalculate. A value of on, 1, or true indicates the application should attempt to calculate circular references. The calculation engine will perform iterative `iterateCount` calculations to before stopping. A value of off, 0, or false indicates that the application should not attempt to calculate formulas with circular references. The calculation engine will stop on the first iteration when it encounters a circular references. The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.
iterateCount (Iteration Count)	Specifies the number of iterations the calculation engine will attempt when calculating a workbook with circular references, when `iterate` is true. The default value for this attribute is 100.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
iterateDelta (Iterative Calculation Delta)	Specifies a double that contains the maximum change for iterative calculations. The application stops calculating after iterateCount iterations or after all values in the circular reference change by less than iterateDelta between iterations, whichever comes first. The default value for this attribute is "0.001" The possible values for this attribute are defined by the XML Schema double datatype.
refMode (Reference Mode)	Specifies the reference style for this workbook. Instead of using letters for columns and numbers for rows ("A1"), this options enables using numbers for both rows and columns. Cells are then referred to in this format: R1C1. The default value for this attribute is "A1." The possible values for this attribute are defined by the ST_RefMode simple type (§3.18.66).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CalcPr">
    <attribute name="calcId" type="xsd:unsignedInt"/>
    <attribute name="calcMode" type="ST_CalcMode" use="optional" default="auto"/>
    <attribute name="fullCalcOnLoad" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="refMode" type="ST_RefMode" use="optional" default="A1"/>
    <attribute name="iterate" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="iterateCount" type="xsd:unsignedInt" use="optional" default="100"/>
    <attribute name="iterateDelta" type="xsd:double" use="optional" default="0.001"/>
    <attribute name="fullPrecision" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="calcCompleted" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="calcOnSave" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="concurrentCalc" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="concurrentManualCount" type="xsd:unsignedInt" use="optional"/>
    <attribute name="forceFullCalc" type="xsd:boolean" use="optional"/>
</complexType>
```

3.2.3 customWorkbookView (Custom Workbook View)

This element specifies a single custom workbook view. A custom workbook view consists of a set of display and print settings that you can name and apply to a workbook. You can create more than one view of the same workbook without saving separate copies of the workbook. Custom workbook views are created by the end-user via tools in the application user interface.

Parent Elements
customWorkbookViews (§3.2.4)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10

Attributes	Description
activeSheetId (Active Sheet in Book View)	Specifies the sheetId of a sheet in the workbook that is the active sheet in this book view. Correpsonds to a sheetId of a sheet in the sheets collection. This attribute is required. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
autoUpdate (Auto Update)	Specifies a boolean value that indicates that this application will automatically update changes at the interval specified by the `mergeInterval` attribute. This is only applicable for shared workbooks. A value of on, 1, or true indicates the application will update changes at the interval specified in the `mergeInterval` attribute. A value of off, 0, or false indicates the application will update changes whenever the workbook is saved by the user. The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.
changesSavedWin (Changes Saved Win)	Specifies a boolean value that indicates that when conflicts are found, the changes being saved always take precedence. This is only applicable for shared workbooks in automatic refresh mode. A value of on, 1, or true indicates that changes being saved take precedence when conflicts in data are found in a shared workbook. A value of off, 0, or false indicates that changes being saved do not take precedence over other changes if conflicts are found in a shared workbook. The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.
guid (Custom View GUID)	Specifies a globally unique identifier (GUID) for this custom view The possible values for this attribute are defined by the ST_Guid simple type (§3.18.41).
includeHiddenRowCol (Include Hidden	Specifies a boolean value that indicates whether to include hidden rows, columns, and filter settings in this custom view.

Attributes	Description
Rows & Columns)	A value of on, 1, or true indicates that hidden rows, columns, and filter settings are included in this custom view. A value of off, 0, or false indicates that hidden rows, columns, and filter settings are not included. The default value for this attribute is true. The possible values for this attribute are defined by the XML Schema boolean datatype.
includePrintSettings (Include Print Settings)	Specifies a boolean value that indicates whether to include print settings in this custom view. A value of on, 1, or true indicates that print settings are included in this custom view. A value of off, 0, or false indicates print settings are not included in this custom view. The default value for this attribute is true. The possible values for this attribute are defined by the XML Schema boolean datatype.
maximized (Maximized)	Specifies a boolean value that indicates whether the book window is maximized. A value of on, 1, or true indicates the workbook window is maximized. A value of off, 0, or false indicates the workbook window is not maximized. The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.
mergeInterval (Merge Interval)	Automatic update interval (in minutes). Only applicable for shared workbooks in automatic refresh mode. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
minimized (Minimized)	Specifies a boolean value that indicates whether the workbook window is minimized. A value of on, 1, or true indicates the workbook window is minimized. A value of off, 0, or false indicates the workbook window is not minimized. The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.
name (Custom View	Specifies the name of the custom view.

Attributes	Description
Name)	This attribute is required. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
onlySync (Only Synch)	Specifies a boolean value that indicates, during automatic refresh, the current user's changes will not be saved. The workbook will only be updated with other users' changes. Only applicable for shared workbooks in automatic refresh mode. A value of on, 1, or true indicates the current user's changes will not be saved during automatic refresh. A value of off, 0, or false indicates the current user's will be saved during automatic refresh. The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.
personalView (Personal View)	Specifies a boolean value that indicates that this custom view is a personal view for a shared workbook user. Only applicable for shared workbooks. Personal views allow each user of a shared workbook to store their individual print and filter settings. A value of on, 1, or true indicates this custom view is a personal view for a shared workbook user. A value of off, 0, or false indicates this view is not a personal view. The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.
showComments (Show Comments)	Specifies how comments are displayed in this custom view The possible values for this attribute are defined by the ST_Comments simple type (§3.18.15).
showFormulaBar (Show Formula Bar)	Specifies a boolean value that indicates whether to display the formula bar in the application user interface. A value of on, 1, or true indicates the formula bar is shown in the user interface. A value of off, 0, or false indicates the formula bar is not shown in the user interface. The default value for this attribute is true. The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description
showHorizontalScroll (Show Horizontal Scroll)	Specifies a boolean value that indicates whether to display the horizontal scroll bar in the user interface. A value of on, 1, or true indicates that the horizontal scrollbar is shown. A value of off, 0, or false indicates that the horizontal scrollbar is not shown. The default value for this attribute is true. The possible values for this attribute are defined by the XML Schema boolean datatype.
showObjects (Show Objects)	Specifies how objects are displayed in this custom view. The default value for this attribute is "all." The possible values for this attribute are defined by the ST_Objects simple type (§3.18.50).
showSheetTabs (Show Sheet Tabs)	Specifies a boolean value that indicates whether to display the sheet tabs in the user interface. A value of on, 1, or true indicates that sheet tabs shall be shown. A value of off, 0, or false indicates that sheet tabs shall not be shown. The default value for this attribute is true. The possible values for this attribute are defined by the XML Schema boolean datatype.
showStatusbar (Show Status Bar)	Specifies a boolean value that indicates whether to display the status bar in the user interface. A value of on, 1, or true indicates that the status bar is shown. A value of off, 0, or false indicates the status bar is not shown. The default value for this attribute is true. The possible values for this attribute are defined by the XML Schema boolean datatype.
showVerticalScroll (Show Vertical Scroll)	Specifies a boolean value that indicates whether to display the vertical scroll bar. A value of on, 1, or true indicates the vertical scrollbar shall be shown. A value of off, 0, or false indicates the vertical scrollbar shall not be shown. The default value for this attribute is true. The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description
tabRatio (Sheet Tab Ratio)	Specified the ratio between the workbook tabs bar and the horizontal scroll bar. `tabRatio` is assumed to be out of 1000 of the horizontal window width. The default value for this attribute is 600. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
windowHeight (Window Height)	Specifies the height of the workbook window. The unit of measurement for this value is twips. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
windowWidth (Window Width)	Specifies the width of the workbook window. The unit of measurement for this value is twips. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
xWindow (Top Left Corner (X Coordinate))	Specifies the X coordinate for the upper left corner of the book window. The unit of measurement for this value is twips. The possible values for this attribute are defined by the XML Schema int datatype.
yWindow (Top Left Corner (Y Coordinate))	Specifies the Y coordinate for the upper left corner of the book window. The unit of measurement for this value is twips. The possible values for this attribute are defined by the XML Schema int datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustomWorkbookView">
   <sequence>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="name" type="ST_Xstring" use="required"/>
   <attribute name="guid" type="ST_Guid" use="required"/>
   <attribute name="autoUpdate" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="mergeInterval" type="xsd:unsignedInt" use="optional"/>
   <attribute name="changesSavedWin" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="onlySync" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="personalView" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="includePrintSettings" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="includeHiddenRowCol" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="maximized" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="minimized" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="showHorizontalScroll" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="showVerticalScroll" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="showSheetTabs" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="xWindow" type="xsd:int" use="optional" default="0"/>
   <attribute name="yWindow" type="xsd:int" use="optional" default="0"/>
   <attribute name="windowWidth" type="xsd:unsignedInt" use="required"/>
   <attribute name="windowHeight" type="xsd:unsignedInt" use="required"/>
   <attribute name="tabRatio" type="xsd:unsignedInt" use="optional" default="600"/>
   <attribute name="activeSheetId" type="xsd:unsignedInt" use="required"/>
   <attribute name="showFormulaBar" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="showStatusbar" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="showComments" type="ST_Comments" use="optional" default="commIndicator"/>
   <attribute name="showObjects" type="ST_Objects" use="optional" default="all"/>
</complexType>
```

3.2.4 customWorkbookViews (Custom Workbook Views)

This element defines the collection of custom workbook views that are defined for this workbook. There is no limit on the number of custom views that a user can create. Users create custom views when there is more than one way of viewing the workbook. Each view on the workbook data might be complex and time consuming to set up. Naming and persisting view settings enables the user to switch between the views easily.

[*Example*:

```
<customWorkbookViews>
  <customWorkbookView name="CustomView"
    guid="{CE6681F1-E999-414D-8446-68A031534B57}" maximized="1" xWindow="1"
    yWindow="1" windowWidth="1024" windowHeight="547" activeSheetId="1"/>
</customWorkbookViews>
```

end example]

Parent Elements
workbook (§3.2.27)

Child Elements	Subclause
customWorkbookView (Custom Workbook View)	§3.2.3

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustomWorkbookViews">
    <sequence>
        <element name="customWorkbookView" minOccurs="1" maxOccurs="unbounded"
          type="CT_CustomWorkbookView"/>
    </sequence>
</complexType>
```

3.2.5 definedName (Defined Name)

This element defines the defined names that are defined within this workbook. Defined names are descriptive text that is used to represents a cell, range of cells, formula, or constant value. Use easy-to-understand names, such as Products, to refer to hard to understand ranges, such as Sales!C20:C30.

A defined name in a formula can make it easier to understand the purpose of the formula. For example, the formula =SUM(FirstQuarterSales) might be easier to identify than =SUM(C20:C30).

Names are available to any sheet. For example, if the name ProjectedSales refers to the range A20:A30 on the first worksheet in a workbook, you can use the name ProjectedSales on any other sheet in the same workbook to refer to range A20:A30 on the first worksheet.

Names can also be used to represent formulas or values that do not change (constants). For example, you can use the name SalesTax to represent the sales tax amount (such as 6.2 percent) applied to sales transactions.

You can also link to a defined name in another workbook, or define a name that refers to cells in another workbook. For example, the formula =SUM(Sales.xls!ProjectedSales) refers to the named range ProjectedSales in the workbook named Sales.

A compliant producer or consumer will consider a defined name in the range A1-XFD1048576 to be invalid.

All other names outside this range can be defined as names and will override a cell reference if an ambiguity exists.

For clarification: LOG10 is always a cell reference, LOG10() is always formula, LOGO1000 can be a defined name that overrides a cell reference.

Parent Elements
definedNames (§3.2.6)

Attributes	Description

Attributes	Description
comment (Comment)	Specifies the comment the user provided when the name was created. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
customMenu (Custom Menu Text)	Specifies custom menu text for the defined name. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
description (Description)	Specifies description text for the defined name. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
function (Function)	Specifies a boolean value that indicates that the defined name refers to a user-defined function. This attribute is used when there is an add-in or other code project associated with the file. A value of on, 1, or true indicates the name refers to a function. A value of off, 0, or false indicates the name does not refer to a function. The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.
functionGroupId (Function Group Id)	Specifies the function group index if the defined name refers to a function. The function group defines the general category for the function. This attribute is used when there is an add-in or other code project associated with the file. The following functionGroupIds are defined in SpreadsheetML for applications that support the association of an add-in or code project for their workbook: • 1 Financial • 2 Date and Time • 3 Math and Trig • 4 Statistical • 5 Lookup and Reference • 6 Database • 7 Text • 8 Logical • 9 Information • 10 Commands • 11 Customizing • 12 Macro Control • 13 DDE / External • 14 User Defined • 15 Engineering

Attributes	Description
	• 16 Cube The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
help (Help)	Specifies the help topic to display for this defined name. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
hidden (Hidden Name)	Specifies a boolean value that indicates whether the defined name is hidden in the user interface. A value of on, 1, or true indicates the name is hidden. A value of off, 0, or false indicates the name is not hidden. The possible values for this attribute are defined by the XML Schema boolean datatype.
localSheetId (Local Name Sheet Id)	Specifies the sheet index in this workbook where data from an external reference is displayed. In the following example, the defined name refers to a range whose data source is an external database called “Northwind_Database”: [*Example*: `<definedName name="Northwind_Database" localSheetId="2">Sheet5!$A$1:$T$47</definedName>` *end example*] The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
name (Defined Name)	Specifies the name that will appear in the user interface for the defined name. This attribute is required. The following built-in names are defined in this SpreadsheetML specification: **Print** • _xlnm .Print_Area: this defined name specifies the workbook's print area. • _xlnm .Print_Titles: this defined name specifies the row(s) or column(s) to repeat at the top of each printed page. **Filter & Advanced Filter** • _xlnm .Criteria: this defined name refers to a range containing the criteria values to be used in applying an advanced filter to a range of data. • _xlnm ._FilterDatabase: can be one of the following a. this defined name refers to a range to which an advanced filter has been applied. This represents the source data range, unfiltered.

Attributes	Description
	b. This defined name refers to a range to which an AutoFilter has been applied. • _xlnm .Extract: this defined name refers to the range containing the filtered output values resulting from applying an advanced filter criteria to a source range. **Miscellaneous** • _xlnm .Consolidate_Area: the defined name refers to a consolidation area. • _xlnm .Database: the range specified in the defined name is from a database data source. • _xlnm .Sheet_Title: the defined name refers to a sheet title. Built-in names reserved by SpreadsheetML begin with "_xlnm.". End users shall not use this string for custom names in the user interface. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
publishToServer (Publish To Server)	Specifies a boolean value that indicates whether the defined name is included in the version of the workbook that is published to or rendered on a Web or application server. A value of on, 1, or true indicates the name shall be published. A value of off, 0, or false indicates the name shall not be published. The possible values for this attribute are defined by the XML Schema boolean datatype.
shortcutKey (Shortcut Key)	Specifies the keyboard shortcut for the defined name. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
statusBar (Status Bar)	Specifies text that is displayed on the application status bar when the user places focus on the defined name. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
vbProcedure (Procedure)	Specified a boolean value that indicates whether the defined name is related to an external function, command, or other executable code. A value of on, 1, or true indicates the name is related to an external function, command, or other executable code, and the loading application can optionally decide whether to load and/or execute the commands. A value of off, 0, or false indicates the name does not refer to an external function, command, or other executable code.

Attributes	Description
	The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.
workbookParameter (Workbook Parameter (Server))	Specifies a boolean value that indicates that the name is used as a workbook parameter on a version of the workbook that is published to or rendered on a Web or application server. A value of on, 1, or true indicates is a workbook parameter on the application server. A value of off, 0, or false indicates is not a workbook parameter on the application server. The possible values for this attribute are defined by the XML Schema boolean datatype.
xlm (External Function)	Specifies a boolean value that indicates whether the defined name is related to an external function, command, or other executable code. A value of on, 1, or true indicates the name is related to an external function, command, or other executable code, and the loading application can optionally decide whether to load and/or execute the commands. A value of off, 0, or false indicates the name does not refer to an external function, command, or other executable code. The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DefinedName">
   <simpleContent>
      <extension base="ST_Formula">
         <attribute name="name" type="ST_Xstring" use="required"/>
         <attribute name="comment" type="ST_Xstring" use="optional"/>
         <attribute name="customMenu" type="ST_Xstring" use="optional"/>
         <attribute name="description" type="ST_Xstring" use="optional"/>
         <attribute name="help" type="ST_Xstring" use="optional"/>
         <attribute name="statusBar" type="ST_Xstring" use="optional"/>
         <attribute name="localSheetId" type="xsd:unsignedInt" use="optional"/>
         <attribute name="hidden" type="xsd:boolean" use="optional" default="false"/>
         <attribute name="function" type="xsd:boolean" use="optional" default="false"/>
         <attribute name="vbProcedure" type="xsd:boolean" use="optional" default="false"/>
         <attribute name="xlm" type="xsd:boolean" use="optional" default="false"/>
         <attribute name="functionGroupId" type="xsd:unsignedInt" use="optional"/>
         <attribute name="shortcutKey" type="ST_Xstring" use="optional"/>
         <attribute name="publishToServer" type="xsd:boolean" use="optional" default="false"/>
         <attribute name="workbookParameter" type="xsd:boolean" use="optional" default="false"/>
      </extension>
   </simpleContent>
</complexType>
```

3.2.6 definedNames (Defined Names)

This element defines the collection of defined names for this workbook. Defined names are descriptive names to represent cells, ranges of cells, formulas, or constant values. Defined names can be used to represent a range on any worksheet.

[*Example*:

```
<definedNames>
  <definedName name="NamedFormula"
    comment="Comment text for defined name.">SUM(Sheet3!$B$2:$B$9)</definedName>
  <definedName name="NamedRange">Sheet3!$A$1:$C$12</definedName>
  <definedName name="NamedRangeFromExternalReference" localSheetId="2"
    hidden="1">Sheet5!$A$1:$T$47</definedName>
</definedNames>
```

end example]

Parent Elements
workbook (§3.2.27)

Child Elements	Subclause
definedName (Defined Name)	§3.2.5

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DefinedNames">
   <sequence>
      <element name="definedName" type="CT_DefinedName" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.2.7 ext (Extension)

Each ext element contains extensions to the standard SpreadsheetML feature set.

Parent Elements
extLst (§3.2.10)

Child Elements	Subclause
Any element from any namespace	n/a

Attributes	Description
uri (URI)	A token to identify version and application information for this particular extension. The possible values for this attribute are defined by the XML Schema token datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Extension">
   <sequence>
      <any processContents="lax"/>
   </sequence>
   <attribute name="uri" type="xsd:token"/>
</complexType>
```

3.2.8 externalReference (External Reference)

This element defines an external reference that stores data for workbook elements.

Parent Elements
externalReferences (§3.2.9)

Attributes	Description
id (Relationship Id) Namespace: .../officeDocument/2006/relationshi	Specifies a unique identifier that is used to identify a relationship to another part in the file. Relationship identifiers link the element definition with the part where data for the element is stored. The possible values for this attribute are defined by the ST_RelationshipId simple type

Attributes	Description
ps	(§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ExternalReference">
   <attribute ref="r:id" use="required"/>
</complexType>
```

3.2.9 externalReferences (External References)

This element defines the collection of external references for this workbook.

Parent Elements
workbook (§3.2.27)

Child Elements	Subclause
externalReference (External Reference)	§3.2.8

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ExternalReferences">
   <sequence>
      <element name="externalReference" type="CT_ExternalReference" minOccurs="1"
        maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.2.10 extLst (Future Feature Data Storage Area)

This element defines flexible storage extensions for implementing applications.

Parent Elements
autoFilter (§3.3.1.1); bk (§3.9.2); c (§3.3.1.3); cacheField (§3.10.1.3); cacheHierarchy (§3.10.1.6); cacheSource (§3.10.1.7); calcChain (§3.6.2); calculatedItem (§3.10.1.8); calculatedMember (§3.10.1.10); cellStyle (§3.8.7); cfRule (§3.3.1.9); cfvo (§3.3.1.10); chartsheet (§3.3.1.11); comments (§3.7.5); conditionalFormat (§3.10.1.18); conditionalFormatting (§3.3.1.17); connection (§3.13.1); customSheetView (§3.3.1.23); customWorkbookView (§3.2.3); dataField (§3.10.1.22); dialogsheet (§3.3.1.32); dxf (§3.8.14); externalLink (§3.14.8); filter (§3.10.1.33); filterColumn (§3.3.2.7); format (§3.10.1.35); futureMetadata (§3.9.4); groupLevel (§3.10.1.39); header (§3.11.1.1); ignoredErrors (§3.3.1.48); metadata (§3.9.8); nc (§3.11.1.3); ndxf (§3.11.1.4); oc (§3.11.1.5); odxf (§3.11.1.6); pageField (§3.10.1.62); pivotArea (§3.3.1.66); pivotCacheDefinition (§3.10.1.67); pivotCacheRecords (§3.10.1.68); pivotField (§3.10.1.69); pivotHierarchy (§3.10.1.72); pivotTableDefinition (§3.10.1.73); queryTable (§3.12.2); queryTableField (§3.12.4); queryTableRefresh (§3.12.6); rcc (§3.11.1.9); rdn (§3.11.1.13); reference (§3.10.2.1); rfmt (§3.11.1.17); row (§3.3.1.71); rsnm (§3.11.1.22); sheetView (§3.3.1.84); sheetView (§3.3.1.83); sheetViews (§3.3.1.86); sheetViews (§3.3.1.85); singleXmlCell (§3.5.2.1); sortState (§3.3.1.89); sst (§3.4.9); styleSheet (§3.8.39); table (§3.5.1.2); tableColumn (§3.5.1.3); tupleCache (§3.10.1.94); userInfo (§3.11.2.1); volTypes (§3.15.6); workbook (§3.2.27); workbookView (§3.2.30); worksheet (§3.3.1.96); xf (§3.8.45); xmlCellPr (§3.5.2.3); xmlColumnPr (§3.5.1.7);

Parent Elements
xmlPr (§3.5.2.4)

Child Elements	Subclause
ext (Extension)	§3.2.7

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ExtensionList">
   <sequence>
      <group ref="EG_ExtensionList" minOccurs="0"/>
   </sequence>
</complexType>
```

3.2.11 fileRecoveryPr (File Recovery Properties)

This element defines properties that track the state of the workbook file, such as whether the file was saved during a crash, or whether it should be opened in auto-recover mode.

Parent Elements
workbook (§3.2.27)

Attributes	Description
autoRecover (Auto Recover)	Specifies a boolean value that indicates whether the file is mark for auto-recovery. Applications typically mark files for auto-recover following a crash. A value of on, 1, or true indicates the file is marked for auto-recover. A value of off, 0, or false indicates the file is not marked for auto-recover. The default value for this attribute is /false. The possible values for this attribute are defined by the XML Schema boolean datatype.
crashSave (Crash Save)	Specifies a boolean value that indicates whether the application last saved the workbook file after a crash. A value of on, 1, or true indicates the workbook was last saved after a crash. A value of off, 0, or false indicates was not last saved as part of a crash. The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.
dataExtractLoad	Specifies a boolean value that indicates whether the application last opened the

Attributes	Description
(Data Extract Load)	workbook for data recovery. A value of on, 1, or true indicates the workbook was last opened for data recovery. A value of off, 0, or false indicates was not last opened for data recovery. The default value for this attribute is true. The possible values for this attribute are defined by the XML Schema boolean datatype.
repairLoad (Repair Load)	Specifies a boolean value that indicates whether the application last opened the workbook in safe or repair mode. A value of on, 1, or true indicates the workbook was last opened in safe or repair mode. A value of off, 0, or false indicates the workbook was last opened without problems. The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FileRecoveryPr">
    <attribute name="autoRecover" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="crashSave" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="dataExtractLoad" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="repairLoad" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.2.12 fileSharing (File Sharing)

This element tracks file sharing File sharing settings for the workbook.

Parent Elements
workbook (§3.2.27)

Attributes	Description
readOnlyRecommended (Read Only Recommended)	Specifies a boolean value that indicates on open, whether the application alerts the user that the file be marked as read-only. The possible values for this attribute are defined by the XML Schema boolean datatype.
reservationPassword (Write Reservation Password)	Specifies the hash of the password required for editing this workbook. This hash is optional and may be ignored. The hash is generated from an 8-bit wide character. 16-bit Unicode characters must be converted down to 8 bits before the hash is computed, using the logic defined in the revisionsPassword attribute of §3.2.29.

Attributes	Description
	The resulting value is hashed using the algorithm defined below. [*Note*: An example algorithm to hash the user input into the value stored is as follows: `// Function Input:` `//  szPassword: NULL terminated C-Style string` `//  cchPassword: The number of characters in szPassword (not including the NULL terminator)` `WORD GetPasswordHash(const CHAR *szPassword, int cchPassword) {` `  WORD wPasswordHash;` `  const CHAR *pch;` `  wPasswordHash = 0;` `  if (cchPassword > 0)` `  {` `  pch = &szPassword[cchPassword];` `  while (pch-- != szPassword)` `  {` ` wPasswordHash = ((wPasswordHash >> 14) & 0x01)	((wPasswordHash << 1) & 0x7fff);` ` wPasswordHash ^= *pch;` ` }` ` wPasswordHash ^= (0x8000	('N' << 8)	'K');` ` }`  ` return(wPasswordHash);` `}` *end note*] The possible values for this attribute are defined by the ST_UnsignedShortHex simple type (§3.18.87).
userName (User Name)	Specifies the username of the person with write reservation for this workbook. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FileSharing">
   <attribute name="readOnlyRecommended" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="userName" type="ST_Xstring"/>
   <attribute name="reservationPassword" type="ST_UnsignedShortHex"/>
</complexType>
```

3.2.13 fileVersion (File Version)

This element defines properties that track which version of the application accessed the data and source code contained in the file.

Parent Elements
workbook (§3.2.27)

Attributes	Description
appName (Application Name)	Specifies the application name. When saving, applications can write their appName value and optionally write `lastEdited` and `lowestEdited` attributes to track the version of the application that performed those actions. When opening the workbook, applications can examine the value of `appName` and decide how to interpret the `lastEdited`, `lowestEdited`, and `rupBuild` attributes. The possible values for this attribute are defined by the XML Schema string datatype.
codeName (Code Name)	Specifies the GUID that identifies the code project that is associated with the workbook. [*Note*: the primary use of this attribute is to track the version of the compiled code.] The possible values for this attribute are defined by the ST_Guid simple type (§3.18.41).
lastEdited (Last Edited Version)	Specifies the version of the application that last saved the workbook. This attribute is application-dependent. The possible values for this attribute are defined by the XML Schema string datatype.
lowestEdited (Lowest Edited Version)	Specifies the earliest version of the application that saved the workbook. This value is reset any time an application that can read all data in the file saves the file. This attribute is application-dependent. The possible values for this attribute are defined by the XML Schema string datatype.
rupBuild (Build Version)	Specifies the incremental public release of the application. For example, betas, service packs, and versions. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FileVersion">
    <attribute name="appName" type="xsd:string" use="optional"/>
    <attribute name="lastEdited" type="xsd:string" use="optional"/>
    <attribute name="lowestEdited" type="xsd:string" use="optional"/>
    <attribute name="rupBuild" type="xsd:string" use="optional"/>
    <attribute name="codeName" type="ST_Guid" use="optional"/>
</complexType>
```

3.2.14 functionGroup (Function Group)

This element represents a single function group.

Parent Elements
functionGroups (§3.2.15)

Attributes	Description
name (Name)	Specifies the name of the function group. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FunctionGroup">
   <attribute name="name" type="ST_Xstring"/>
</complexType>
```

3.2.15 functionGroups (Function Groups)

This element defines the collection of function groups for the workbook.

Parent Elements
workbook (§3.2.27)

Child Elements	Subclause
functionGroup (Function Group)	§3.2.14

Attributes	Description
builtInGroupCount (Built-in Function Group Count)	Specifies the count of built-in function groups that the application provides in this workbook. The default value for this attribute is 16. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FunctionGroups">
   <sequence maxOccurs="unbounded">
      <element name="functionGroup" type="CT_FunctionGroup" minOccurs="0"/>
   </sequence>
   <attribute name="builtInGroupCount" type="xsd:unsignedInt" default="16" use="optional"/>
</complexType>
```

3.2.16 oleSize (Embedded Object Size)

This element defines the embedded object server for this workbook.

Parent Elements
workbook (§3.2.27)

Attributes	Description
ref (Reference)	Specifies the reference for the embedded object. The possible values for this attribute are defined by the ST_Ref simple type (§3.18.64).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OleSize">
   <attribute name="ref" type="ST_Ref" use="required"/>
</complexType>
```

3.2.17 pivotCache (PivotCache)

This element represents a cache of data for pivot tables and formulas in the workbook.

Parent Elements
pivotCaches (§3.2.18)

Attributes	Description
cacheId (PivotCache Id)	Specifies the unique identifier for the pivot cache for this workbook in the pivot cache part. This attribute is required. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
id (Relationship Id) Namespace: .../officeDocument	Specifies the identifier to a pivot cache definition part where cached data is stored. This attribute is required.

Attributes	Description
/2006/relationships	The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PivotCache">
   <attribute name="cacheId" type="xsd:unsignedInt" use="required"/>
   <attribute ref="r:id" use="required"/>
</complexType>
```

3.2.18 pivotCaches (PivotCaches)

This element enumerates pivot cache definition parts used by pivot tables and formulas in this workbook.

[*Example*:

```
<pivotCaches>
  <pivotCache cacheId="4" r:id="rId8"/>
</pivotCaches>
```

end example]

Parent Elements
workbook (§3.2.27)

Child Elements	Subclause
pivotCache (PivotCache)	§3.2.17

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PivotCaches">
   <sequence>
      <element name="pivotCache" type="CT_PivotCache" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.2.19 sheet (Sheet Information)

This element defines a sheet in this workbook. Sheet data is stored in a separate part.

Parent Elements
sheets (§3.2.20)

Attributes	Description
id (Relationship Id)	Specifies the identifier of the sheet part where the definition for this sheet is stored.

Attributes	Description
Namespace: .../officeDocument/2006/relationships	This attribute is required. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
name (Sheet Name)	Specifies the name of the sheet. This name must be unique. This attribute is required. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
sheetId (Sheet Tab Id)	Specifies the internal identifier for the sheet. This identifier must be unique. This attribute is required. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
state (Visible State)	Specifies the visible state of this sheet. The default value for this attribute is "visible." The possible values for this attribute are defined by the ST_SheetState simple type (§3.18.70).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Sheet">
   <attribute name="name" type="ST_Xstring" use="required"/>
   <attribute name="sheetId" type="xsd:unsignedInt" use="required"/>
   <attribute name="state" type="ST_SheetState" use="optional" default="visible"/>
   <attribute ref="r:id" use="required"/>
</complexType>
```

3.2.20 sheets (Sheets)

This element represents the collection of sheets in the workbook. There are different types of sheets you can create in SpreadsheetML. The most common sheet type is a worksheet; also called a spreadsheet. A worksheet is the primary document that you use in SpreadsheetML to store and work with data. A worksheet consists of cells that are organized into columns and rows.

Some workbooks might have a modular design where there is one sheet for data and another worksheet for each type of analysis. In a complex modular system, you might have dozens of sheets, each dedicated to a specific task.

[*Example*:

```
<sheets>
  <sheet name="Sheet1" sheetId="1" r:id="rId1"/>
  <sheet name="Sheet2" sheetId="2" r:id="rId2"/>
  <sheet name="Sheet5" sheetId="3" r:id="rId3"/>
  <sheet name="Chart1" sheetId="4" type="chartsheet" r:id="rId4"/>
</sheets>
```

end example]

Parent Elements
workbook (§3.2.27)

Child Elements	Subclause
sheet (Sheet Information)	§3.2.19

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Sheets">
   <sequence>
      <element name="sheet" type="CT_Sheet" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.2.21 smartTagPr (Smart Tag Properties)

This element defines a collection of smart tag properties that determine smart tag behavior in the workbook.

[*Example*:

```
<smartTagPr embed="1" show="noIndicator"/>
```

end example]

Parent Elements
workbook (§3.2.27)

Attributes	Description
embed (Embed SmartTags)	Specifies a boolean value that indicates whether the application saves smart tags with the workbook. Smart tag information is saved both in the workbook part and the sheet parts. A value of on, 1, or true indicates the application saves smart tags with the workbook.

Attributes	Description
	A value of off, 0, or false indicates the application does not save smart tags with the workbook. The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.
show (Show Smart Tags)	Specifies how the application displays smart tags in the user interface. The default value for this attribute is "all." The possible values for this attribute are defined by the ST_SmartTagShow simple type (§3.18.73).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SmartTagPr">
   <attribute name="embed" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="show" type="ST_SmartTagShow" use="optional" default="all"/>
</complexType>
```

3.2.22 smartTagType (Smart Tag Type)

This element represents a smart tag in the workbook.

Parent Elements
smartTagTypes (§3.2.23)

Attributes	Description
name (Name)	Specifies the element name used for a smart tag that is used by the application. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
namespaceUri (SmartTag Namespace URI)	Specifies the namespace Uniform Resource Identifier (URI) for a smart tag used by the application. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
url (Smart Tag URL)	Specifies the URL for a smart tag provided by the smart tag provider in the application. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SmartTagType">
   <attribute name="namespaceUri" type="ST_Xstring" use="optional"/>
   <attribute name="name" type="ST_Xstring" use="optional"/>
   <attribute name="url" type="ST_Xstring" use="optional"/>
</complexType>
```

3.2.23 smartTagTypes (Smart Tag Types)

This element defines the collection of smart tag types in the workbook. Smart tags represent data that is recognized and labeled as a particular type. For example, a person's name or address can be recognized and labeled with a smart tag.

[*Example*:

```
<smartTagTypes>
  <smartTagType namespaceUri="urn:schemas-openxmlformats-org:office:smarttags"
    name="date"/>
</smartTagTypes>
```

end example]

Parent Elements
workbook (§3.2.27)

Child Elements	Subclause
smartTagType (Smart Tag Type)	§3.2.22

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SmartTagTypes">
   <sequence>
      <element name="smartTagType" type="CT_SmartTagType" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.2.24 webPublishing (Web Publishing Properties)

This element defines properties that relate to publishing this workbook to the Web.

Parent Elements
workbook (§3.2.27)

Attributes	Description
allowPng (Allow PNG)	Specifies a boolean value that indicates whether the application saves images in the PNG (Portable Network Graphics) graphic format.

Attributes	Description
	A value of on, 1, or true indicates the application supports PNG . A value of off, 0, or false indicates does not support PNG. The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.
codePage (Code Page)	Specifies the encoding the application will use when a Web page is saved. A code is table that relates the binary character codes used by a program to keys on the keyboard or to the appearance of characters on the display. Code pages are a means of providing support for the languages used in different countries. [*Note*: There are a number of code page technologies. One example of potential values can be found at: http://www.unicode.org/Public/MAPPINGS/VENDORS/MICSFT/WindowsBestFit/ *end note*] The default value for this attribute is the workbook's encoding. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
css (Use CSS)	Specifies a boolean value that indicates whether the application will use Cascading Style Sheet (CSS) for font formatting on Web pages. A value of on, 1, or true indicates the application will use CSS for font formats in Web pages. A value of off, 0, or false indicates the application will not use CSS for font formats. The default value for this attribute is true. The possible values for this attribute are defined by the XML Schema boolean datatype.
dpi (DPI)	Specifies the DPI (defined as the number of pixels per inch) that will be used to display images in Web pages. The specified DPI affects the size of graphics relative to the size of text on the screen. The default value for this attribute is 96. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
longFileNames (Enable Long File Names)	Specifies a boolean value that indicates whether the application allows file names longer than 8 characters for Web pages. The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description
targetScreenSize (Target Screen Size)	Specifies the screen size on which Web pages will be displayed. The specified screen size might affect the size and layout of images on web pages. The default value of this attribute is "800x600." The possible values for this attribute are defined by the ST_TargetScreenSize simple type (§3.18.81).
thicket (Thicket)	Specifies a boolean value that indicates that the application stores supporting files such as bullets, background textures, and graphics in a separate folder from the Web page The possible values for this attribute are defined by the XML Schema boolean datatype.
vml (VML in Browsers)	Specifies a boolean value that indicates whether the application uses VML (Vector Markup Language) to display graphics in Web browsers The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_WebPublishing">
    <attribute name="css" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="thicket" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="longFileNames" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="vml" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="allowPng" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="targetScreenSize" type="ST_TargetScreenSize" use="optional" default="800x600"/>
    <attribute name="dpi" type="xsd:unsignedInt" use="optional" default="96"/>
    <attribute name="codePage" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.2.25 webPublishObject (Web Publishing Object)

This element defines a single Web publishing object for the workbook. This element tracks basic information about an object in the workbook, such as a named range, that is published to the Web.

Parent Elements
webPublishObjects (§3.2.26)

Attributes	Description
autoRepublish (Auto Republish)	Specifies a boolean value that indicates whether the object specified in `sourceObject` will be automatically published every time the workbook is saved. A value of on, 1, or true indicates the application will publish the `sourceObject` when the workbook is saved. A value of off, 0, or false indicates the application will not publish the `sourceObject`

Attributes	Description
	when the workbook is saved. The possible values for this attribute are defined by the XML Schema boolean datatype.
destinationFile (Destination File)	Specifies the destination file name to which the sourceObject will be published. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
divId (Div Id)	Specifies the destination bookmark (div id) for the published object. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
id (Id)	Specifies the number, in "nnnnn" format, used in generated div id, in style id's, token filenames, and other variables. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
sourceObject (Source Object)	Specifies the named range to be published. If omitted, the entire workbook is published. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
title (Title)	Specifies the title of the published item. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_WebPublishObject">
    <attribute name="id" type="xsd:unsignedInt" use="required"/>
    <attribute name="divId" type="ST_Xstring" use="required"/>
    <attribute name="sourceObject" type="ST_Xstring" use="optional"/>
    <attribute name="destinationFile" type="ST_Xstring" use="required"/>
    <attribute name="title" type="ST_Xstring" use="optional"/>
    <attribute name="autoRepublish" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.2.26 webPublishObjects (Web Publish Objects)

This element defines the collection of Web publishing objects in the workbook.

Parent Elements
workbook (§3.2.27)

Child Elements	Subclause

Child Elements	Subclause
webPublishObject (Web Publishing Object)	§3.2.25

Attributes	Description
count (Count)	Specifies the number of items in the collection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_WebPublishObjects">
   <sequence>
      <element name="webPublishObject" type="CT_WebPublishObject" minOccurs="1"
        maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.2.27 workbook (Workbook)

This element is the root part of the workbook in SpreadsheetML. It defines the structure of the workbook:

- Sheets: represents the collection of worksheets in the workbook. The worksheet is the primary document that you use to store and work with data.
- Views: SpreadsheetML defines a collection of Workbook views that define basic window dimensions and position of the workbook. It also defines a collection of Custom Workbook Views that allow the end-user to define a series of rich views on their workbook data. Users can create more than one view of the same workbook without saving separate copies of the workbook.
- Properties: the workbook has several property collection that store basic workbook settings, such as the date system to use, file protection settings, calculation settings, and smart tag behaviors.
- Names: represent descriptive that represent cells, ranges of cells, formulas, or constant values.

[*Example*:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<workbook xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/5/main"
mlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships">
  <fileVersion lastEdited="4" lowestEdited="4" rupBuild="4017"/>
  <workbookPr date1904="1" vbName="ThisWorkbook" defaultThemeVersion="123820"/>
  <bookViews>
    <workbookView xWindow="120" yWindow="45" windowWidth="15135"
      windowHeight="7650" activeTab="4"/>
  </bookViews>
```

```
<sheets>
  <sheet name="Sheet1" sheetId="1" r:id="rId1"/>
  <sheet name="Sheet2" sheetId="2" r:id="rId2"/>
  <sheet name="Sheet5" sheetId="3" r:id="rId3"/>
  <sheet name="Chart1" sheetId="4" type="chartsheet" r:id="rId4"/>
</sheets>
<definedNames>
  <definedName name="MyDefinedName">Sheet3!$A$1:$C$12</definedName>
</definedNames>
<calcPr calcId="122211" calcMode="autoNoTable" refMode="R1C1" iterate="1"
  fullPrecision="0"/>
<customWorkbookViews>
  <customWorkbookView name="CustomView1"
    guid="{CE6681F1-E999-414D-8446-68A031534B57}" maximized="1" xWindow="1"
    yWindow="1" windowWidth="1024" windowHeight="547" activeSheetId="1"/>
</customWorkbookViews>
<pivotCaches>
  <pivotCache cacheId="0" r:id="rId8"/>
</pivotCaches>
<smartTagPr embed="1" show="noIndicator"/>
<smartTagTypes>
  <smartTagType namespaceUri="urn:schemas-openxmlformats-org:office:smarttags"
    name="date"/>
</smartTagTypes>
<webPublishing codePage="1252"/>
</workbook>
```

end example]

Parent Elements
Root element of SpreadsheetML Workbook part

Child Elements	Subclause
bookViews (Workbook Views)	§3.2.1
calcPr (Calculation Properties)	§3.2.2
customWorkbookViews (Custom Workbook Views)	§3.2.4
definedNames (Defined Names)	§3.2.6
externalReferences (External References)	§3.2.9
extLst (Future Feature Data Storage Area)	§3.2.10
fileRecoveryPr (File Recovery Properties)	§3.2.11
fileSharing (File Sharing)	§3.2.12

Child Elements	Subclause
fileVersion (File Version)	§3.2.13
functionGroups (Function Groups)	§3.2.15
oleSize (Embedded Object Size)	§3.2.16
pivotCaches (PivotCaches)	§3.2.18
sheets (Sheets)	§3.2.20
smartTagPr (Smart Tag Properties)	§3.2.21
smartTagTypes (Smart Tag Types)	§3.2.23
webPublishing (Web Publishing Properties)	§3.2.24
webPublishObjects (Web Publish Objects)	§3.2.26
workbookPr (Workbook Properties)	§3.2.28
workbookProtection (Workbook Protection)	§3.2.29

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Workbook">
   <sequence>
      <element name="fileVersion" type="CT_FileVersion" minOccurs="0" maxOccurs="1"/>
      <element name="fileSharing" type="CT_FileSharing" minOccurs="0" maxOccurs="1"/>
      <element name="workbookPr" type="CT_WorkbookPr" minOccurs="0" maxOccurs="1"/>
      <element name="workbookProtection" type="CT_WorkbookProtection" minOccurs="0" maxOccurs="1"/>
      <element name="bookViews" type="CT_BookViews" minOccurs="0" maxOccurs="1"/>
      <element name="sheets" type="CT_Sheets" minOccurs="1" maxOccurs="1"/>
      <element name="functionGroups" type="CT_FunctionGroups" minOccurs="0" maxOccurs="1"/>
      <element name="externalReferences" type="CT_ExternalReferences" minOccurs="0" maxOccurs="1"/>
      <element name="definedNames" type="CT_DefinedNames" minOccurs="0" maxOccurs="1"/>
      <element name="calcPr" type="CT_CalcPr" minOccurs="0" maxOccurs="1"/>
      <element name="oleSize" type="CT_OleSize" minOccurs="0" maxOccurs="1"/>
      <element name="customWorkbookViews" type="CT_CustomWorkbookViews" minOccurs="0"
        maxOccurs="1"/>
      <element name="pivotCaches" type="CT_PivotCaches" minOccurs="0" maxOccurs="1"/>
      <element name="smartTagPr" type="CT_SmartTagPr" minOccurs="0" maxOccurs="1"/>
      <element name="smartTagTypes" type="CT_SmartTagTypes" minOccurs="0" maxOccurs="1"/>
      <element name="webPublishing" type="CT_WebPublishing" minOccurs="0" maxOccurs="1"/>
      <element name="fileRecoveryPr" type="CT_FileRecoveryPr" minOccurs="0" maxOccurs="unbounded"/>
      <element name="webPublishObjects" type="CT_WebPublishObjects" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

3.2.28 workbookPr (Workbook Properties)

This element defines a collection of workbook properties.

[*Example*:

```
<workbookPr date1904="1" showObjects="none" saveExternalLinkValues="0"
  defaultThemeVersion="123820"/>
```

end example]

Parent Elements
workbook (§3.2.27)

Attributes	Description
allowRefreshQuery (Allow Refresh Query)	Specifies a boolean value that indicates whether the application will refresh query table in this workbook. A value of on, 1, or true indicates the application will refresh query tables when the workbook is loaded. A value of off, 0, or false indicates the application will not refresh query tables. The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.
autoCompressPictures (Auto Compress Pictures)	Specifies a boolean value that indicates the application automatically compressed pictures in the workbook. A value of on, 1, or true indicates the application automatically compresses pictures of the workbook. When a picture is compresses, the application: • Reduces resolution (to 96 dots per inch (dpi) for Web and 200 dpi for print), and unnecessary information is discarded. • Discards extra information. For example, when a picture has been cropped or resized, the "hidden" parts of the picture are stored in the file. • Compress the picture, if possible. A value of off, 0, or false indicates the application does not compress pictures in this workbook. The default value for this attribute is true. The possible values for this attribute are defined by the XML Schema boolean datatype.
backupFile (Create Backup File)	Specifies a boolean value that indicates whether the application creates a backup of the workbook on save. A value of on, 1, or true indicates the application creates a backup of the workbook on save. A value of off, 0, or false indicates the application does not create a backup.

Attributes	Description
	The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.
checkCompatibility (Check Compatibility On Save)	Specifies a boolean value that indicates whether the application checks for compatibility when saving this workbook to older file formats. A value of on, 1, or true indicates the application performs a compatibility check when saving to legacy binary formats. A value of off, 0, or false indicates the application does not perform a compatibility check when saving to legacy binary formats. The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.
codeName (Code Name)	Specifies the codename of the application that created this workbook. Use this attribute to track file content in incremental releases of the application. The possible values for this attribute are defined by the XML Schema string datatype.
date1904 (Date 1904)	Specifies a boolean value that indicates whether the date systems used in the workbook starts in 1904. A value of on, 1, or true indicates the date system starts in 1904. A value of off, 0, or false indicates the workbook uses the 1900 date system, where 1/1/1900 is the first day in the system. The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.
defaultThemeVersion (Default Theme Version)	Specifies the default version of themes to apply in the workbook. The value for defaultThemeVersion depends on the application. SpreadsheetML defaults to the form [version][build], where [version] refers to the version of the application, and [build] refers to the build of the application when the themes in the user interface changed. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
filterPrivacy (Filter Privacy)	Specifies a boolean value that indicates whether the application has been inspected the workbook for personally identifying information (PII). If this flag is set, the application warns the user any time the user performs do an action that will insert PII into the document. For example, inserting a comment might inserts the user's name.

Attributes	Description
	A value of on, 1, or true indicates the application will warn the user when they insert PII into the workbook. A value of off, 0, or false indicates the application will not warn the user when they insert PII into the workbook; the workbook has not been inspected for PII. The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.
hidePivotFieldList (Hide Pivot Field List)	Specifies a boolean value that indicates whether a list of fields is shown for pivot tables in the application user interface. A value of on, 1, or true indicates a list of fields is show for pivot tables. A value of off, 0, or false indicates a list of fields is not shown for pivot tables. The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.
promptedSolutions (Prompted Solutions)	Specifies a boolean value that indicates whether the user has received an alert to load Smart Document components. A value of on, 1, or true indicates the user received an alert to load SmartDoc. A value of off, 0, or false indicates the user did not receive an alert. The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.
publishItems (Publish Items)	Specifies a boolean value that indicates whether the publish the workbook or workbook items to the application server. A value of on, 1, or `true` indicates that workbook items are published. A value of `off`, `0`, or `false` indicates that the workbook is published. The default value for this attribute is `false`. The possible values for this attribute are defined by the XML Schema boolean datatype.
refreshAllConnections (Refresh all Connections on Open)	Specifies a boolean value that indicates whether the workbok shall refresh all the connections to data sources during load. The default value for this attribute is `false`. The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description
saveExternalLinkValues (Save External Link Values)	Specifies a boolean value that indicates whether the application will cache values retrieved from other workbooks via an externally linking formula. Data is cached at save. A value of on, 1, or true indicates data from externally linked formulas is cached. A supporting part is written out containing a cached cell table from the external workbook. A value of off, 0, or false indicates data from externally linked formulas is not cached. The default value for this attribute is true. The possible values for this attribute are defined by the XML Schema boolean datatype.
showBorderUnselectedTables (Show Border Unselected Table)	Specifies a boolean value that indicates whether a border is drawn around unselected tables in the workbook. A value of on, 1, or true indicates borders are drawn around unselected tables. A value of off, 0, or false indicates borders are not drawn around unselected tables. The default value for this attribute is true. The possible values for this attribute are defined by the XML Schema boolean datatype.
showInkAnnotation (Show Ink Annotations)	Specifies a boolean value that indicates whether the book shows ink annotations. A value of on, 1, or true indicates that ink annotations are shown in the workbook. A value of off, 0, or false indicates that ink annotations are not shown in the workbook. The default value for this attribute is true. The possible values for this attribute are defined by the XML Schema boolean datatype.
showObjects (Show Objects)	Specifies how the application shows embedded objects in the workbook. This attribute it optional. The default value for this attribute is "all." The possible values for this attribute are defined by the ST_Objects simple type (§3.18.50).
showPivotChartFilter (Show Pivot Chart Filter)	Specifies a boolean value that indicates whether filtering options are shown for pivot charts in the workbook. A value of on, 1, or true indicates filtering options shall be shown for pivot charts. A value of off, 0, or false indicates filtering options shall not be shown.

Attributes	Description
	The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.
updateLinks (Update Links Behavior)	Specifies how the application updates external links when the workbook is opened. The default value for this attribute is userSet. The possible values for this attribute are defined by the ST_UpdateLinks simple type (§3.18.88).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_WorkbookPr">
    <attribute name="date1904" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="showObjects" type="ST_Objects" use="optional" default="all"/>
    <attribute name="showBorderUnselectedTables" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="filterPrivacy" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="promptedSolutions" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="showInkAnnotation" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="backupFile" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="saveExternalLinkValues" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="updateLinks" type="ST_UpdateLinks" use="optional" default="userSet"/>
    <attribute name="codeName" type="xsd:string" use="optional"/>
    <attribute name="hidePivotFieldList" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="showPivotChartFilter" type="xsd:boolean" default="false"/>
    <attribute name="allowRefreshQuery" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="publishItems" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="checkCompatibility" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="autoCompressPictures" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="refreshAllConnections" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="defaultThemeVersion" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.2.29 workbookProtection (Workbook Protection)

This element specifies options for protecting data in the workbook. Applications may use workbook protection to prevent anyone from accidentally changing, moving, or deleting important data. This protection may be ignored by applications which choose not to support this optional protection mechanism.

[*Note*: Worksheet or workbook element protection should not be confused with file security. It is not meant to make your workbook safe from unintentional modification, and cannot protect it from malicious modification. *end note*]

Parent Elements
workbook (§3.2.27)

Attributes	Description

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>lockRevision (Lock Revisions)</td><td>Specifies a boolean value that indicates whether the workbook is locked for revisions.

The possible values for this attribute are defined by the XML Schema boolean datatype.</td></tr>
<tr><td>lockStructure (Lock Structure)</td><td>Specifies a boolean value that indicates whether structure of workbook is locked.

A value of on, 1, or true indicates the structure of the workbook is locked. Worksheets in the workbook can't be moved, deleted, hidden, unhidden, or renamed, and new worksheets can't be inserted.

A value of off, 0, or false indicates the structure of the workbook is not locked.

The default value for this attribute is false.

The possible values for this attribute are defined by the XML Schema boolean datatype.</td></tr>
<tr><td>lockWindows (Lock Windows)</td><td>Specifies a boolean value that indicates whether the windows that comprise the workbook are locked.

A value of on, 1, or true indicates the workbook windows are locked. Windows are the same size and position each time the workbook is opened.

A value of off, 0, or false indicates the workbook windows are not locked.

The default value for this attribute is false.

The possible values for this attribute are defined by the XML Schema boolean datatype.</td></tr>
<tr><td>revisionsPassword (Revisions Password)</td><td>Specifies the hash of the password required for unlocking revisions in this workbook. The hash is generated from an 8-bit wide character. 16-bit Unicode characters must be converted down to 8 bits before the hash is computed, using the following logic:

For SpreadsheetML password hash purposes, Unicode UTF-16 input code points are converted to an “ansi” single or double byte code page from the following list:
<table>
<tr><td>874</td><td>windows-874</td><td>ANSI/OEM Thai (same as 28605, ISO 8859-15); Thai (Windows)</td></tr>
<tr><td>932</td><td>shift_jis</td><td>ANSI/OEM Japanese; Japanese (Shift-JIS)</td></tr>
<tr><td>936</td><td>gb2312</td><td>ANSI/OEM Simplified Chinese (PRC, Singapore); Chinese Simplified (GB2312)</td></tr>
<tr><td>949</td><td>ks_c_5601-1987</td><td>ANSI/OEM Korean (Unified Hangul Code)</td></tr>
<tr><td>950</td><td>big5</td><td>ANSI/OEM Traditional Chinese (Taiwan; Hong Kong SAR, PRC); Chinese Traditional (Big5)</td></tr>
<tr><td>1250</td><td>windows-1250</td><td>ANSI Central European; Central European (Windows)</td></tr>
<tr><td>1251</td><td>windows-1251</td><td>ANSI Cyrillic; Cyrillic (Windows)</td></tr>
<tr><td>1252</td><td>windows-1252</td><td>ANSI Latin 1; Western European (Windows)</td></tr>
</table>
</td></tr>
</table>

Attributes	Description

1253	windows-1253	ANSI Greek; Greek (Windows)
1254	windows-1254	ANSI Turkish; Turkish (Windows)
1255	windows-1255	ANSI Hebrew; Hebrew (Windows)
1256	windows-1256	ANSI Arabic; Arabic (Windows)
1257	windows-1257	ANSI Baltic; Baltic (Windows)
1258	windows-1258	ANSI/OEM Vietnamese; Vietnamese (Windows)

Code points with no representation in the target code page are replaced with Unicode character 0x3f (?).

The necessary mapping tables can be found at the following location: http://www.unicode.org/Public/MAPPINGS/VENDORS/MICSFT/WindowsBestFit/ .

Code pages 932, 936, 949, and 950 are “Double Byte” code pages. The remainder of the “ANSI” code pages supported by windows are “Single Byte” code pages.

For single byte code pages each Unicode code point is replaced by a single byte or 0x3f if an appropriate character doesn’t exist in the code page.

For double byte code pages, each Unicode code point is replaced by either a single byte, or a two byte sequence, depending on the input character, or 0x3f if an appropriate character doesn’t exist in the code page. In our tables the target is a single byte sequence if the most significant byte is 0x00, otherwise it is a double byte sequence, with the lead byte being the most significant byte.

To convert, first check if conversion is being done to a single or double byte code page and load the appropriate WCTABLE code page table.

For each input WCHAR, look up the code point in the WCTABLE. There are 3 possibilities: Not found, single byte, or double byte.

If the input WCHAR is not found, append 0x3f and continue to the next WCHAR.
If the result is a single byte, check to make sure the entry in the MBTABLE matches the input. If it matches, append the single byte to the output. If it does not match, append 0x3f to the output.

If the result is a double byte,check to make sure the entry in the DBCSENTRY table for the appropriate lead byte matches the input WCHAR. If it matches, append the lead byte and trail byte to the output. If it does not match, append 0x3f to the output.

The following pueudocode describes how this conversion should be done:

```
int WideCharToMultiByte(WCHAR* wszInput, byte* szOutput)
{
```

Attributes	Description

```
    // Remember output start so we can return length
    byte* szOutputStart = szOutput;

    // Ask the system for the current ANSI code page, which
    // on windows is a system setting.
    int iCodePage = GetCurrentAnsiCodePage();

    // Load Code Page Tables
    // This will depend on how the code pages are represented
on
    // the target machine.  TABLECLASS represents some abstract
    // representation of this structure here.
    TABLECLASS pTables = LoadCodePageTables(iCodePage);

    bool bDoubleByte = false;
    if (iCodePage == 932 ||
        iCodePage == 936 ||
        iCodePage == 949 ||
        iCodePage == 950)
        bDoubleByte = true;

    while (*wszInput != 0)
    {
        if (bDoubleByte)
            szOutput = AppendDoubleByte(pTables, *wszInput,
szOutput);
        else
            szOutput = AppendSingleByte(pTables, *wszInput,
szOutput);

        // Read next input WCHAR
        wszInput++;
    }

    // Null terminate the output
    *szOutput = 0;

    // Return output length
    return szOutput – szOutputStart;
}

byte* AppendSingleByte(TABLECLASS pTables, WCHAR wcIn, byte*
szOutput)
{
    // Look up byte that we want to append.
    byte bOut = pTables->LookUpSingleByte(wcIn);

    // Make sure that bOut matches the input, otherwise use ?
```

Attributes	Description

```
    // (ie: no best fit behavior allowed)
    if (wcIn != pTables->LookUpWideChar(bOut))
        bOut = 0x3f;

    *szOutput = bOut;
    szOutput++;
    return szOutput;
}

byte* AppendDoubleByte(TABLECLASS pTables, WCHAR wcIn, byte*
szOutput)
{
    // Look up bytes that we want to append.
    UINT16 bytesOut = pTables->LookUpDoubleByte(wcIn);

    // See if it is a single or double byte sequence
    if (bytesOut & 0xFF00)
    {
        // It is a double byte sequence
        // Make sure that bytesOut matches the input, otherwise
use ?
        // (ie: no best fit behavior allowed)
        if (wcIn != pTables->LookUpWideChar(bytesOut))
        {
            // Use ?, it will be added below
            bytesOut = 0x003f;
        }
        else
        {
            // It matched, use the lead byte we found
            // trail byte will be added below
            *szOutput = bytesOut >> 8;
            szOutput++;
    }
    else
    {
        // It is a single byte sequence
        // Make sure that bytesOut matches the input, otherwise
use ?
        // (ie: no best fit behavior allowed)
        if (wcIn != pTables->LookUpWideChar(bytesOut & 0xFF))
            bytesOut = 0x003f;
    }

    // Add the single or trail byte
    *szOutput = bytesOut & 0xFF;
    szOutput++;
```

Attributes	Description

```
    return szOutput;
}

class pTables
{
    // Construction depends on how you choose to store & load
the
    // table files

    byte LookUpSingleByte(WCHAR wcIn)
    {
        // How you access the table depends on your storage
mechanism.
        // Look up the line in WCTABLE where the first column
matches wcIn,
        // and then return the byte value from the second
column.
        if (exists WCTABLE{wcIn})
            return WCTABLE{wcIn}.SecondColumn;

        // If it doesn't exist, return ?
        return 0x3f;
    }

    UINT16 LookUpDoubleByte(WCHAR wcIn)
    {
        // How you access the table depends on your storage
mechanism.
        // Look up the line in WCTABLE where the first column
matches wcIn,
        // and then return the double byte value from the
second column.
        if (exists WCTABLE{wcIn})
            return WCTABLE{wcIn}.SecondColumn;

        // If it doesn't exist, return ?
        return 0x003f;
    }

    // Overload that looks up wide chars from single byte code
points.
    WCHAR LookUpWideChar(byte bIn)
    {
        // How you access the table depends on your storage
mechanism.
        // Look up the line in MBTABLE where the first column
matches bIn,
        // and then return the WCHAR value from the second
```

Attributes	Description

```
column.
        if (exists MBTABLE{bIn})
            return MBTABLE{bIn}.SecondColumn;

        // If it doesn't exist, return ?
        return 0x003f;
    }

    // Overload that looks up wide chars from double byte code
points
    WCHAR LookUpWideChar(UINT16 bytesIn)
    {
        // How you access the table depends on your storage
mechanism.
        // First find the DBCSTABLE where the LeadByte matches
        // the lead (most significant) input byte.
        if (exists DBCSTABLE{bytesIn >> 8))
        {
            DbcsTable = DBCSTABLE{bytesIn >> 8);

            // Look up the line in DbcsTable where the first
column
            // matches the input trail (least significant)
byte,
            // and then return the WCHAR value from the second
column.
            if (exists DbcsTable{bytesIn & 0xFF})
            return DbcsTable{bytesIn & 0xFF}.SecondColumn;
        }

        // Either the lead byte table or specific trail byte
        // doesn't exist in the table, return ?
        return 0x003f;
    }
}
```

The resulting value is hashed using the algorithm defined below.

[*Note*: An example algorithm to hash the resulting single-byte user input into the value stored is as follows:

```
// Function Input:
//    szPassword: NULL terminated C-Style string
//    cchPassword: The number of characters in szPassword (not
including the NULL terminator)
WORD GetPasswordHash(const CHAR *szPassword, int cchPassword) {
      WORD wPasswordHash;
      const CHAR *pch;
```

Attributes	Description

	``
	wPasswordHash = 0;
	
	if (cchPassword > 0)
	{
	pch = &szPassword[cchPassword];
	while (pch-- != szPassword)
	{
	wPasswordHash = ((wPasswordHash >> 14) &
	0x01) | ((wPasswordHash << 1) & 0x7fff);
	wPasswordHash ^= *pch;
	}
	wPasswordHash ^= (0x8000 | ('N' << 8) | 'K');
	}
	
	return(wPasswordHash);
	}
	``
	end note]
	
	The possible values for this attribute are defined by the ST_UnsignedShortHex simple type (§3.18.87).

workbookPassword (Workbook Password)	Specifies the hash of the password required for unlocking revisions in this workbook. The hash is generated from an 8-bit wide character. 16-bit Unicode characters must be converted down to 8 bits before the hash is computed, using the following logic:
	
	For SpreadsheetML password hash purposes, Unicode UTF-16 input code points are converted to an “ansi” single or double byte code page using the logic defined in the preceding revisionsPassword attribute.
	
	The resulting value is hashed using the algorithm defined below.
	
	[*Note*: An example algorithm to hash the user input into the value stored is as follows:
	
	``
	// Function Input:
	// szPassword: NULL terminated C-Style string
	// cchPassword: The number of characters in szPassword (not
	including the NULL terminator)
	WORD GetPasswordHash(const CHAR *szPassword, int cchPassword) {
	WORD wPasswordHash;
	const CHAR *pch;
	
	wPasswordHash = 0;
	
	if (cchPassword > 0)
	{
	pch = &szPassword[cchPassword];
	while (pch-- != szPassword)
	``

Attributes	Description

	{
	wPasswordHash = ((wPasswordHash >> 14) & 0x01) | ((wPasswordHash << 1) & 0x7fff);
	wPasswordHash ^= *pch;
	}
	wPasswordHash ^= (0x8000 | ('N' << 8) | 'K');
	}
	
	return(wPasswordHash);
	}
	end note]
	
	The possible values for this attribute are defined by the ST_UnsignedShortHex simple type (§3.18.87).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_WorkbookProtection">
   <attribute name="workbookPassword" type="ST_UnsignedShortHex" use="optional"/>
   <attribute name="revisionsPassword" type="ST_UnsignedShortHex" use="optional"/>
   <attribute name="lockStructure" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="lockWindows" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="lockRevision" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.2.30 workbookView (Workbook View)

This element specifies a single Workbook view.

Units for window widths and other dimensions are expressed in twips. Twip measurements are portable between different display resolutions. The formula is (screen pixels) * (20 * 72) / (logical device dpi), where the logical device dpi can be different for x and y coordinates.

Parent Elements
bookViews (§3.2.1)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10

Attributes	Description
activeTab (Active Sheet Index)	Specifies an unsignedInt that contains the index to the active sheet in this book view. The default value for this attribute is 0.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
autoFilterDateGrouping (AutoFilter Date Grouping)	Specifies a boolean value that indicates whether to group dates when presenting the user with filtering options in the user interface. A value of on, 1, or true indicates that dates are grouped. A value of off, 0, or false indicates that dates are not grouped. The default value for this attribute is true. The possible values for this attribute are defined by the XML Schema boolean datatype.
firstSheet (First Sheet)	Specifies the index to the first sheet in this book view. The default value for this attribute is 0. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
minimized (Minimized)	Specifies a boolean value that indicates whether the book window is minimized. A value of on, 1, or true indicates the book window is minimized. A value of off, 0, or false indicates the book window is not minimized. The default value for this attribute is false. The possible values for this attribute are defined by the XML Schema boolean datatype.
showHorizontalScroll (Show Horizontal Scroll)	Specifies a boolean value that indicates whether to display the horizontal scroll bar in the user interface. A value of on, 1, or true indicates that the horizontal scrollbar shall be shown. A value of off, 0, or false indicates that the horizontal scrollbar shall not be shown. The default value for this attribute is true. The possible values for this attribute are defined by the XML Schema boolean datatype.
showSheetTabs (Show Sheet Tabs)	Specifies a boolean value that indicates whether to display the sheet tabs in the user interface. A value of on, 1, or true indicates that sheet tabs shall be shown. A value of off, 0, or false indicates that sheet tabs shall not be shown.

Attributes	Description
	The default value for this attribute is true. The possible values for this attribute are defined by the XML Schema boolean datatype.
showVerticalScroll (Show Vertical Scroll)	Specifies a boolean value that indicates whether to display the vertical scroll bar. A value of on, 1, or true indicates the vertical scrollbar shall be shown. A value of off, 0, or false indicates the vertical scrollbar shall not be shown. The default value for this attribute is true. The possible values for this attribute are defined by the XML Schema boolean datatype.
tabRatio (Sheet Tab Ratio)	Specifies ratio between the workbook tabs bar and the horizontal scroll bar. The default value for this attribute is 600. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
visibility (Visibility)	Specifies visible state of the book window. The default value for this attribute is "visible." The possible values for this attribute are defined by the ST_Visibility simple type (§3.18.91).
windowHeight (Window Height)	Specifies the height of the workbook window. The unit of measurement for this value is twips. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
windowWidth (Window Width)	Specifies the width of the workbook window. The unit of measurement for this value is twips. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
xWindow (Upper Left Corner (X Coordinate))	Specifies the X coordinate for the upper left corner of the book window. The unit of measurement for this value is twips. The possible values for this attribute are defined by the XML Schema int datatype.
yWindow (Upper Left Corner (Y Coordinate))	Specifies the Y coordinate for the upper left corner of the book window. The unit of measurement for this value is twips. The possible values for this attribute are defined by the XML Schema int datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BookView">
   <sequence>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="visibility" type="ST_Visibility" use="optional" default="visible"/>
   <attribute name="minimized" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="showHorizontalScroll" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="showVerticalScroll" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="showSheetTabs" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="xWindow" type="xsd:int" use="optional"/>
   <attribute name="yWindow" type="xsd:int" use="optional"/>
   <attribute name="windowWidth" type="xsd:unsignedInt" use="optional"/>
   <attribute name="windowHeight" type="xsd:unsignedInt" use="optional"/>
   <attribute name="tabRatio" type="xsd:unsignedInt" use="optional" default="600"/>
   <attribute name="firstSheet" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="activeTab" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="autoFilterDateGrouping" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

3.3 Worksheets

Sheets are the central structures within a workbook, and are where the user does most of their spreadsheet work. The most common type of sheet is the worksheet, which is represented as a grid of cells. Worksheet cells can contain text, numbers, dates, and formulas. Cells can be formatted as well. Sheets often have Workbooks usually contain more than one sheet. To aid in the analysis of data and making informed decisions, spreadsheet applications often implement features and objects which help calculate, sort, filter, organize, and graphically display information. Since these features are often connected very tightly with the spreadsheet grid, these are also included in the sheet definition on disk.

Other types of sheets include chart sheets and dialog sheets.

Note that sheet information is organized into three main sections:

- Top-level sheet properties (everything before sheetData)
- The cell table (sheetData)
- Supporting sheet features (everything after sheetData)

3.3.1 Worksheets

The following elements define a sheet and its contents:

3.3.1.1 autoFilter (AutoFilter Settings)

AutoFilter temporarily hides rows based on a filter criteria, which is applied column by column to a table of data in the worksheet. This collection expresses AutoFilter settings.

[*Example*: This example expresses a filter indicating to 'show only values greater than 0.5'. The filter is being applied to the range B3:E8, and the criteria is being applied to values in the column whose colId='1' (zero based

column numbering, from left to right). Therefore any rows shall be hidden if the value in that particular column is less than or equal to 0.5.

```
<autoFilter ref="B3:E8">
  <filterColumn colId="1">
    <customFilters>
      <customFilter operator="greaterThan" val="0.5"/>
    </customFilters>
  </filterColumn>
</autoFilter>
```

end example]

Parent Elements
customSheetView (§3.3.1.23); filter (§3.10.1.33); table (§3.5.1.2); worksheet (§3.3.1.96)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
filterColumn (AutoFilter Column)	§3.3.2.7
sortState (Sort State)	§3.3.1.89

Attributes	Description
ref (Cell or Range Reference)	Reference to the cell range to which the AutoFilter is applied. The possible values for this attribute are defined by the ST_Ref simple type (§3.18.64).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AutoFilter">
   <sequence>
      <element name="filterColumn" minOccurs="0" maxOccurs="unbounded" type="CT_FilterColumn"/>
      <element name="sortState" minOccurs="0" maxOccurs="1" type="CT_SortState"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="ref" type="ST_Ref"/>
</complexType>
```

3.3.1.2 brk (Break)

Individual row or column breaks

Parent Elements

Parent Elements
colBreaks (§3.3.1.13); rowBreaks (§3.3.1.72)

Attributes	Description
id (Id)	Zero-based row or column Id of the page break. Breaks occur above the specified row and left of the specified column. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
man (Manual Page Break)	Manual Break flag. '1' means the break is a manually inserted break. The possible values for this attribute are defined by the XML Schema boolean datatype.
max (Maximum)	Zero-based index of end row or column of the break. For row breaks, specifies column index; for column breaks, specifies row index. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
min (Minimum)	Zero-based index of start row or column of the break. For row breaks, specifies column index; for column breaks, specifies row index. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
pt (Pivot-Created Page Break)	Flag indicating that a PivotTable created this break. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Break">
   <attribute name="id" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="min" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="max" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="man" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="pt" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.3.1.3 c (Cell)

This collection represents a cell in the worksheet. Information about the cell's location (reference), value, data type, formatting, and formula is expressed here.

[*Example*: This example shows the information stored for a cell whose address in the grid is C6, whose style index is '6', and whose value metadata index is '15'. The cell contains a formula as well as a calculated result of that formula.

```
<c r="C6" s="1" vm="15">
  <f>CUBEVALUE("xlextdat9 Adventure Works",C$5,$A6)</f>
  <v>2838512.355</v>
</c>
```

end example]

While a cell can have a formula element f and a value element v, when the cell's type t is `inlineStr` then only the element is is allowed as a child element.

[*Example*:

Here is an example of expressing a string in the cell rather than using the shared string table.

```
<row r="1" spans="1:1">
  <c r="A1" t="inlineStr">
    <is><t>This is inline string example</t></is>
  </c>
</row>
```

end example]

Parent Elements
row (§3.3.1.71)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
f (Formula)	§3.3.1.37
is (Rich Text Inline)	§3.3.1.50
v (Cell Value)	§3.3.1.93

Attributes	Description
cm (Cell Metadata Index)	The zero-based index of the cell metadata record associated with this cell. Metadata information is found in the Metadata Part. Cell metadata is extra information stored at the cell level, and is attached to the cell (travels through moves, copy / paste, clear, etc). Cell metadata is not accessible via formula reference. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
ph (Show Phonetic)	A Boolean value indicating if the spreadsheet application should show phonetic information. Phonetic information is displayed in the same cell across the top of the cell and serves as a 'hint' which indicates how the text should be pronounced. This should

Attributes	Description
	only be used for East Asian languages. The possible values for this attribute are defined by the XML Schema boolean datatype.
r (Reference)	An A1 style reference to the location of this cell The possible values for this attribute are defined by the ST_CellRef simple type (§3.18.8).
s (Style Index)	The index of this cell's style. Style records are stored in the Styles Part. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
t (Cell Data Type)	An enumeration representing the cell's data type. The possible values for this attribute are defined by the ST_CellType simple type (§3.18.12).
vm (Value Metadata Index)	The zero-based index of the value metadata record associated with this cell's value. Metadata records are stored in the Metadata Part. Value metadata is extra information stored at the cell level, but associated with the value rather than the cell itself. Value metadata is accessible via formula reference. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Cell">
   <sequence>
      <element name="f" type="CT_CellFormula" minOccurs="0" maxOccurs="1"/>
      <element name="v" type="ST_Xstring" minOccurs="0" maxOccurs="1"/>
      <element name="is" type="CT_Rst" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="r" type="ST_CellRef" use="optional"/>
   <attribute name="s" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="t" type="ST_CellType" use="optional" default="n"/>
   <attribute name="cm" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="vm" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="ph" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.3.1.4 cellSmartTag (Cell Smart Tag)

Single smart tag associated with a cell. There can be more than one cellSmartTag for a cell.

Parent Elements
cellSmartTags (§3.3.1.6)

Child Elements	Subclause
cellSmartTagPr (Smart Tag Properties)	§3.3.1.5

Attributes	Description
deleted (Deleted)	Boolean flag indicating that the application shouldn't display a particular smart tag in the cell, for example when the user has chosen to explicitly remove the Smart Tag by interacting with the application's user interface. The possible values for this attribute are defined by the XML Schema boolean datatype.
type (Smart Tag Type Index)	Book-level zero-based index of the smart tag type. This index references a `<smartTagType>` element in the `<smartTagTypes>` collection in the workbook start part. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
xmlBased (XML Based)	Boolean flag indicating the Smart Tag recognition is triggered because the cell is associated with an XML map (schema-based semantic recognition), as contrasted with the more usual cell-content-based recognition type of smart tags. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CellSmartTag">
   <sequence>
      <element name="cellSmartTagPr" minOccurs="0" maxOccurs="unbounded" type="CT_CellSmartTagPr"/>
   </sequence>
   <attribute name="type" type="xsd:unsignedInt" use="required"/>
   <attribute name="deleted" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="xmlBased" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.3.1.5 cellSmartTagPr (Smart Tag Properties)

Represents a single property of a smart tag in a cell; contains a key-value pair.

Parent Elements
cellSmartTag (§3.3.1.4)

Attributes	Description
key (Key Name)	Key name of a single property of a smart tag in a cell. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

Attributes	Description
val (Value)	String value of a single property of a smart tag in a cell. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CellSmartTagPr">
   <attribute name="key" type="ST_Xstring" use="required"/>
   <attribute name="val" type="ST_Xstring" use="required"/>
</complexType>
```

3.3.1.6 cellSmartTags (Cell Smart Tags)

The element is used to label the cell with a smart tag. A cell may be determined to have semantic meaning and the cell containing this data can be labeled with a smart tag. The type of actions you can take depend on the semantic meaning of the data and the actions that the application decides to associate with that type of smart tag.

[*Example*: If you recently sent mail to "Chad Rothschiller", and you type the name into a cell on the worksheet, the name is recognized and given a smart tag with actions you can take including Send Mail, Schedule a Meeting, Open Contact, or Add to Contacts.

end example]

An application may decide that the smart tag indicators appear in the cell in the worksheet.

This collection represents a collection of smart tags on a cell.

[*Example*: This example expresses a smart tag associated with cell A1. The @type is used to associate this smart tag with a workbook-level smart tag type defined in the workbook start part.

```
<cellSmartTags r="A1">
  <cellSmartTag type="0"/>
</cellSmartTags>
```

end example]

Parent Elements
smartTags (§3.3.1.87)

Child Elements	Subclause
cellSmartTag (Cell Smart Tag)	§3.3.1.4

Attributes	Description
r (Reference)	Reference to the cell that contains this set of smart tags. The possible values for this attribute are defined by the ST_CellRef simple type (§3.18.8).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CellSmartTags">
   <sequence>
      <element name="cellSmartTag" type="CT_CellSmartTag" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="r" type="ST_CellRef" use="required"/>
</complexType>
```

3.3.1.7 cellWatch (Cell Watch Item)

The watch window is a single UI location where the application user can keep track of certain cell formulas & values which they have chosen to be in the set of watched cells. This element expresses the cell address of a cell being watched. It is always a reference to a single cell.

Parent Elements
cellWatches (§3.3.1.8)

Attributes	Description
r (Reference)	Cell reference of the cell being watched. The possible values for this attribute are defined by the ST_CellRef simple type (§3.18.8).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CellWatch">
   <attribute name="r" type="ST_CellRef" use="required"/>
</complexType>
```

3.3.1.8 cellWatches (Cell Watch Items)

Collection of cells on this worksheet being watched in the 'watch window'.

[*Example*: In this example, cells B3 and B4 are being watched.

```
<cellWatches>
  <cellWatch r="B3"/>
  <cellWatch r="B4"/>
</cellWatches>
```

end example]

Parent Elements
worksheet (§3.3.1.96)

Child Elements	Subclause
cellWatch (Cell Watch Item)	§3.3.1.7

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CellWatches">
   <sequence>
      <element name="cellWatch" type="CT_CellWatch" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.3.1.9 cfRule (Conditional Formatting Rule)

This collection represents a description of a conditional formatting rule.

[*Example*:

This example shows a conditional formatting rule highlighting cells whose values are greater than 0.5. Note that in this case the content of `<formula>` is a static value, but can also be a formula expression.

```
<conditionalFormatting sqref="E3:E9">
  <cfRule type="cellIs" dxfId="0" priority="1" operator="greaterThan">
    <formula>0.5</formula>
  </cfRule>
</conditionalFormatting>
```

end example]

Only rules with a type value of `expression` support formula syntax.

Parent Elements
conditionalFormatting (§3.3.1.17)

Child Elements	Subclause
colorScale (Color Scale)	§3.3.1.15
dataBar (Data Bar)	§3.3.1.26
extLst (Future Feature Data Storage Area)	§3.2.10
formula (Formula)	§3.3.1.40
iconSet (Icon Set)	§3.3.1.46

Attributes	Description
aboveAverage (Above Or Below Average)	Indicates whether the rule is an "above average" rule. '1' indicates 'above average'. Valid only for type = aboveAverage. The possible values for this attribute are defined by the XML Schema boolean datatype.
bottom (Bottom N)	Indicates whether a "top/bottom n" rule is a "bottom n" rule. '1' indicates 'bottom'. Valid only for type = top10. The possible values for this attribute are defined by the XML Schema boolean datatype.
dxfId (Differential Formatting Id)	This is an index to a dxf element in the Styles Part indicating which cell formatting to apply when the conditional formatting rule criteria is met. The possible values for this attribute are defined by the ST_DxfId simple type (§3.18.26).
equalAverage (Equal Average)	Flag indicating whether the 'aboveAverage' and 'belowAverage' criteria is inclusive of the average itself, or exclusive of that value. '1' indicates to include the average value in the criteria. Valid only for type = aboveAverage. The possible values for this attribute are defined by the XML Schema boolean datatype.
operator (Operator)	The operator in a "cell value is" conditional formatting rule. Valid only when type = cellIs The possible values for this attribute are defined by the ST_ConditionalFormattingOperator simple type (§3.18.16).
percent (Top 10 Percent)	Indicates whether a "top/bottom n" rule is a "top/bottom n percent" rule. Valid only for type = top10. The possible values for this attribute are defined by the XML Schema boolean datatype.
priority (Priority)	The priority of this conditional formatting rule. This value is used to determine which format should be evaluated and rendered. Lower numeric values are higher priority than higher numeric values, where '1' is the highest priority. The possible values for this attribute are defined by the XML Schema int datatype.
rank (Rank)	The value of "n" in a "top/bottom n" conditional formatting rule. Valid only for type = top10. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
stdDev (StdDev)	The number of standard deviations to include above or below the average in the conditional formatting rule. Valid only for type = aboveAverage. If a value is present for stdDev and the rule type = aboveAverage, then this rule is automatically an "above or below N standard deviations" rule.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema int datatype.
stopIfTrue (Stop If True)	If this flag is '1', no rules with lower priority may be applied over this rule, when this rule evaluates to true. The possible values for this attribute are defined by the XML Schema boolean datatype.
text (Text)	The text value in a "text contains" conditional formatting rule. Valid only for type = `containsText`. The possible values for this attribute are defined by the XML Schema string datatype.
timePeriod (Time Period)	The applicable time period in a "date occurring…" conditional formatting rule. Valid only for type = `timePeriod`. The possible values for this attribute are defined by the ST_TimePeriod simple type (§3.18.82).
type (Type)	Type of conditional formatting rule. The possible values for this attribute are defined by the ST_CfType simple type (§3.18.13).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CfRule">
   <sequence>
      <element name="formula" type="ST_Formula" minOccurs="0" maxOccurs="3"/>
      <element name="colorScale" type="CT_ColorScale" minOccurs="0" maxOccurs="1"/>
      <element name="dataBar" type="CT_DataBar" minOccurs="0" maxOccurs="1"/>
      <element name="iconSet" type="CT_IconSet" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="type" type="ST_CfType"/>
   <attribute name="dxfId" type="ST_DxfId" use="optional"/>
   <attribute name="priority" type="xsd:int" use="required"/>
   <attribute name="stopIfTrue" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="aboveAverage" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="percent" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="bottom" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="operator" type="ST_ConditionalFormattingOperator" use="optional"/>
   <attribute name="text" type="xsd:string" use="optional"/>
   <attribute name="timePeriod" type="ST_TimePeriod" use="optional"/>
   <attribute name="rank" type="xsd:unsignedInt" use="optional"/>
   <attribute name="stdDev" type="xsd:int" use="optional"/>
   <attribute name="equalAverage" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.3.1.10 cfvo (Conditional Format Value Object)

Describes the values of the interpolation points in a gradient scale.

[*Example*: This example demonstrates a color scale conditional formatting rule, which defines a color for the minimum value in the range of cell values, a color for the midpoint value, and a color for the maximum value in the in the range of cell values. Information is given about how to define the midpoint. In this case, it is the 50 percent mark.

```
<colorScale>
  <cfvo type="min" val="0"/>
  <cfvo type="percent" val="50"/>
  <cfvo type="max" val="0"/>
  <color rgb="FFFF0000"/>
  <color rgb="FFFFFF00"/>
  <color rgb="FF00B050"/>
</colorScale>
```

The first <cfvo> element corresponds with the first <color> definition, and so on.

end example]

Parent Elements
colorScale (§3.3.1.15); dataBar (§3.3.1.26); iconSet (§3.3.1.46)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10

Attributes	Description
gte (Greater Than Or Equal)	For icon sets, determines whether this threshold value uses the greater than or equal to operator. '0' indicates 'greater than' is used instead of 'greater than or equal to'. The possible values for this attribute are defined by the XML Schema boolean datatype.
type (Type)	The type of this conditional formatting value object. For example 'min' and 'max' would be used (in conjunction with @val) to express the lower and upper values to be used in a gradient. The possible values for this attribute are defined by the ST_CfvoType simple type (§3.18.14).
val (Value)	The value of this conditional formatting value object. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Cfvo">
   <sequence>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="type" type="ST_CfvoType" use="required"/>
   <attribute name="val" type="ST_Xstring" use="optional"/>
   <attribute name="gte" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

3.3.1.11 chartsheet (Chart Sheet)

This is the root element of Sheet Parts that are of type 'chartsheet'.

Parent Elements
Root element of SpreadsheetML Chartsheet part

Child Elements	Subclause
customSheetViews (Custom Chart Sheet Views)	§3.3.1.24
drawing (Drawing)	§3.3.1.34
extLst (Future Feature Data Storage Area)	§3.2.10
headerFooter (Header Footer Settings)	§3.3.1.43
legacyDrawing (Legacy Drawing Reference)	§3.3.1.51
legacyDrawingHF (Legacy Drawing Reference in Header Footer)	§3.3.1.52
pageMargins (Page Margins)	§3.3.1.60
pageSetup (Chart Sheet Page Setup)	§3.3.1.62
picture (Background Image)	§3.3.1.65
sheetPr (Chart Sheet Properties)	§3.3.1.80
sheetProtection (Chart Sheet Protection)	§3.3.1.82
sheetViews (Chart Sheet Views)	§3.3.1.86
webPublishItems (Web Publishing Items)	§3.3.1.95

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Chartsheet">
   <sequence>
      <element name="sheetPr" type="CT_ChartsheetPr" minOccurs="0" maxOccurs="1"/>
      <element name="sheetViews" type="CT_ChartsheetViews" minOccurs="1" maxOccurs="1"/>
      <element name="sheetProtection" type="CT_ChartsheetProtection" minOccurs="0" maxOccurs="1"/>
      <element name="customSheetViews" type="CT_CustomChartsheetViews" minOccurs="0" maxOccurs="1"/>
      <element name="pageMargins" minOccurs="0" type="CT_PageMargins"/>
      <element name="pageSetup" type="CT_CsPageSetup" minOccurs="0" maxOccurs="1"/>
      <element name="headerFooter" minOccurs="0" type="CT_HeaderFooter"/>
      <element name="drawing" type="CT_Drawing" minOccurs="1" maxOccurs="1"/>
      <element name="legacyDrawing" type="CT_LegacyDrawing" minOccurs="0" maxOccurs="1"/>
      <element name="legacyDrawingHF" type="CT_LegacyDrawing" minOccurs="0" maxOccurs="1"/>
      <element name="picture" type="CT_SheetBackgroundPicture" minOccurs="0" maxOccurs="1"/>
      <element name="webPublishItems" type="CT_WebPublishItems" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

3.3.1.12 col (Column Width & Formatting)

Defines column width and column formatting for one or more columns of the worksheet.

[*Example*: This example shows that column 5 (E) has width and style information applied.

```
<col min="5" max="5" width="9.140625" style="3"/>
```

end example]

Parent Elements
cols (§3.3.1.16)

Attributes	Description
bestFit (Best Fit Column Width)	Flag indicating if the specified column(s) is set to 'best fit'. 'Best fit' is set to `true` under these conditions: • The column width has never been manually set by the user, AND • The column width is not the default width 'Best fit' means that when numbers are typed into a cell contained in a 'best fit' column, the column width should automatically resize to display the number. Note: In best fit cases, column width shall not be made smaller, only larger. The possible values for this attribute are defined by the XML Schema boolean datatype.
collapsed (Collapsed)	Flag indicating if the outlining of the affected column(s) is in the collapsed state. See description of row collapsed and outlinePr element's summaryBelow and summaryRight attributes for detailed information.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema boolean datatype.
customWidth (Custom Width)	Flag indicating that the column width for the affected column(s) is different from the default or has been manually set. The possible values for this attribute are defined by the XML Schema boolean datatype.
hidden (Hidden Columns)	Flag indicating if the affected column(s) are hidden on this worksheet. The possible values for this attribute are defined by the XML Schema boolean datatype.
max (Maximum Column)	Last column affected by this 'column info' record. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
min (Minimum Column)	First column affected by this 'column info' record. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
outlineLevel (Outline Level)	Outline level of affected column(s). Range is 0 to 7. See description of outlinePr element's summaryBelow and summaryRight attributes for detailed information. The possible values for this attribute are defined by the XML Schema unsignedByte datatype.
phonetic (Show Phonetic Information)	Flag indicating if the phonetic information should be displayed by default for the affected column(s) of the worksheet. The possible values for this attribute are defined by the XML Schema boolean datatype.
style (Style)	Default style for the affected column(s). Affects cells not yet allocated in the column(s). In other words, this style applies to new columns. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
width (Column Width)	Column width measured as the number of characters of the maximum digit width of the numbers 0, 1, 2, ..., 9 as rendered in the normal style's font. There are 4 pixels of margin padding (two on each side), plus 1 pixel padding for the gridlines. width = Truncate([{Number of Characters} * {Maximum Digit Width} + {5 pixel padding}]/{Maximum Digit Width}*256)/256 Using the Calibri font as an example, the maximum digit width of 11 point font size is 7 pixels (at 96 dpi). In fact, each digit is the same width for this font. Therefore if the cell width is 8 characters wide, the value of this attribute shall be Truncate([8*7+5]/7*256)/256 = 8.7109375. To translate the value of width in the file into the column width value at runtime

Attributes	Description
	(expressed in terms of pixels), use this calculation: =Truncate(((256 * {width} + Truncate(128/{Maximum Digit Width}))/256)*{Maximum Digit Width}) Using the same example as above, the calculation would be Truncate(((256*8.7109375+Truncate(128/7))/256)*7) = 61 pixels To translate from pixels to character width, use this calculation: =Truncate(({pixels}-5)/{Maximum Digit Width} * 100+0.5)/100 Using the example above, the calculation would be Truncate((61-5)/7*100+0.5)/100 = 8 characters. Note: when wide borders are applied, part of the left/right border shall overlap with the 2 pixel padding on each side. Wide borders do not affect the width calculation of the column. Note: When the sheet is in the mode to view formulas instead of values, the pixel width of the column is doubled. The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Col">
    <attribute name="min" type="xsd:unsignedInt" use="required"/>
    <attribute name="max" type="xsd:unsignedInt" use="required"/>
    <attribute name="width" type="xsd:double" use="optional"/>
    <attribute name="style" type="xsd:unsignedInt" use="optional" default="0"/>
    <attribute name="hidden" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="bestFit" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="customWidth" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="phonetic" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="outlineLevel" type="xsd:unsignedByte" use="optional" default="0"/>
    <attribute name="collapsed" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.3.1.13 colBreaks (Vertical Page Breaks)

Vertical page break information used for print layout view, page layout view, drawing print breaks in normal view, and for printing the worksheet.

[*Example*:

In this example, a page break has been inserted at C3 (the break occurs left and above C3).

```
<colBreaks count="1" manualBreakCount="1">
  <brk id="2" max="1048575" man="1"/>
</colBreaks>
```

end example]

Parent Elements
customSheetView (§3.3.1.23); worksheet (§3.3.1.96)

Child Elements	Subclause
brk (Break)	§3.3.1.2

Attributes	Description
count (Page Break Count)	Number of breaks in the collection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
manualBreakCount (Manual Break Count)	Number of manual breaks in the collection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PageBreak">
   <sequence>
      <element name="brk" type="CT_Break" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="manualBreakCount" type="xsd:unsignedInt" use="optional" default="0"/>
</complexType>
```

3.3.1.14 color (Data Bar Color)

One of the colors associated with the data bar or color scale.

Note: the auto attribute is not used in the context of data bars.

Parent Elements
bottom (§3.8.6); colorScale (§3.3.1.15); dataBar (§3.3.1.26); diagonal (§3.8.13); font (§3.8.21); horizontal (§3.8.24); left (§3.8.27); mruColors (§3.8.28); right (§3.8.35); rPr (§3.4.7); stop (§3.8.38); top (§3.8.43); vertical (§3.8.44)

Attributes	Description
auto (Automatic)	A boolean value indicating the color is automatic and system color dependent. The possible values for this attribute are defined by the XML Schema boolean datatype.
indexed (Index)	Indexed color value. Only used for backwards compatibility. References a color in indexedColors. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
rgb (Alpha Red Green Blue Color Value)	Standard Alpha Red Green Blue color value (ARGB). The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§3.18.86).
theme (Theme Color)	Index into the <clrScheme> collection, referencing a particular <sysClr> or <srgbClr> value expressed in the Theme part. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
tint (Tint)	Specifies the tint value applied to the color. If tint is supplied, then it is applied to the RGB value of the color to determine the final color applied. The tint value is stored as a double from -1.0 .. 1.0, where -1.0 means 100% darken and 1.0 means 100% lighten. Also, 0.0 means no change. In loading the RGB value, it is converted to HLS where HLS values are (0..HLSMAX), where HLSMAX is currently 255. [*Example*: Here are some examples of how to apply tint to color: **If (tint < 0)** Lum' = Lum * (1.0 + tint) For example: Lum = 200; tint = -0.5; Darken 50% Lum‘ = 200 * (0.5) => 100 For example: Lum = 200; tint = -1.0; Darken 100% (make black) Lum‘ = 200 * (1.0-1.0) => 0 **If (tint > 0)** Lum‘ = Lum * (1.0-tint) + (HLSMAX – HLSMAX * (1.0-tint)) For example: Lum = 100; tint = 0.75; Lighten 75%

Attributes	Description
	Lum‘ = 100 * (1-.75) + (HLSMAX – HLSMAX*(1-.75)) = 100 * .25 + (255 – 255 * .25) = 25 + (255 – 63) = 25 + 192 = 217 For example: Lum = 100; tint = 1.0; Lighten 100% (make white) Lum‘ = 100 * (1-1) + (HLSMAX – HLSMAX*(1-1)) = 100 * 0 + (255 – 255 * 0) = 0 + (255 – 0) = 255 *end example*] The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <attribute name="auto" type="xsd:boolean" use="optional"/>
   <attribute name="indexed" type="xsd:unsignedInt" use="optional"/>
   <attribute name="rgb" type="ST_UnsignedIntHex" use="optional"/>
   <attribute name="theme" type="xsd:unsignedInt" use="optional"/>
   <attribute name="tint" type="xsd:double" use="optional" default="0.0"/>
</complexType>
```

3.3.1.15 colorScale (Color Scale)

Describes a gradated color scale in this conditional formatting rule.

[*Example*:

```
<colorScale>
  <cfvo type="min" val="0"/>
  <cfvo type="max" val="0"/>
  <color theme="5"/>
  <color rgb="FFFFEF9C"/>
</colorScale>
```

end example]

Parent Elements
cfRule (§3.3.1.9)

Child Elements	Subclause
cfvo (Conditional Format Value Object)	§3.3.1.10
color (Data Bar Color)	§3.3.1.14

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ColorScale">
   <sequence>
      <element name="cfvo" type="CT_Cfvo" minOccurs="2" maxOccurs="unbounded"/>
      <element name="color" type="CT_Color" minOccurs="2" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.3.1.16 cols (Column Information)

Information about whole columns of the worksheet.

[*Example*:

This example shows that column 4 (D) has 'best fit' applied to it, which is also a custom width. Also, column 5 (E) is listed as having a custom width and a style applied at the column level (as opposed to the cell level).

```
<cols>
  <col min="4" max="4" width="12" bestFit="1" customWidth="1"/>
  <col min="5" max="5" width="9.140625" style="3"/>
</cols>
```

end example]

Parent Elements
worksheet (§3.3.1.96)

Child Elements	Subclause
col (Column Width & Formatting)	§3.3.1.12

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Cols">
   <sequence>
      <element name="col" type="CT_Col" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.3.1.17 conditionalFormatting (Conditional Formatting)

A Conditional Format is a format, such as cell shading or font color, that a spreadsheet applicaiton can automatically apply to cells if a specified condition is true. This collection expresses conditional formatting rules applied to a particular cell or range.

[*Example*: This example applies a 'top10' rule to the cells C3:C8. The `@dxfId` references the formatting (defined in the styles part) to be applied to cells that match the criteria.

```
<conditionalFormatting sqref="C3:C8">
  <cfRule type="top10" dxfId="1" priority="3" rank="2"/>
</conditionalFormatting>
```

end example]

Parent Elements
worksheet (§3.3.1.96)

Child Elements	Subclause
cfRule (Conditional Formatting Rule)	§3.3.1.9
extLst (Future Feature Data Storage Area)	§3.2.10

Attributes	Description
pivot (PivotTable Conditional Formatting)	Flag indicating if this is conditional formatting associated with a PivotTable. The possible values for this attribute are defined by the XML Schema boolean datatype.
sqref (Sequence of Refernces)	Range over which these conditional formatting rules apply. The possible values for this attribute are defined by the ST_Sqref simple type (§3.18.78).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ConditionalFormatting">
   <sequence>
      <element name="cfRule" type="CT_CfRule" minOccurs="1" maxOccurs="unbounded"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="pivot" type="xsd:boolean" default="false"/>
   <attribute name="sqref" type="ST_Sqref"/>
</complexType>
```

3.3.1.18 control (Embedded Control)

A single embedded control.

Parent Elements
controls (§3.3.1.19)

Attributes	Description
id (Relationship Id) Namespace:	This relationship ID references an Embedded Control Data part which contains control-specific properties and state information about this particular embedded control.

Attributes	Description
.../officeDocument/2006/relationships	The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
name (Control Name)	The code name of the control. The possible values for this attribute are defined by the XML Schema string datatype.
shapeId (Shape Id)	ID of the drawing shape in the Legacy Drawing part with which this control is associated. The drawing is used to draw the control in the sheet. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Control">
   <attribute name="shapeId" type="xsd:unsignedInt" use="required"/>
   <attribute ref="r:id" use="required"/>
   <attribute name="name" type="xsd:string" use="optional"/>
</complexType>
```

3.3.1.19 controls (Embedded Controls)

Worksheets can have embedded controls embedded in them. This collection is a listing of embedded controls in this worksheet. This collection is used to reference individual Embedded Control Data part definitions, enumerate the code name of each control, and reference drawing information used to draw the control.

Parent Elements
worksheet (§3.3.1.96)

Child Elements	Subclause
control (Embedded Control)	§3.3.1.18

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Controls">
   <sequence>
      <element name="control" type="CT_Control" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.3.1.20 customPr (Custom Property)

The custom property element provides a mechanism to store name/value pairs of arbitrary user-defined data. The name is stored in the attribute name, the arbitrary data is stored in the binary part referenced by the relationshipId.

[*Note*: There is nothing in the binary part except the arbitrary data itself.

Custom XML Data Properties provide a preferred mechanism for storing arbitrary data. The customPr supports legacy third-party document components, as well as those situations that have a stringent need for binary parts. *end note*]

Parent Elements
customProperties (§3.3.1.21)

Attributes	Description
id (Relationship Id) Namespace: .../officeDocument/2006/relationships	This relationship references the binary part containing the specified custom properties. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
name (Custom Property Name)	Name of the custom property The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustomProperty">
   <attribute name="name" type="ST_Xstring" use="required"/>
   <attribute ref="r:id" use="required"/>
</complexType>
```

3.3.1.21 customProperties (Custom Properties)

This collection is used to reference binary parts containing arbitrary user-defined data.

Parent Elements
worksheet (§3.3.1.96)

Child Elements	Subclause
customPr (Custom Property)	§3.3.1.20

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustomProperties">
   <sequence>
      <element name="customPr" type="CT_CustomProperty" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.3.1.22 customSheetView (Custom Chart Sheet View)

This element defines custom view properties for chart sheets. See customSheetView (§3.3.1.23) for an example.

Parent Elements
customSheetViews (§3.3.1.24)

Child Elements	Subclause
headerFooter (Header Footer Settings)	§3.3.1.43
pageMargins (Page Margins)	§3.3.1.60
pageSetup (Chart Sheet Page Setup)	§3.3.1.62

Attributes	Description
guid (GUID)	Unique identifier of this custom view The possible values for this attribute are defined by the ST_Guid simple type (§3.18.41).
scale (Print Scale)	Print scaling, representing percent values. Valid values range from 10 to 400. Horizontal & Vertical scale together. For example: 10 - 10% 20 - 20% … 100 - 100% … 400 - 400% The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
state (Visible State)	Visibility state of the sheet. The possible values for this attribute are defined by the ST_SheetState simple type (§3.18.70).
zoomToFit (Zoom To Fit)	Flag indicating whether chart sheet is zoom to fit window. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustomChartsheetView">
   <sequence>
      <element name="pageMargins" type="CT_PageMargins" minOccurs="0" maxOccurs="1"/>
      <element name="pageSetup" type="CT_CsPageSetup" minOccurs="0" maxOccurs="1"/>
      <element name="headerFooter" type="CT_HeaderFooter" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="guid" type="ST_Guid" use="required"/>
   <attribute name="scale" type="xsd:unsignedInt" default="100"/>
   <attribute name="state" type="ST_SheetState" default="visible"/>
   <attribute name="zoomToFit" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.3.1.23 customSheetView (Custom Sheet View)

This collection stores information pertaining to one custom sheet view definition. A custom view is a collection of settings defining a particular view of the sheet. These views may be selected by the user for quick access to predefined views of the sheet.

[*Example*: This example indicates that there is both a horizontal and vertical split in the sheet view, and that the top left cell of the bottom right pane is F7. Page margin, print options, page setup, and header / footer information is also stored with this view.

```
<customSheetView guid="{F3A061A9-D5FD-4F9C-A7CD-483AD476BA25}"
  sizeWithWindow="0">
  <pane xSplit="5" ySplit="6" topLeftCell="F7"/>
  <selection/>
  <pageMargins left="0.7" right="0.7" top="0.75" bottom="0.75" header="0.3"
    footer="0.3"/>
  <printOptions gridLinesSet="0"/>
  <pageSetup paperSize="0" scale="0" orientation="portrait" printDriver="0"
    horizontalDpi="0" verticalDpi="0" copies="0"/>
  <headerFooter/>
</customSheetView>
```

end example]

Parent Elements
customSheetViews (§3.3.1.25)

Child Elements	Subclause
autoFilter (AutoFilter Settings)	§3.3.1.1
colBreaks (Vertical Page Breaks)	§3.3.1.13
extLst (Future Feature Data Storage Area)	§3.2.10
headerFooter (Header Footer Settings)	§3.3.1.43

Child Elements	Subclause
pageMargins (Page Margins)	§3.3.1.60
pageSetup (Page Setup Settings)	§3.3.1.61
pane (View Pane)	§3.3.1.64
printOptions (Print Options)	§3.3.1.68
rowBreaks (Horizontal Page Breaks (Row))	§3.3.1.72
selection (Selection)	§3.3.1.75

Attributes	Description
colorId (Color Id)	Index to the color value for the text in row/column headings and gridlines for this custom view. This is an 'index color value' (ICV) rather than rgb value. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
filter (Filtered List)	Flag indicating whether the view contains a filtered range. The possible values for this attribute are defined by the XML Schema boolean datatype.
filterUnique (Filter)	Indicates whether an advanced filter has been applied, and the option to filter out duplicate records from the data list has been selected, in this custom view. The possible values for this attribute are defined by the XML Schema boolean datatype.
fitToPage (Fit To Page)	Flag indicating whether this view should be fit to page when printing this custom view. The possible values for this attribute are defined by the XML Schema boolean datatype.
guid (GUID)	Unique identifier of this custom view. This is used to ensure uniqueness. It is generated when the view is created. Must correspond to a customWorkbookView guid value in the workbook Start Part. The possible values for this attribute are defined by the ST_Guid simple type (§3.18.41).
hiddenColumns (Hidden Columns)	Flag indicating that there is one or more hidden column(s) in this custom view. The possible values for this attribute are defined by the XML Schema boolean datatype.
hiddenRows (Hidden Rows)	Flag indicating that there is one or more hidden row(s) in this custom view. The possible values for this attribute are defined by the XML Schema boolean datatype.
outlineSymbols (Show Outline Symbols)	Flag indicating whether outline symbols are displayed in this custom view. The possible values for this attribute are defined by the XML Schema boolean datatype.
printArea (Print Area Defined)	Flag indicating whether a print area is defined as part of this custom view. The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description
scale (Print Scale)	Print scaling for this custom view. Valid values range from 10 to 400. For example: 10 - 10% 20 - 20% … 100 - 100% … 400 - 400% The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
showAutoFilter (Show AutoFitler Drop Down Controls)	Flag indicating whether the autofilter dropdown buttons are visible in this custom view. The possible values for this attribute are defined by the XML Schema boolean datatype.
showFormulas (Show Formulas)	Flag indicating whether formulas are shown in this custom view. The possible values for this attribute are defined by the XML Schema boolean datatype.
showGridLines (Show Grid Lines)	Flag indicating whether gridlines are shown in this custom view. The possible values for this attribute are defined by the XML Schema boolean datatype.
showPageBreaks (Show Page Breaks)	Flag indicating whether page breaks are shown in this custom view. The possible values for this attribute are defined by the XML Schema boolean datatype.
showRowCol (Show Headers)	Flag indicating whether row and column headers are shown in this custom view. The possible values for this attribute are defined by the XML Schema boolean datatype.
showRuler (Show Ruler)	Flag indicating whether to show the ruler in this custom view. Only applicable if this Custom View is in Page Layout View. The possible values for this attribute are defined by the XML Schema boolean datatype.
state (Visible State)	Visibility state for this custom view. The possible values for this attribute are defined by the ST_SheetState simple type (§3.18.70).
topLeftCell (Top Left Visible Cell)	Location of the top left visible cell in the bottom right pane in this custom view (when in Left-to-Right mode). The possible values for this attribute are defined by the ST_CellRef simple type (§3.18.8).
view (View Type)	Indicates the view type for this Custom View

Attributes	Description
	The possible values for this attribute are defined by the ST_SheetViewType simple type (§3.18.71).
zeroValues (Show Zero Values)	Flag indicating whether the window should display 0 (zero) values in this custom view. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustomSheetView">
   <sequence>
      <element name="pane" type="CT_Pane" minOccurs="0" maxOccurs="1"/>
      <element name="selection" type="CT_Selection" minOccurs="0" maxOccurs="1"/>
      <element name="rowBreaks" type="CT_PageBreak" minOccurs="0" maxOccurs="1"/>
      <element name="colBreaks" type="CT_PageBreak" minOccurs="0" maxOccurs="1"/>
      <element name="pageMargins" type="CT_PageMargins" minOccurs="0" maxOccurs="1"/>
      <element name="printOptions" type="CT_PrintOptions" minOccurs="0" maxOccurs="1"/>
      <element name="pageSetup" type="CT_PageSetup" minOccurs="0" maxOccurs="1"/>
      <element name="headerFooter" type="CT_HeaderFooter" minOccurs="0" maxOccurs="1"/>
      <element name="autoFilter" type="CT_AutoFilter" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="guid" type="ST_Guid" use="required"/>
   <attribute name="scale" type="xsd:unsignedInt" default="100"/>
   <attribute name="colorId" type="xsd:unsignedInt" default="64"/>
   <attribute name="showPageBreaks" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="showFormulas" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="showGridLines" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="showRowCol" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="outlineSymbols" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="zeroValues" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="fitToPage" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="printArea" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="filter" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="showAutoFilter" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="hiddenRows" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="hiddenColumns" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="state" type="ST_SheetState" default="visible"/>
   <attribute name="filterUnique" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="view" type="ST_SheetViewType" default="normal"/>
   <attribute name="showRuler" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="topLeftCell" type="ST_CellRef" use="optional"/>
</complexType>
```

3.3.1.24 customSheetViews (Custom Chart Sheet Views)

Collection of custom Chart Sheet View information.

Parent Elements
chartsheet (§3.3.1.11)

Child Elements	Subclause
customSheetView (Custom Chart Sheet View)	§3.3.1.22

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustomChartsheetViews">
   <sequence>
      <element name="customSheetView" minOccurs="0" maxOccurs="unbounded"
        type="CT_CustomChartsheetView"/>
   </sequence>
</complexType>
```

3.3.1.25 customSheetViews (Custom Sheet Views)

This is a collection of custom sheet views.

Parent Elements
dialogsheet (§3.3.1.32); worksheet (§3.3.1.96)

Child Elements	Subclause
customSheetView (Custom Sheet View)	§3.3.1.23

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustomSheetViews">
   <sequence>
      <element name="customSheetView" minOccurs="1" maxOccurs="unbounded"
        type="CT_CustomSheetView"/>
   </sequence>
</complexType>
```

3.3.1.26 dataBar (Data Bar)

Describes a data bar conditional formatting rule.

[*Example*:

In this example a data bar conditional format is expressed, which spreads across all cell values in the cell range, and whose color is blue.

```
<dataBar>
  <cfvo type="min" val="0"/>
  <cfvo type="max" val="0"/>
  <color rgb="FF638EC6"/>
</dataBar>
```

end example]

The length of the data bar for any cell can be calculated as follows:

Data bar length = minLength + (cell value - minimum value in the range) / (maximum value in the range - minimum value in the range) * (maxLength - minLength),

where min and max length are a fixed percentage of the column width (by default, 10% and 90% respectively.)

The minimum difference in length (or increment amount) is 1 pixel.

Parent Elements
cfRule (§3.3.1.9)

Child Elements	Subclause
cfvo (Conditional Format Value Object)	§3.3.1.10
color (Data Bar Color)	§3.3.1.14

Attributes	Description
maxLength (Maximum Length)	The maximum length of the data bar, as a percentage of the cell width. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
minLength (Minimum Length)	The minimum length of the data bar, as a percentage of the cell width. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
showValue (Show Values)	Indicates whether to show the values of the cells on which this data bar is applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DataBar">
   <sequence>
      <element name="cfvo" type="CT_Cfvo" minOccurs="2" maxOccurs="2"/>
      <element name="color" type="CT_Color" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="minLength" type="xsd:unsignedInt" use="optional" default="10"/>
   <attribute name="maxLength" type="xsd:unsignedInt" use="optional" default="90"/>
   <attribute name="showValue" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

3.3.1.27 dataConsolidate (Data Consolidate)

Data consolidation settings. The dataRefs are the set of source ranges containing data to consolidate. The function indicates the function that shall be used to consolidate the data.

[*Example*:

This example demonstrates consolidating the ranges A1:C1 and A3:C3 by using the 'count' function.

```
<dataConsolidate function="count">
  <dataRefs count="2">
    <dataRef ref="A1:C1" sheet="Sheet1"/>
    <dataRef ref="A3:C3" sheet="Sheet1"/>
  </dataRefs>
</dataConsolidate>
```

end example]

Parent Elements
worksheet (§3.3.1.96)

Child Elements	Subclause
dataRefs (Data Consolidation References)	§3.3.1.29

Attributes	Description
function (Function Index)	Indicates which function to use when consolidating the ranges. The possible values for this attribute are defined by the ST_DataConsolidateFunction simple type (§3.18.18).
leftLabels (Use Left Column Labels)	Use labels in left column. Both leftLabels and topLabels can be true at the same time. The possible values for this attribute are defined by the XML Schema boolean datatype.
link (Link)	Create links to source data. The possible values for this attribute are defined by the XML Schema boolean datatype.
topLabels (Labels In Top Row)	Use labels in top row. Both leftLabels and topLabels can be true at the same time. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DataConsolidate">
   <sequence>
      <element name="dataRefs" type="CT_DataRefs" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="function" type="ST_DataConsolidateFunction" use="optional" default="sum"/>
   <attribute name="leftLabels" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="topLabels" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="link" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.3.1.28 dataRef (Data Consolidation Reference)

A single data consolidate reference. One dataRef shall use either name or sheet & ref, but not both on the same dataRef.

Parent Elements
dataRefs (§3.3.1.29)

Attributes	Description
id (relationship Id) Namespace: .../officeDocument/2006/relationships	Used only when the source range is external to this workbook. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
name (Named Range)	Named range, either in this workbook or the external workbook referenced by r:Id. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
ref (Reference)	Cell range. The possible values for this attribute are defined by the ST_Ref simple type (§3.18.64).
sheet (Sheet Name)	Sheet name. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DataRef">
   <attribute name="ref" type="ST_Ref" use="optional"/>
   <attribute name="name" type="ST_Xstring" use="optional"/>
   <attribute name="sheet" type="ST_Xstring" use="optional"/>
   <attribute ref="r:id" use="optional"/>
</complexType>
```

3.3.1.29 dataRefs (Data Consolidation References)

Data consolidate reference collection.

Parent Elements
dataConsolidate (§3.3.1.27)

Child Elements	Subclause
dataRef (Data Consolidation Reference)	§3.3.1.28

Attributes	Description
count (Data Consolidation Reference Count)	Count of data consolidate references. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DataRefs">
   <sequence>
      <element name="dataRef" type="CT_DataRef" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.3.1.30 dataValidation (Data Validation)

A single item of data validation defined on a range of the worksheet.

Parent Elements
dataValidations (§3.3.1.31)

Child Elements	Subclause
formula1 (Formula 1)	§3.3.1.41
formula2 (Formula 2)	§3.3.1.42

Attributes	Description
allowBlank (Allow Blank)	A boolean value indicating whether the data validation treats empty or blank entries as valid. '1' means empty entries are OK and do not violate the validation constraints. The possible values for this attribute are defined by the XML Schema boolean datatype.
error (Error Message)	Message text of error alert. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
errorStyle (Data Validation Error Style)	The style of error alert used for this data validation. The possible values for this attribute are defined by the ST_DataValidationErrorStyle simple type (§3.18.19).
errorTitle (Error Alert Text)	Title bar text of error alert. The possible values for this attribute are defined by the ST_Xstring simple type

Attributes	Description
	(§3.18.96).
imeMode (IME Mode Enforced)	The IME (input method editor) mode enforced by this data validation. Only applies for these languages: • Chinese Simplified • Chinese Traditional • Japanese • Korean When imeMode is set, the input for the cell can be restricted to specific sets of characters, as specified by the value of imeMode. See the simple type referenced below for additional details. When imeMode is set but the application's language is not one of the languages listed above, then the default value is noControl. The possible values for this attribute are defined by the ST_DataValidationImeMode simple type (§3.18.20).
operator (Operator)	The relational operator used with this data validation. The possible values for this attribute are defined by the ST_DataValidationOperator simple type (§3.18.21).
prompt (Input Prompt)	Message text of input prompt. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
promptTitle (Prompt Title)	Title bar text of input prompt. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
showDropDown (Show Drop Down)	A boolean value indicating whether to display the dropdown combo box for a list type data validation. The possible values for this attribute are defined by the XML Schema boolean datatype.
showErrorMessage (Show Error Message)	A boolean value indicating whether to display the error alert message when an invalid value has been entered, according to the criteria specified. The possible values for this attribute are defined by the XML Schema boolean datatype.
showInputMessage (Show Input Message)	A boolean value indicating whether to display the input prompt message. The possible values for this attribute are defined by the XML Schema boolean datatype.
sqref (Sequence of References)	Range over which data validation is applied. The possible values for this attribute are defined by the ST_Sqref simple type (§3.18.78).

Attributes	Description
type (Data Validation Type)	The type of data validation. The possible values for this attribute are defined by the ST_DataValidationType simple type (§3.18.22).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DataValidation">
   <sequence>
      <element name="formula1" type="ST_Formula" minOccurs="0" maxOccurs="1"/>
      <element name="formula2" type="ST_Formula" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="type" type="ST_DataValidationType" use="optional" default="none"/>
   <attribute name="errorStyle" type="ST_DataValidationErrorStyle" use="optional" default="stop"/>
   <attribute name="imeMode" type="ST_DataValidationImeMode" use="optional" default="noControl"/>
   <attribute name="operator" type="ST_DataValidationOperator" use="optional" default="between"/>
   <attribute name="allowBlank" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="showDropDown" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="showInputMessage" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="showErrorMessage" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="errorTitle" type="ST_Xstring" use="optional"/>
   <attribute name="error" type="ST_Xstring" use="optional"/>
   <attribute name="promptTitle" type="ST_Xstring" use="optional"/>
   <attribute name="prompt" type="ST_Xstring" use="optional"/>
   <attribute name="sqref" type="ST_Sqref" use="required"/>
</complexType>
```

3.3.1.31 dataValidations (Data Validations)

This collection expresses all data validation information for cells in a sheet which have data validation features applied.

Data validation is used to specify constaints on the type of data that can be entered into a cell. Additional UI can be provided to help the user select valid values (e.g., a dropdown control on the cell or hover text when the cell is active), and to help the user understand why a particular entry was considered invalid (e.g., alerts and messages).

Various data types can be selected, and logical operators (e.g., greater than, less than, equal to, etc) can be used. Additionally, instead of specifying an explicit set of values that are valid, a cell or range reference may be used.

An input message can be specified to help the user know what kind of value is expected, and a warning message (and warning type) can be specified to alert the user when they've entered invalid data.

[*Example*:

```
<dataValidations count="1">
  <dataValidation type="whole" errorStyle="warning" operator="greaterThan"
    showInputMessage="1" showErrorMessage="1" errorTitle="Invalid Data"
    error="The value must be a whole number greater than 0."
    promptTitle="Whole Number"
    prompt="Please enter a whole number greater than 0." sqref="A1">
    <formula1>0</formula1>
  </dataValidation>
</dataValidations>
```

end example]

Parent Elements
worksheet (§3.3.1.96)

Child Elements	Subclause
dataValidation (Data Validation)	§3.3.1.30

Attributes	Description
count (Data Validation Item Count)	The expected number of data validation items for this worksheet. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
disablePrompts (Disable Prompts)	A boolean value indicating whether all input prompts for the worksheet are disabled. The possible values for this attribute are defined by the XML Schema boolean datatype.
xWindow (Top Left Corner (X Coodrinate))	The x-coordinate (relative to window) of top-left corner of the data validation input prompt (textbox). This is per sheet, not per cell. Units in pixels. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
yWindow (Top Left Corner (Y Coordinate))	The y-coordinate (relative to window) of top-left corner of the data validation input prompt (textbox). This is per sheet, not per cell. Units in pixels. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DataValidations">
   <sequence>
      <element name="dataValidation" type="CT_DataValidation" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="disablePrompts" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="xWindow" type="xsd:unsignedInt" use="optional"/>
   <attribute name="yWindow" type="xsd:unsignedInt" use="optional"/>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.3.1.32 dialogsheet (Dialog Sheet)

This is the root element for Sheet Parts of type 'dialogsheet'.

Parent Elements
Root element of SpreadsheetML Dialogsheet part

Child Elements	Subclause
customSheetViews (Custom Sheet Views)	§3.3.1.25
drawing (Drawing)	§3.3.1.34
extLst (Future Feature Data Storage Area)	§3.2.10
headerFooter (Header Footer Settings)	§3.3.1.43
legacyDrawing (Legacy Drawing Reference)	§3.3.1.51
legacyDrawingHF (Legacy Drawing Reference in Header Footer)	§3.3.1.52
oleObjects (Embedded Objects)	§3.3.1.58
pageMargins (Page Margins)	§3.3.1.60
pageSetup (Page Setup Settings)	§3.3.1.61
printOptions (Print Options)	§3.3.1.68
sheetFormatPr (Sheet Format Properties)	§3.3.1.78
sheetPr (Sheet Properties)	§3.3.1.79
sheetProtection (Sheet Protection Options)	§3.3.1.81
sheetViews (Sheet Views)	§3.3.1.85

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Dialogsheet">
   <sequence>
      <element name="sheetPr" minOccurs="0" type="CT_SheetPr"/>
      <element name="sheetViews" minOccurs="0" type="CT_SheetViews"/>
      <element name="sheetFormatPr" minOccurs="0" type="CT_SheetFormatPr"/>
      <element name="sheetProtection" type="CT_SheetProtection" minOccurs="0" maxOccurs="1"/>
      <element name="customSheetViews" minOccurs="0" type="CT_CustomSheetViews"/>
      <element name="printOptions" minOccurs="0" type="CT_PrintOptions"/>
      <element name="pageMargins" minOccurs="0" type="CT_PageMargins"/>
      <element name="pageSetup" minOccurs="0" type="CT_PageSetup"/>
      <element name="headerFooter" minOccurs="0" type="CT_HeaderFooter"/>
      <element name="drawing" minOccurs="0" type="CT_Drawing"/>
      <element name="legacyDrawing" minOccurs="0" type="CT_LegacyDrawing"/>
      <element name="legacyDrawingHF" type="CT_LegacyDrawing" minOccurs="0" maxOccurs="1"/>
      <element name="oleObjects" type="CT_OleObjects" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
</complexType>
```

3.3.1.33 dimension (Worksheet Dimensions)

This element specifies the used range of the worksheet. It specifies the row and column bounds of used cells in the worksheet. This is optional and is not required. Used cells include cells with formulas, text content, and cell formatting. When an entire column is formatted, only the first cell in that column is considered used.

[*Example*:

```
<dimension ref="A1:C2"/>
```

end example]

Parent Elements
worksheet (§3.3.1.96)

Attributes	Description
ref (Reference)	The row and column bounds of all cells in this worksheet. Corresponds to the range that would contain all c elements written under sheetData. Does not support whole column or whole row reference notation. The possible values for this attribute are defined by the ST_Ref simple type (§3.18.64).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SheetDimension">
   <attribute name="ref" type="ST_Ref" use="required"/>
</complexType>
```

3.3.1.34 drawing (Drawing)

This element indicates that the sheet contains drawing components built on the drawingML platform. The relationship Id references the part containing the drawingML definitions.

Parent Elements
chartsheet (§3.3.1.11); dialogsheet (§3.3.1.32); worksheet (§3.3.1.96)

Attributes	Description
id (Relationship id) Namespace: .../officeDocument/2006/relationships	Relationship Id referencing a part containing drawingML definitions for this worksheet. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Drawing">
   <attribute ref="r:id" use="required"/>
</complexType>
```

3.3.1.35 evenFooter (Even Page Footer)

Even page footer value. Corresponds to even printed pages. Even page(s) in the sheet may not be printed, for example, if the print area is specified to be a range such that it falls outside an even page's scope.

If no even footer is specified, then the odd footer's value is assumed for even page footers. See the evenHeader element (§3.3.1.36) description for full discussion of value content.

The possible values for this element are defined by the ST_Xstring simple type (§3.18.96).

Parent Elements
headerFooter (§3.3.1.43)

3.3.1.36 evenHeader (Even Page Header)

Even page header value. Corresponds to even printed pages. Even page(s) in the sheet may not be printed, for example, if the print area is specified to be a range such that it falls outside an even page's scope.

If no even header is specified, then odd header value is assumed for even page headers.

Header/Footer Formatting Syntax

There are a number of formatting codes that can be written inline with the actual header / footer text, which affect the formatting in the header or footer.

[*Example*:

This example shows the text "Center **Bold** Header" on the first line (center section), and the date on the second line (center section).

```
<headerFooter>
  <oddHeader>&amp;CCenter &amp;"-,Bold"Bold
    &amp;"-,Regular"Header_x000A_&amp;D</oddHeader>
</headerFooter>
```

end example]

General Rules:

There is no required order in which these codes must appear.

The first occurrence of the following codes turns the formatting ON, the second occurrence turns it OFF again:

- strikethrough
- superscript
- subscript

Superscript and subscript cannot both be ON at same time. Whichever comes first wins and the other is ignored, while the first is ON.

&L - code for "left section" (there are three header / footer locations, "left", "center", and "right"). When two or more occurrences of this section marker exist, the contents from all markers are concatenated, in the order of appearance, and placed into the left section.

&P - code for "current page #"

&N - code for "total pages"

&*font size* - code for "text font size", where *font size* is a font size in points.

&K - code for "text font color"

RGB Color is specified as RRGGBB

Theme Color is specifed as TTSNN where TT is the theme color Id, S is either "+" or "-" of the tint/shade value, NN is the tint/shade value.

&S - code for "text strikethrough" on / off

&X - code for "text super script" on / off

&Y - code for "text subscript" on / off

&C - code for "center section". When two or more occurrences of this section marker exist, the contents from all markers are concatenated, in the order of appearance, and placed into the center section.

&D - code for "date"

&T - code for "time"

&G - code for "picture as background"

&U - code for "text single underline"

&E - code for "double underline"

&R - code for "right section". When two or more occurrences of this section marker exist, the contents from all markers are concatenated, in the order of appearance, and placed into the right section.

&Z - code for "this workbook's file path"

&F - code for "this workbook's file name"

&A - code for "sheet tab name"

&+ - code for add to page #.

&- - code for subtract from page #.

&"*font name*,*font type*" - code for "text font name" and "text font type", where *font name* and *font type* are strings specifying the name and type of the font, separated by a comma. When a hyphen appears in *font name*, it means "none specified". Both of *font name* and *font type* can be localized values.

&"-,Bold" - code for "bold font style"

&B - also means "bold font style".

&"-,Regular" - code for "regular font style"

&"-,Italic" - code for "italic font style"

&I - also means "italic font style"

&"-,Bold Italic" code for "bold italic font style"

&O - code for "outline style"

&H - code for "shadow style"

The possible values for this element are defined by the ST_Xstring simple type (§3.18.96).

Parent Elements
headerFooter (§3.3.1.43)

3.3.1.37 f (Formula)

Formula for the cell. The formula expression is contained in the character node of this element.

[*Example*:

```
<f>SUM(C4:E4)</f>
```

end example]

The possible values for the t attribute have type ST_CellFormulaType, and are as follows:

- `array` (Array Entered)
- `dataTable` (Table Formula, see below)
- `normal` (Normal)
- `shared` (Shared Formula)

A *data table* is a range of cells that shows how changing certain values in one or more formulas affects the results of those formulas. A data table provides a shortcut for calculating multiple versions in one operation, and a way to view and compare the results of all of the different variations together on a worksheet.

Both one- and two-input variable data tables can be created (see attribute dt2D). [*Example*: A one-input variable data table might be used to see how different interest rates affect a monthly mortgage payment, while a two-input variable data table might be used to show how different interest rates and loan terms will affect the mortgage payment. *end example*]

Data tables shall be recalculated whenever a worksheet is recalculated.

In a one-input variable data table, values are listed either down a column (column-oriented) or across a row (row-oriented) (see attribute dtr).

Formulas that are used in a one-input variable data table shall refer to an input cell (see attribute r1), the cell in which each input value from a data table is substituted. Any cell on a worksheet can be the input cell. Although the input cell does not need to be part of the data table, the formulas in data tables shall refer to that input cell.

Two-input variable data tables use only one formula with two lists of input values. The formula shall refer to two input cells (see attributes r1 and r2).

The top-left cell in the data table is called the *master cell*.

Parent Elements
c (§3.3.1.3); nc (§3.11.1.3); oc (§3.11.1.5)

Attributes	Description
aca (Always	`true` indicates that this formula is an array formula and the entire array shall be

Attributes	Description
Calculate Array)	calculated in full. If `false` the individual cells of the array shall be calculated as needed. [*Note*: The primary case where an array formula shall be calcuated in part instead of in full is when some cells in the array depend on other cells that are semi-calculated, e.g., contains the function =RAND(). *end note*] The possible values for this attribute are defined by the XML Schema boolean datatype.
bx (Assigns Value to Name)	Specifies that this formula assigns a value to a name. The possible values for this attribute are defined by the XML Schema boolean datatype.
ca (Calculate Cell)	Indicates that this formula needs to be recalculated the next time calculation is performed. For example, this is always set on volatile functions, like =RAND(), and circular references. The possible values for this attribute are defined by the XML Schema boolean datatype.
del1 (Input 1 Deleted)	Whether the first input cell for data table has been deleted. Applies to data table formula only. Written on master cell of data table formula only. The possible values for this attribute are defined by the XML Schema boolean datatype.
del2 (Input 2 Deleted)	Whether the second input cell for data table has been deleted. Applies to data table formula only. Written on master cell of data table formula only. The possible values for this attribute are defined by the XML Schema boolean datatype.
dt2D (Data Table 2-D)	Data table is two-dimentional. Only applies to the data tables function. Written on master cell of data table formula only. The possible values for this attribute are defined by the XML Schema boolean datatype.
dtr (Data Table Row)	`true` if one-dimentional data table is a row, otherwise it's a column. Only applies to the data tables function. Written on master cell of data table formula only. The possible values for this attribute are defined by the XML Schema boolean datatype.
r1 (Data Table Cell 1)	First input cell for data table. Only applies to the data tables array function "TABLE()". Written on master cell of data table formula only. The possible values for this attribute are defined by the ST_CellRef simple type (§3.18.8).
r2 (Input Cell 2)	Second input cell for data table when dt2D is '1'. Only applies to the data tables array function "TABLE()".Written on master cell of data table formula only. The possible values for this attribute are defined by the ST_CellRef simple type (§3.18.8).
ref (Range of Cells)	Range of cells which the formula applies to. Only required for shared formula, array formula or data table. Only written on the master formula, not subsequent formula's belonging to the same shared group, array, or data table.

Attributes	Description
	The possible values for this attribute are defined by the ST_Ref simple type (§3.18.64).
si (Shared Group Index)	Optional attribute to optimize load performance by sharing formulas. When a formula is a shared formula (t value is shared) then this value indicates the group to which this particular cell's formula belongs. The first formula in a group of shared formulas is saved in the f element. This is considered the 'master' formula cell. Subsequent cells sharing this formula need not have the formula written in their f element. Instead, the attribute si value for a particular cell is used to figure what the formula expression should be based on the cell's relative location to the master formula cell. A cell is shared only when si is used and t is shared. The formula expression for a cell that is specified to be part of a shared formula (and is not the master) shall be ignored, and the master formula shall override. If a master cell of a shared formula range specifies that a particular cell is part of the shared formula range, and that particular cell does not use the si and t attributes to indicate that it is shared, then the particular cell's formula shall override the shared master formula. If this cell occurs in the middle of a range of shared formula cells, the earlier and later formulas shall continue sharing the master formula, and the cell in question shall not share the formula of the master cell formula. Loading and handling of a cell and formula using an si attribute and whose t value is shared, located outside the range specified in the master cell associated with the si group, is implementation defined. Master cell references on the same sheet shall not overlap with each other. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
t (Formula Type)	Type of formula. The possible values for this attribute are defined by the ST_CellFormulaType simple type (§3.18.7).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CellFormula">
   <simpleContent>
      <extension base="ST_Formula">
         <attribute name="t" type="ST_CellFormulaType" use="optional" default="normal"/>
         <attribute name="aca" type="xsd:boolean" use="optional" default="false"/>
         <attribute name="ref" type="ST_Ref" use="optional"/>
         <attribute name="dt2D" type="xsd:boolean" use="optional" default="false"/>
         <attribute name="dtr" type="xsd:boolean" use="optional" default="false"/>
         <attribute name="del1" type="xsd:boolean" use="optional" default="false"/>
         <attribute name="del2" type="xsd:boolean" use="optional" default="false"/>
         <attribute name="r1" type="ST_CellRef" use="optional"/>
         <attribute name="r2" type="ST_CellRef" use="optional"/>
         <attribute name="ca" type="xsd:boolean" use="optional" default="false"/>
         <attribute name="si" type="xsd:unsignedInt" use="optional"/>
         <attribute name="bx" type="xsd:boolean" use="optional" default="false"/>
      </extension>
   </simpleContent>
</complexType>
```

3.3.1.38 firstFooter (First Page Footer)

First page footer content. Only used when `headerFooter@differentFirst` is '1'. Corresponds to first printed page. The first logical page in the sheet may not be printed, for example, if the print area is specified to be a range such that it falls outside the first page's scope.

See evenHeader (§3.3.1.36) description for full discussion of value content.

The possible values for this element are defined by the ST_Xstring simple type (§3.18.96).

Parent Elements
headerFooter (§3.3.1.43)

3.3.1.39 firstHeader (First Page Header)

First page header content. Only used when `headerFooter@differentFirst` is '1'. Corresponds to first printed page. The first logical page in the sheet may not be printed, for example, if the print area is specified to be a range such that it falls outside the first page's scope.

See evenHeader (§3.3.1.36) description for full discussion of value content.

The possible values for this element are defined by the ST_Xstring simple type (§3.18.96).

Parent Elements
headerFooter (§3.3.1.43)

3.3.1.40 formula (Formula)

The content of this element is a formula whose calculated value specifies the criteria for the conditional formatting rule.

The possible values for this element are defined by the ST_Formula simple type (§3.18.36).

Parent Elements
cfRule (§3.3.1.9); rdn (§3.11.1.13)

3.3.1.41 formula1 (Formula 1)

The first formula in the DataValidation dropdown. It used as a bounds for 'between' and 'notBetween' relational operators, and the only formula used for other relational operators (equal, notEqual, lessThan, lessThanOrEqual, greaterThan, greaterThanOrEqual), or for custom or list type data validation. The content can be a formula or a constant or a list series (comma separated values).

The possible values for this element are defined by the ST_Formula simple type (§3.18.36).

Parent Elements
dataValidation (§3.3.1.30)

3.3.1.42 formula2 (Formula 2)

The second formula in the DataValidation dropdown. It used as a bounds for 'between' and 'notBetween' relational operators only.

The possible values for this element are defined by the ST_Formula simple type (§3.18.36).

Parent Elements
dataValidation (§3.3.1.30)

3.3.1.43 headerFooter (Header Footer Settings)

Header and footer settings.

[*Example*:

This example demonstrates "Header" at the top and "Footer" at the bottom of a page.

```
<headerFooter>
  <oddHeader>&amp;CHeader</oddHeader>
  <oddFooter>&amp;CFooter</oddFooter>
</headerFooter>
```

end example]

The tokens in the header & footer elements can be localized. An application may decide which locales are supported. Even when a locale is not supported, the header and footer text must be loaded, and only the formatting is discarded.

Parent Elements

Parent Elements
chartsheet (§3.3.1.11); customSheetView (§3.3.1.23); customSheetView (§3.3.1.22); dialogsheet (§3.3.1.32); worksheet (§3.3.1.96)

Child Elements	Subclause
evenFooter (Even Page Footer)	§3.3.1.35
evenHeader (Even Page Header)	§3.3.1.36
firstFooter (First Page Footer)	§3.3.1.38
firstHeader (First Page Header)	§3.3.1.39
oddFooter (Odd Page Footer)	§3.3.1.55
oddHeader (Odd Header)	§3.3.1.56

Attributes	Description
alignWithMargins (Align Margins)	Align header footer margins with page margins. When `true`, as left/right margins grow and shrink, the header and footer edges stay aligned with the margins. When `false`, headers and footers are aligned on the paper edges, regardless of margins. The possible values for this attribute are defined by the XML Schema boolean datatype.
differentFirst (Different First Page)	Different first page header and footer. When `true` then firstHeader and firstFooter specify first page header and footer values. If `false` and firstHeader / firstFooter are present, they are ignored. The possible values for this attribute are defined by the XML Schema boolean datatype.
differentOddEven (Different Odd Even Header Footer)	Different odd and even page headers and footers. When `true` then oddHeader / oddFooter and evenHeader / evenFooter specify page header and footer values for odd and even pages. If `false` then oddHeader / oddFooter is used, even when evenHeader / evenFooter are present. The possible values for this attribute are defined by the XML Schema boolean datatype.
scaleWithDoc (Scale Header & Footer With Document)	Scale header and footer with document scaling. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HeaderFooter">
   <sequence>
      <element name="oddHeader" type="ST_Xstring" minOccurs="0" maxOccurs="1"/>
      <element name="oddFooter" type="ST_Xstring" minOccurs="0" maxOccurs="1"/>
      <element name="evenHeader" type="ST_Xstring" minOccurs="0" maxOccurs="1"/>
      <element name="evenFooter" type="ST_Xstring" minOccurs="0" maxOccurs="1"/>
      <element name="firstHeader" type="ST_Xstring" minOccurs="0" maxOccurs="1"/>
      <element name="firstFooter" type="ST_Xstring" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="differentOddEven" type="xsd:boolean" default="false"/>
   <attribute name="differentFirst" type="xsd:boolean" default="false"/>
   <attribute name="scaleWithDoc" type="xsd:boolean" default="true"/>
   <attribute name="alignWithMargins" type="xsd:boolean" default="true"/>
</complexType>
```

3.3.1.44 hyperlink (Hyperlink)

A single hyperlink

Parent Elements
hyperlinks (§3.3.1.45)

Attributes	Description
display (Display String)	Display string, if different from string in string table. This is a property on the hyperlink object, but does not need to appear in the spreadsheet application UI. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
id (Relationship Id) Namespace: .../officeDocument/2006/relationships	Relationship Id in this sheet's relationships part, expressing the target location of the resource. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
location (Location)	Location within target. If target is a workbook (or this workbook) this shall refer to a sheet and cell or a defined name. Can also be an HTML anchor if target is HTML file. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
ref (Reference)	Cell location of hyperlink on worksheet. The possible values for this attribute are defined by the ST_Ref simple type (§3.18.64).
tooltip (Tool Tip)	This is additional text to help the user understand more about the hyperlink. This can be displayed as hover text when the mouse is over the link, for example.

Attributes	Description
	The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Hyperlink">
   <attribute name="ref" type="ST_Ref" use="required"/>
   <attribute ref="r:id" use="optional"/>
   <attribute name="location" type="ST_Xstring" use="optional"/>
   <attribute name="tooltip" type="ST_Xstring" use="optional"/>
   <attribute name="display" type="ST_Xstring" use="optional"/>
</complexType>
```

3.3.1.45 hyperlinks (Hyperlinks)

Collection of hyperlinks.

[*Example*:

This example shows a hyperlink in cell A11, with hover text displaying "Search Page". The relationship Id references a relationship from the sheet to the external target resource.

```
<hyperlinks>
  <hyperlink ref="A11" r:id="rId1" tooltip="Search Page"/>
</hyperlinks>
```

end example]

Parent Elements
worksheet (§3.3.1.96)

Child Elements	Subclause
hyperlink (Hyperlink)	§3.3.1.44

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Hyperlinks">
   <sequence>
      <element name="hyperlink" type="CT_Hyperlink" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.3.1.46 iconSet (Icon Set)

Describes an icon set conditional formatting rule.

[*Example*: This example demonstrates the "3Arrows" style of icons. The first icon in the set shall be shown if the cell's value is less than the 33rd percentile. The second icon in the set shall be shown if the cell's value is less

than the 67th percentile, and greater than or equal to the 33rd percentile. The third icon in the set shall be shown if the cell's value is greater than or equal to the 67th percentile.

```
<iconSet iconSet="3Arrows">
  <cfvo type="percentile" val="0"/>
  <cfvo type="percentile" val="33"/>
  <cfvo type="percentile" val="67"/>
</iconSet>
```

end example]

Parent Elements
cfRule (§3.3.1.9)

Child Elements	Subclause
cfvo (Conditional Format Value Object)	§3.3.1.10

Attributes	Description
iconSet (Icon Set)	The icon set to display. The possible values for this attribute are defined by the ST_IconSetType simple type (§3.18.44).
percent (Percent)	Indicates whether the thresholds indicate percentile values, instead of number values. The possible values for this attribute are defined by the XML Schema boolean datatype.
reverse (Reverse Icons)	If '1', reverses the default order of the icons in this icon set. The possible values for this attribute are defined by the XML Schema boolean datatype.
showValue (Show Value)	Indicates whether to show the values of the cells on which this icon set is applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_IconSet">
   <sequence>
      <element name="cfvo" type="CT_Cfvo" minOccurs="2" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="iconSet" type="ST_IconSetType" use="optional" default="3TrafficLights1"/>
   <attribute name="showValue" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="percent" type="xsd:boolean" default="true"/>
   <attribute name="reverse" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.3.1.47 ignoredError (Ignored Error)

A single ignored error type for a range of cells.

A cell is considered to have an error condition when it meets one of the conditions specified in the attribute descriptions below. For example, if a cell is formatted as text but contains a numeric value, this is considered to be a potential error because the number won't be treated as a number, for example, in calculations.

Note that this simply a guess by the implementing application, and a recommendation to the user. Cells with the errors specified below may have perfectly valid reasons for being in such a state, for example a cell formatted as text which contains numeric Postal Codes or Order numbers. It is useful to format these cells as text so that leading zeros remain as part of the value instead of being removed.

An `<ignoreError>` element is not written in the file unless the user has specifically reviewed the error and decided to keep the cell state as it is, and no longer wishes to be alerted about it for this cell. This can be helpful for the application to decide which errors should be surfaced to the user vs kept quiet because the user doesn't want these to be surfaced (e.g., because they are legitimate cell states).

[*Example*: This example shows that cells A1 and B2 both contain numbers stored as text, and this error has been reviewed and specifically flagged to be no longer surfaced as an error to the user.

```
<ignoredErrors>
  <ignoredError sqref="A1 B2" numberStoredAsText="1"/>
</ignoredErrors>
```

end example]

Note: more than one kind of error can exist on a cell. These flags are not mutually exclusive.

Parent Elements
ignoredErrors (§3.3.1.48)

Attributes	Description
calculatedColumn (Calculated Column)	Ignore errors when cells contain a value different from a calculated column formula. In other words, for a calculated column, a cell in that column is considered to have an error if its formula is different from the calculated column formula, or doesn't contain a formula at all. The possible values for this attribute are defined by the XML Schema boolean datatype.
emptyCellReference (Empty Cell Reference)	Ignore errors when formulas refer to empty cells. The possible values for this attribute are defined by the XML Schema boolean datatype.
evalError (Evaluation Error)	Ignore errors when cells contain formulas that result in an error.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema boolean datatype.
formula (Formula)	Ignore errors when a formula in a region of your worksheet differs from other formulas in the same region. The possible values for this attribute are defined by the XML Schema boolean datatype.
formulaRange (Formula Range)	Ignore errors when formulas omit certain cells in a region. The possible values for this attribute are defined by the XML Schema boolean datatype.
listDataValidation (List Data Validation)	Ignore errors when a cell's value in a Table does not comply with the Data Validation rules specified. The possible values for this attribute are defined by the XML Schema boolean datatype.
numberStoredAsText (Number Stored As Text)	Ignore errors when numbers are formatted as text or are preceded by an apostrophe. The possible values for this attribute are defined by the XML Schema boolean datatype.
sqref (Sequence of References)	Reference to a range of cells that have this ignored error. The possible values for this attribute are defined by the ST_Sqref simple type (§3.18.78).
twoDigitTextYear (Two Digit Text Year)	Ignore errors when formulas contain text formatted cells with years represented as 2 digits. The possible values for this attribute are defined by the XML Schema boolean datatype.
unlockedFormula (Unlocked Formula)	Ignore errors when unlocked cells contain formulas. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_IgnoredError">
    <attribute name="sqref" type="ST_Sqref" use="required"/>
    <attribute name="evalError" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="twoDigitTextYear" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="numberStoredAsText" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="formula" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="formulaRange" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="unlockedFormula" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="emptyCellReference" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="listDataValidation" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="calculatedColumn" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.3.1.48 ignoredErrors (Ignored Errors)

A collection of ignored errors, by cell range.

Parent Elements

Parent Elements
worksheet (§3.3.1.96)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
ignoredError (Ignored Error)	§3.3.1.47

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_IgnoredErrors">
   <sequence>
      <element name="ignoredError" type="CT_IgnoredError" minOccurs="1" maxOccurs="unbounded"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

3.3.1.49 inputCells (Input Cells)

This collection describes each input cell for the scenario.

Parent Elements
scenario (§3.3.1.73)

Attributes	Description
deleted (Deleted)	Input cell was deleted. This input cell shall be present in the file format, but shall not be presented to the user as part of the scenario inputs, nor run as part of the scenario. The possible values for this attribute are defined by the XML Schema boolean datatype.
numFmtId (Number Format Id)	This number format Id is used only when displaying the scenario manager input UI, and is used to properly format for display the cached input values (see val attribute) for the scenario. The possible values for this attribute are defined by the ST_NumFmtId simple type (§3.18.49).
r (Reference)	Cell reference indicating the input cell address. The possible values for this attribute are defined by the ST_CellRef simple type (§3.18.8).
undone (Undone)	Cell's deletion was undone. When true the r (reference) value shall not adjust in response to the cell moving due to row / column insert or delete,or cell move. The possible values for this attribute are defined by the XML Schema boolean datatype.
val (Value)	Value that should be used for the cell when this scenario is run.

Attributes	Description
	Note: val does not need a corresponding data type, the value is put into the cell when the scenario is run. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_InputCells">
   <attribute name="r" type="ST_CellRef" use="required"/>
   <attribute name="deleted" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="undone" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="val" type="ST_Xstring" use="required"/>
   <attribute name="numFmtId" type="ST_NumFmtId" use="optional"/>
</complexType>
```

3.3.1.50 is (Rich Text Inline)

This element allows for strings to be expressed directly in the cell definition instead of implementing the shared string table.

[*Example*:

```
<c r="A1">
  <is>
    <t>String</t>
  </is>
</c>
```

end example]

Parent Elements
c (§3.3.1.3); nc (§3.11.1.3); oc (§3.11.1.5)

Child Elements	Subclause
phoneticPr (Phonetic Properties)	§3.4.3
r (Rich Text Run)	§3.4.4
rPh (Phonetic Run)	§3.4.6
t (Text)	§3.4.12

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Rst">
   <sequence>
      <element name="t" type="ST_Xstring" minOccurs="0" maxOccurs="1"/>
      <element name="r" type="CT_RElt" minOccurs="0" maxOccurs="unbounded"/>
      <element name="rPh" type="CT_PhoneticRun" minOccurs="0" maxOccurs="unbounded"/>
      <element name="phoneticPr" minOccurs="0" maxOccurs="1" type="CT_PhoneticPr"/>
   </sequence>
</complexType>
```

3.3.1.51 legacyDrawing (Legacy Drawing Reference)

This element is present when the sheet contains drawing shapes defined by VML. In this case, the element contains an explicit relationship whose ID points to the part containing the VML definitions.

[*Example*:

```
<drawing r:id="rId1"/>
```

end example]

Parent Elements
chartsheet (§3.3.1.11); dialogsheet (§3.3.1.32); worksheet (§3.3.1.96)

Attributes	Description
id (Relationship Id) Namespace: .../officeDocument/2006/relationships	This value references a relationship Id for the sheet. The relationship shall point to the part containing the VML definition. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LegacyDrawing">
   <attribute ref="r:id" use="required"/>
</complexType>
```

3.3.1.52 legacyDrawingHF (Legacy Drawing Reference in Header Footer)

This element specifies the explicit relationship to the part containing the VML defining pictures rendered in the header / footer of the sheet.

Parent Elements
chartsheet (§3.3.1.11); dialogsheet (§3.3.1.32); worksheet (§3.3.1.96)

Attributes	Description

Attributes	Description
id (Relationship Id) Namespace: .../officeDocument/2006/relationships	This value references a relationship Id for the sheet. The relationship shall point to the part containing the VML definition. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LegacyDrawing">
   <attribute ref="r:id" use="required"/>
</complexType>
```

3.3.1.53 mergeCell (Merged Cell)

A single merged cell

Parent Elements
mergeCells (§3.3.1.54)

Attributes	Description
ref (Reference)	Range defined by merge cell. The possible values for this attribute are defined by the ST_Ref simple type (§3.18.64).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MergeCell">
   <attribute name="ref" type="ST_Ref" use="required"/>
</complexType>
```

3.3.1.54 mergeCells (Merge Cells)

This collection expresses all the merged cells in the sheet.

[*Example*:

This example shows that three ranges are merged. The formatting and content for the merged range is always stored in the top left cell.

```
<mergeCells>
  <mergeCell ref="C2:F2"/>
  <mergeCell ref="B19:C20"/>
  <mergeCell ref="E19:G19"/>
</mergeCells>
```

end example]

Parent Elements
worksheet (§3.3.1.96)

Child Elements	Subclause
mergeCell (Merged Cell)	§3.3.1.53

Attributes	Description
count (Count)	A count of merged cell collections. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MergeCells">
   <sequence>
      <element name="mergeCell" type="CT_MergeCell" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.3.1.55 oddFooter (Odd Page Footer)

Odd page footer value. Corresponds to odd printed pages. Odd page(s) in the sheet may not be printed, for example, if the print area is specified to be a range such that it falls outside an odd page's scope.

See evenHeader (§3.3.1.36) description for full discussion of value content.

The possible values for this element are defined by the ST_Xstring simple type (§3.18.96).

Parent Elements
headerFooter (§3.3.1.43)

3.3.1.56 oddHeader (Odd Header)

Odd page header value. Corresponds to odd printed pages. Odd page(s) in the sheet may not be printed, for example, if the print area is specified to be a range such that it falls outside an odd page's scope.

See evenHeader (§3.3.1.36) description for full discussion of value content.

The possible values for this element are defined by the ST_Xstring simple type (§3.18.96).

Parent Elements
headerFooter (§3.3.1.43)

3.3.1.57 oleObject (Embedded Object)

Information for an individual embedded object.

Parent Elements
oleObjects (§3.3.1.58)

Attributes	Description
autoLoad (Auto Load)	Specifies whether the host application for the embedded object shall be called to load the object data automatically when the parent workbook is opened. The possible values for this attribute are defined by the XML Schema boolean datatype.
dvAspect (Data or View Aspect)	Specifies the desired Data or View Aspect of the object when drawing or getting data The possible values for this attribute are defined by the ST_DvAspect simple type (§3.18.25).
id (Relationship Id) Namespace: .../officeDocument/2006/relationships	Relationship Id of the relationship pointing to the object persistence part. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
link (Embedded Object's Link Moniker)	The embedded object's link moniker. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
oleUpdate (Linked Embedded Object Update)	Indicates whether the linked object updates the cached data automatically or only when the container requests an update, only present if the embedded object is linked. The possible values for this attribute are defined by the ST_OleUpdate simple type (§3.18.51).
progId (Embedded Object ProgId)	ProgId of the embedded object. The possible values for this attribute are defined by the XML Schema string datatype.
shapeId (Shape Id)	Id of the shape this object is associated with. Corresponds with the shape @id in the drawingML part. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OleObject">
    <attribute name="progId" type="xsd:string" use="optional"/>
    <attribute name="dvAspect" type="ST_DvAspect" use="optional" default="DVASPECT_CONTENT"/>
    <attribute name="link" type="ST_Xstring" use="optional"/>
    <attribute name="oleUpdate" type="ST_OleUpdate" use="optional"/>
    <attribute name="autoLoad" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="shapeId" type="xsd:unsignedInt" use="required"/>
    <attribute ref="r:id" use="optional"/>
</complexType>
```

3.3.1.58 oleObjects (Embedded Objects)

Embedded objects collection in this worksheet.

[*Example*:

This example shows two embedded objects.

```
<oleObjects>
  <oleObject progId="Word.Document.12" shapeId="1025" r:id="rId4"/>
  <oleObject progId="PowerPoint.Show.12" shapeId="1026" r:id="rId5"/>
</oleObjects>
```

end example]

Parent Elements
dialogsheet (§3.3.1.32); worksheet (§3.3.1.96)

Child Elements	Subclause
oleObject (Embedded Object)	§3.3.1.57

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OleObjects">
    <sequence>
        <element name="oleObject" type="CT_OleObject" minOccurs="1" maxOccurs="unbounded"/>
    </sequence>
</complexType>
```

3.3.1.59 outlinePr (Outline Properties)

Outline properties of the worksheet.

[*Example*: This example indicates that when an outline is applied to data, formatting shall be applied to the outline result.

```
<sheetPr>
  <outlinePr applyStyles="1"/>
</sheetPr>
```

end example]

Parent Elements
sheetPr (§3.3.1.79)

Attributes	Description
applyStyles (Apply Styles in Outline)	Flag indicating whether to apply styles in an outline, when outline is applied. Outline styles are described in Styles (§3.8). The possible values for this attribute are defined by the XML Schema boolean datatype.
showOutlineSymbols (Show Outline Symbols)	Flag indicating whether the sheet has outline symbols visible. This flag shall always be overridden by the showOutlineSymbols attribute on sheetView when there is a conflict. The possible values for this attribute are defined by the XML Schema boolean datatype.
summaryBelow (Summary Below)	Flag indicating whether summary rows appear below detail in an outline, when applying an outline. When `true` a summary row is inserted below the detailed data being summarized and a new outline level is established on that row. When `false` a summary row is inserted above the detailed data being summarized and a new outline level is established on that row. Note that toggling this flag on existing outlines requires an update to cell table, specifically, putting the summary functions in the proper rows, and flagging these rows as new outline levels, and possibly resetting their collapsed state. The possible values for this attribute are defined by the XML Schema boolean datatype.
summaryRight (Summary Right)	Flag indicating whether summary columns appear to the right of detail in an outline, when applying an outline. When `true` a summary column is inserted to the right of the detailed data being summarized and a new outline level is established on that column. When `false` a summary column is inserted to the left of the detailed data being summarized and a new outline level is established on that column. Note that toggling this flag on existing outlines requires an update to cell table, specifically, putting the summary functions in the proper columns, and flagging these columns as new outline levels, and possibly resetting their collapsed state.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OutlinePr">
   <attribute name="applyStyles" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="summaryBelow" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="summaryRight" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="showOutlineSymbols" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

3.3.1.60 pageMargins (Page Margins)

Page margins for a sheet or a custom sheet view.

[*Example*:

```
<pageMargins left="0.7" right="0.7" top="0.75" bottom="0.75" header="0.3"
  footer="0.3"/>
```

end example]

Parent Elements
chartsheet (§3.3.1.11); customSheetView (§3.3.1.23); customSheetView (§3.3.1.22); dialogsheet (§3.3.1.32); worksheet (§3.3.1.96)

Attributes	Description
bottom (Bottom Page Margin)	Bottom Page Margin in inches. The possible values for this attribute are defined by the XML Schema double datatype.
footer (Footer Page Margin)	Footer Page Margin in inches. The possible values for this attribute are defined by the XML Schema double datatype.
header (Header Page Margin)	Header Page Margin in inches. The possible values for this attribute are defined by the XML Schema double datatype.
left (Left Page Margin)	Left Page Margin in inches. The possible values for this attribute are defined by the XML Schema double datatype.
right (Right Page Margin)	Right page margin in inches. The possible values for this attribute are defined by the XML Schema double datatype.
top (Top Page	Top Page Margin in inches.

Attributes	Description
Margin)	The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PageMargins">
   <attribute name="left" type="xsd:double" use="required"/>
   <attribute name="right" type="xsd:double" use="required"/>
   <attribute name="top" type="xsd:double" use="required"/>
   <attribute name="bottom" type="xsd:double" use="required"/>
   <attribute name="header" type="xsd:double" use="required"/>
   <attribute name="footer" type="xsd:double" use="required"/>
</complexType>
```

3.3.1.61 pageSetup (Page Setup Settings)

Page setup settings for the worksheet.

Parent Elements
customSheetView (§3.3.1.23); dialogsheet (§3.3.1.32); worksheet (§3.3.1.96)

Attributes	Description
blackAndWhite (Black And White)	Print black and white. The possible values for this attribute are defined by the XML Schema boolean datatype.
cellComments (Print Cell Comments)	This attribute specifies how to print cell comments. The possible values for this attribute are defined by the ST_CellComments simple type (§3.18.6).
copies (Number Of Copies)	Number of copies to print. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
draft (Draft)	Print without graphics. The possible values for this attribute are defined by the XML Schema boolean datatype.
errors (Print Error Handling)	Specifies how to print cell values for cells with errors. The possible values for this attribute are defined by the ST_PrintError simple type (§3.18.62).
firstPageNumber (First Page Number)	Page number for first printed page. If no value is specified, then 'automatic' is assumed. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

Attributes	Description
fitToHeight (Fit To Height)	Number of vertical pages to fit on. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
fitToWidth (Fit To Width)	Number of horizontal pages to fit on. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
horizontalDpi (Horizontal DPI)	Horizontal print resolution of the device. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
id (Id) Namespace: .../officeDocument/2006/relationships	Relationship Id of the devMode printer settings part. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
orientation (Orientation)	Orientation of the page. The possible values for this attribute are defined by the ST_Orientation simple type (§3.18.52).
pageOrder (Page Order)	Order of printed pages. The possible values for this attribute are defined by the ST_PageOrder simple type (§3.18.53).
paperSize (Paper Size)	Paper size 1 = Letter paper (8.5 in. by 11 in.) 2 = Letter small paper (8.5 in. by 11 in.) 3 = Tabloid paper (11 in. by 17 in.) 4 = Ledger paper (17 in. by 11 in.) 5 = Legal paper (8.5 in. by 14 in.) 6 = Statement paper (5.5 in. by 8.5 in.) 7 = Executive paper (7.25 in. by 10.5 in.) 8 = A3 paper (297 mm by 420 mm) 9 = A4 paper (210 mm by 297 mm) 10 = A4 small paper (210 mm by 297 mm) 11 = A5 paper (148 mm by 210 mm) 12 = B4 paper (250 mm by 353 mm) 13 = B5 paper (176 mm by 250 mm) 14 = Folio paper (8.5 in. by 13 in.) 15 = Quarto paper (215 mm by 275 mm) 16 = Standard paper (10 in. by 14 in.) 17 = Standard paper (11 in. by 17 in.)

Attributes	Description
	18 = Note paper (8.5 in. by 11 in.) 19 = #9 envelope (3.875 in. by 8.875 in.) 20 = #10 envelope (4.125 in. by 9.5 in.) 21 = #11 envelope (4.5 in. by 10.375 in.) 22 = #12 envelope (4.75 in. by 11 in.) 23 = #14 envelope (5 in. by 11.5 in.) 24 = C paper (17 in. by 22 in.) 25 = D paper (22 in. by 34 in.) 26 = E paper (34 in. by 44 in.) 27 = DL envelope (110 mm by 220 mm) 28 = C5 envelope (162 mm by 229 mm) 29 = C3 envelope (324 mm by 458 mm) 30 = C4 envelope (229 mm by 324 mm) 31 = C6 envelope (114 mm by 162 mm) 32 = C65 envelope (114 mm by 229 mm) 33 = B4 envelope (250 mm by 353 mm) 34 = B5 envelope (176 mm by 250 mm) 35 = B6 envelope (176 mm by 125 mm) 36 = Italy envelope (110 mm by 230 mm) 37 = Monarch envelope (3.875 in. by 7.5 in.). 38 = 6 3/4 envelope (3.625 in. by 6.5 in.) 39 = US standard fanfold (14.875 in. by 11 in.) 40 = German standard fanfold (8.5 in. by 12 in.) 41 = German legal fanfold (8.5 in. by 13 in.) 42 = ISO B4 (250 mm by 353 mm) 43 = Japanese double postcard (200 mm by 148 mm) 44 = Standard paper (9 in. by 11 in.) 45 = Standard paper (10 in. by 11 in.) 46 = Standard paper (15 in. by 11 in.) 47 = Invite envelope (220 mm by 220 mm) 50 = Letter extra paper (9.275 in. by 12 in.) 51 = Legal extra paper (9.275 in. by 15 in.) 52 = Tabloid extra paper (11.69 in. by 18 in.) 53 = A4 extra paper (236 mm by 322 mm) 54 = Letter transverse paper (8.275 in. by 11 in.) 55 = A4 transverse paper (210 mm by 297 mm) 56 = Letter extra transverse paper (9.275 in. by 12 in.) 57 = SuperA/SuperA/A4 paper (227 mm by 356 mm) 58 = SuperB/SuperB/A3 paper (305 mm by 487 mm) 59 = Letter plus paper (8.5 in. by 12.69 in.) 60 = A4 plus paper (210 mm by 330 mm) 61 = A5 transverse paper (148 mm by 210 mm) 62 = JIS B5 transverse paper (182 mm by 257 mm) 63 = A3 extra paper (322 mm by 445 mm) 64 = A5 extra paper (174 mm by 235 mm) 65 = ISO B5 extra paper (201 mm by 276 mm)

Attributes	Description
	66 = A2 paper (420 mm by 594 mm) 67 = A3 transverse paper (297 mm by 420 mm) 68 = A3 extra transverse paper (322 mm by 445 mm) The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
scale (Print Scale)	Print scaling. Valid values range from 10 to 400. For example: 10 - 10% 20 - 20% … 100 - 100% … 400 - 400% This setting is overridden when fitToWidth and/or fitToHeight are in use. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
useFirstPageNumber (Use First Page Number)	Use firstPageNumber value for first page number, and do not auto number the pages. The possible values for this attribute are defined by the XML Schema boolean datatype.
usePrinterDefaults (Use Printer Defaults)	Use the printer's defaults settings for page setup values and don't use the default values specified in the schema. For example, if dpi is not present or specified in the XML, the application shall not assume 600dpi as specified in the schema as a default and instead shall let the printer specify the default dpi. The possible values for this attribute are defined by the XML Schema boolean datatype.
verticalDpi (Vertical DPI)	Vertical print resolution of the device. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PageSetup">
    <attribute name="paperSize" type="xsd:unsignedInt" use="optional" default="1"/>
    <attribute name="scale" type="xsd:unsignedInt" use="optional" default="100"/>
    <attribute name="firstPageNumber" type="xsd:unsignedInt" use="optional" default="1"/>
    <attribute name="fitToWidth" type="xsd:unsignedInt" use="optional" default="1"/>
    <attribute name="fitToHeight" type="xsd:unsignedInt" use="optional" default="1"/>
    <attribute name="pageOrder" type="ST_PageOrder" use="optional" default="downThenOver"/>
    <attribute name="orientation" type="ST_Orientation" use="optional" default="default"/>
    <attribute name="usePrinterDefaults" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="blackAndWhite" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="draft" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="cellComments" type="ST_CellComments" use="optional" default="none"/>
    <attribute name="useFirstPageNumber" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="errors" type="ST_PrintError" use="optional" default="displayed"/>
    <attribute name="horizontalDpi" type="xsd:unsignedInt" use="optional" default="600"/>
    <attribute name="verticalDpi" type="xsd:unsignedInt" use="optional" default="600"/>
    <attribute name="copies" type="xsd:unsignedInt" use="optional" default="1"/>
    <attribute ref="r:id" use="optional"/>
</complexType>
```

3.3.1.62 pageSetup (Chart Sheet Page Setup)

This element provides page setup properties for chart sheets.

Parent Elements
chartsheet (§3.3.1.11); customSheetView (§3.3.1.22)

Attributes	Description
blackAndWhite (Black And White)	Print black and white. The possible values for this attribute are defined by the XML Schema boolean datatype.
copies (Number Of Copies)	Number of copies to print. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
draft (Draft)	Print draft quality. The possible values for this attribute are defined by the XML Schema boolean datatype.
firstPageNumber (First Page Number)	Page number for first printed page. If no value is specified, then 'automatic' is assumed. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
horizontalDpi (Horizontal DPI)	Horizontal print resolution of the device. The possible values for this attribute are defined by the XML Schema unsignedInt

Attributes	Description
	datatype.
id (Id) Namespace: .../officeDocument/2006/relationships	Relationship Id of the devMode printer settings part. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
orientation (Orientation)	Orientation of the page. The possible values for this attribute are defined by the ST_Orientation simple type (§3.18.52).
paperSize (Paper Size)	1 = Letter paper (8.5 in. by 11 in.) 2 = Letter small paper (8.5 in. by 11 in.) 3 = Tabloid paper (11 in. by 17 in.) 4 = Ledger paper (17 in. by 11 in.) 5 = Legal paper (8.5 in. by 14 in.) 6 = Statement paper (5.5 in. by 8.5 in.) 7 = Executive paper (7.25 in. by 10.5 in.) 8 = A3 paper (297 mm by 420 mm) 9 = A4 paper (210 mm by 297 mm) 10 = A4 small paper (210 mm by 297 mm) 11 = A5 paper (148 mm by 210 mm) 12 = B4 paper (250 mm by 353 mm) 13 = B5 paper (176 mm by 250 mm) 14 = Folio paper (8.5 in. by 13 in.) 15 = Quarto paper (215 mm by 275 mm) 16 = Standard paper (10 in. by 14 in.) 17 = Standard paper (11 in. by 17 in.) 18 = Note paper (8.5 in. by 11 in.) 19 = #9 envelope (3.875 in. by 8.875 in.) 20 = #10 envelope (4.125 in. by 9.5 in.) 21 = #11 envelope (4.5 in. by 10.375 in.) 22 = #12 envelope (4.75 in. by 11 in.) 23 = #14 envelope (5 in. by 11.5 in.) 24 = C paper (17 in. by 22 in.) 25 = D paper (22 in. by 34 in.) 26 = E paper (34 in. by 44 in.) 27 = DL envelope (110 mm by 220 mm) 28 = C5 envelope (162 mm by 229 mm) 29 = C3 envelope (324 mm by 458 mm) 30 = C4 envelope (229 mm by 324 mm) 31 = C6 envelope (114 mm by 162 mm) 32 = C65 envelope (114 mm by 229 mm) 33 = B4 envelope (250 mm by 353 mm) 34 = B5 envelope (176 mm by 250 mm)

Attributes	Description
	35 = B6 envelope (176 mm by 125 mm) 36 = Italy envelope (110 mm by 230 mm) 37 = Monarch envelope (3.875 in. by 7.5 in.). 38 = 6 3/4 envelope (3.625 in. by 6.5 in.) 39 = US standard fanfold (14.875 in. by 11 in.) 40 = German standard fanfold (8.5 in. by 12 in.) 41 = German legal fanfold (8.5 in. by 13 in.) 42 = ISO B4 (250 mm by 353 mm) 43 = Japanese double postcard (200 mm by 148 mm) 44 = Standard paper (9 in. by 11 in.) 45 = Standard paper (10 in. by 11 in.) 46 = Standard paper (15 in. by 11 in.) 47 = Invite envelope (220 mm by 220 mm) 50 = Letter extra paper (9.275 in. by 12 in.) 51 = Legal extra paper (9.275 in. by 15 in.) 52 = Tabloid extra paper (11.69 in. by 18 in.) 53 = A4 extra paper (236 mm by 322 mm) 54 = Letter transverse paper (8.275 in. by 11 in.) 55 = A4 transverse paper (210 mm by 297 mm) 56 = Letter extra transverse paper (9.275 in. by 12 in.) 57 = SuperA/SuperA/A4 paper (227 mm by 356 mm) 58 = SuperB/SuperB/A3 paper (305 mm by 487 mm) 59 = Letter plus paper (8.5 in. by 12.69 in.) 60 = A4 plus paper (210 mm by 330 mm) 61 = A5 transverse paper (148 mm by 210 mm) 62 = JIS B5 transverse paper (182 mm by 257 mm) 63 = A3 extra paper (322 mm by 445 mm) 64 = A5 extra paper (174 mm by 235 mm) 65 = ISO B5 extra paper (201 mm by 276 mm) 66 = A2 paper (420 mm by 594 mm) 67 = A3 transverse paper (297 mm by 420 mm) 68 = A3 extra transverse paper (322 mm by 445 mm) The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
useFirstPageNumber (Use First Page Number)	Use firstPageNumber value for first page number, and do not auto number the pages. The possible values for this attribute are defined by the XML Schema boolean datatype.
usePrinterDefaults (Use Printer Defaults)	Use the printer's defaults settings for page setup values and don't use the default values specified in the schema. For example, if dpi is not present or specified in the XML, the application shall not assume 600dpi as specified in the schema as a default and instead shall let the printer specify the default dpi. The possible values for this attribute are defined by the XML Schema boolean datatype.
verticalDpi	Vertical print resolution of the device.

Attributes	Description
(Vertical DPI)	The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CsPageSetup">
   <attribute name="paperSize" type="xsd:unsignedInt" use="optional" default="1"/>
   <attribute name="firstPageNumber" type="xsd:unsignedInt" use="optional" default="1"/>
   <attribute name="orientation" type="ST_Orientation" use="optional" default="default"/>
   <attribute name="usePrinterDefaults" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="blackAndWhite" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="draft" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="useFirstPageNumber" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="horizontalDpi" type="xsd:unsignedInt" use="optional" default="600"/>
   <attribute name="verticalDpi" type="xsd:unsignedInt" use="optional" default="600"/>
   <attribute name="copies" type="xsd:unsignedInt" use="optional" default="1"/>
   <attribute ref="r:id" use="optional"/>
</complexType>
```

3.3.1.63 pageSetUpPr (Page Setup Properties)

Page setup properties of the worksheet

Parent Elements
sheetPr (§3.3.1.79)

Attributes	Description
autoPageBreaks (Show Auto Page Breaks)	Flag indicating whether the sheet displays Automatic Page Breaks. The possible values for this attribute are defined by the XML Schema boolean datatype.
fitToPage (Fit To Page)	Flag indicating whether the Fit to Page print option is enabled. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PageSetUpPr">
   <attribute name="autoPageBreaks" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="fitToPage" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.3.1.64 pane (View Pane)

Worksheet view pane

Parent Elements

Parent Elements
customSheetView (§3.3.1.23); sheetView (§3.3.1.83)

Attributes	Description
activePane (Active Pane)	The pane that is active. The possible values for this attribute are defined by the ST_Pane simple type (§3.18.54).
state (Split State)	Indicates whether the pane has horizontal / vertical splits, and whether those splits are frozen. The possible values for this attribute are defined by the ST_PaneState simple type (§3.18.55).
topLeftCell (Top Left Visible Cell)	Location of the top left visible cell in the bottom right pane (when in Left-To-Right mode). The possible values for this attribute are defined by the ST_CellRef simple type (§3.18.8).
xSplit (Horizontal Split Position)	Horizontal position of the split, in 1/20th of a point; 0 (zero) if none. If the pane is frozen, this value indicates the number of columns visible in the top pane. The possible values for this attribute are defined by the XML Schema double datatype.
ySplit (Vertical Split Position)	Vertical position of the split, in 1/20th of a point; 0 (zero) if none. If the pane is frozen, this value indicates the number of rows visible in the left pane. The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Pane">
   <attribute name="xSplit" type="xsd:double" use="optional" default="0"/>
   <attribute name="ySplit" type="xsd:double" use="optional" default="0"/>
   <attribute name="topLeftCell" type="ST_CellRef" use="optional"/>
   <attribute name="activePane" type="ST_Pane" use="optional" default="topLeft"/>
   <attribute name="state" type="ST_PaneState" use="optional" default="split"/>
</complexType>
```

3.3.1.65 picture (Background Image)

Background sheet image.

[*Example*:

```
<picture r:id="rId1"/>
```

end example]

Parent Elements
chartsheet (§3.3.1.11); worksheet (§3.3.1.96)

Attributes	Description
id (Relationship Id) Namespace: .../officeDocument/2006/relationships	Relationship Id pointing to the image part. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SheetBackgroundPicture">
   <attribute ref="r:id" use="required"/>
</complexType>
```

3.3.1.66 pivotArea (Pivot Area)

Rule describing a PivotTable selection.

Parent Elements
autoSortScope (§3.10.1.1); calculatedItem (§3.10.1.8); chartFormat (§3.10.1.12); format (§3.10.1.35); pivotAreas (§3.10.1.66); pivotSelection (§3.3.1.67)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
references (References)	§3.10.2.2

Attributes	Description
axis (Axis)	The region of the PivotTable to which this rule applies. The possible values for this attribute are defined by the ST_Axis simple type (§3.18.1).
cacheIndex (Cache Index)	Flag indicating whether any indexes refer to fields or items in the Pivot cache and not the view. The possible values for this attribute are defined by the XML Schema boolean datatype.
collapsedLevelsAreSubtotals (Collapsed Levels Are Subtotals)	Flag indicating if collapsed levels/dimensions are considered subtotals. The possible values for this attribute are defined by the XML Schema boolean datatype.
dataOnly (Data Only)	Flag indicating whether only the data values (in the data area of the view) for an item selection are selected and does not include the item labels. The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description
field (Field Index)	Index of the field that this selection rule refers to. The possible values for this attribute are defined by the XML Schema int datatype.
fieldPosition (Field Position)	Position of the field within the axis to which this rule applies. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
grandCol (Include Column Grand Total)	Flag indicating whether the column grand total is included. The possible values for this attribute are defined by the XML Schema boolean datatype.
grandRow (Include Row Grand Total)	Flag indicating whether the row grand total is included. The possible values for this attribute are defined by the XML Schema boolean datatype.
labelOnly (Labels Only)	Flag indicating whether only the item labels for an item selection are selected and does not include the data values (in the data area of the view). The possible values for this attribute are defined by the XML Schema boolean datatype.
offset (Offset Reference)	A Reference that specifies a subset of the selection area. Points are relative to the top left of the selection area. The possible values for this attribute are defined by the ST_Ref simple type (§3.18.64).
outline (Outline)	Flag indicating whether the rule refers to an area that is in outline mode. The possible values for this attribute are defined by the XML Schema boolean datatype.
type (Rule Type)	Indicates the type of selection rule. The possible values for this attribute are defined by the ST_PivotAreaType simple type (§3.18.60).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PivotArea">
   <sequence>
      <element name="references" minOccurs="0" type="CT_PivotAreaReferences"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="field" use="optional" type="xsd:int"/>
   <attribute name="type" type="ST_PivotAreaType" default="normal"/>
   <attribute name="dataOnly" type="xsd:boolean" default="true"/>
   <attribute name="labelOnly" type="xsd:boolean" default="false"/>
   <attribute name="grandRow" type="xsd:boolean" default="false"/>
   <attribute name="grandCol" type="xsd:boolean" default="false"/>
   <attribute name="cacheIndex" type="xsd:boolean" default="false"/>
   <attribute name="outline" type="xsd:boolean" default="true"/>
   <attribute name="offset" type="ST_Ref"/>
   <attribute name="collapsedLevelsAreSubtotals" type="xsd:boolean" default="false"/>
   <attribute name="axis" type="ST_Axis" use="optional"/>
   <attribute name="fieldPosition" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.3.1.67 pivotSelection (PivotTable Selection)

A collection of PivotTable structure selections. A PivotTable structure selection is a way of specifying what cells in the PivotTable are selected. Instead of specifying cell addresses in a sqref, a particular area or structure within the PivotTable is specified. In this way there is semantic meaning regarding what is selected, rather than simply a list of cell or ranges contained in the selection. Typically fields on the row or column axis are selected.

[*Example*: For example, the innermost field (Product SubCategory) is selected in this PivotTable:

	A	B	C
1			
2		State	(All)
3		City	(All)
4			
5			Column Labels
6			⊟2001
7			⊟3
8			July
9		Row Labels	Sum of Sales Amount
10		⊟Bikes	209652.9046
11		⊟Mountain Bikes	64424.81
12		Mountain-100 Black, 38	3374.99
13		Mountain-100 Black, 42	3374.99
14		Mountain-100 Black, 44	13499.96
15		Mountain-100 Black, 48	3374.99
16		Mountain-100 Silver, 38	6799.98
17		Mountain-100 Silver, 42	6799.98
18		Mountain-100 Silver, 44	16999.95
19		Mountain-100 Silver, 48	10199.97
20		⊟Road Bikes	145228.0946
21		Road-150 Red, 44	25047.89
22		Road-150 Red, 48	42939.24
23		Road-150 Red, 52	21469.62
24		Road-150 Red, 56	25047.89
25		Road-150 Red, 62	28626.16
26		Road-650 Black, 44	699.0982
27		Road-650 Black, 52	
28		Road-650 Black, 62	699.0982
29		Road-650 Red, 44	699.0982
30		Road-650 Red, 48	
31		Road-650 Red, 52	
32		Road-650 Red, 58	
33		Road-650 Red, 60	
34		Grand Total	209652.9046

The corresponding pivotSelection XML should look like this:

```
<pivotSelection pane="bottomRight" showHeader="1" axis="axisRow" dimension="2"
  activeRow="11" activeCol="1" previousRow="11" previousCol="1" click="1"
  r:id="rId1">
```

```
    <pivotArea dataOnly="0" labelOnly="1" fieldPosition="0">
      <references count="1">
        <reference field="9" count="0"/>
      </references>
    </pivotArea>
  </pivotSelection>
```

axis indicates that this selection is on the row axis, dimension indicates the field level within the row axis that is selected (zero-based index), activeCol and activeRow respectively indicate where in the grid the selection is located, and reference field indicates to which particular field the selection corresponds.

end example]

Parent Elements
sheetView (§3.3.1.83)

Child Elements	Subclause
pivotArea (Pivot Area)	§3.3.1.66

Attributes	Description
activeCol (Active Column)	The column (zero-based) of active cell for structure selection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
activeRow (Active Row)	The row (zero-based) of active cell for structure selection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
axis (Axis)	Axis of the PivotTable on which this selection lies. The possible values for this attribute are defined by the ST_Axis simple type (§3.18.1).
click (Click Count)	Number of clicks for this structure selection. For some selection combinations, subsequent clicks on the same target area cycles the actual selection through some variances. Therefore number of clicks on the selection must be recorded, if it is desirable to restore this state of the selection cycle on load. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
count (Selection Count)	Number of selections for the structure selection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

Attributes	Description
data (Data Selection)	Flag indicating whether the structure selection is for data only. The possible values for this attribute are defined by the XML Schema boolean datatype.
dimension (Dimension)	Indicates the field level within the axis that is selected (zero-based index). The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
extendable (Extendable)	Flag indicating whether the structure selection can have additional selections added to it. The possible values for this attribute are defined by the XML Schema boolean datatype.
id (Relationship Id) Namespace: .../officeDocument/2006/relationships	Relationship Id pointing to the particular PivotTable Part corresponding to this selection. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
label (Label)	Flag indicating whether the structure selection is for labels only (e.g., a grand total row is selected). The possible values for this attribute are defined by the XML Schema boolean datatype.
max (Maximum)	The maximum line the structure selection contains. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
min (Minimum)	The minimum line the structure selection contains. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
pane (Pane)	The pane to which this PivotTable structure selection belongs. The possible values for this attribute are defined by the ST_Pane simple type (§3.18.54).
previousCol (Previous Column Selection)	1-based index to the column immediately left of the structure selection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
previousRow (Previous Row)	1-based index to the row immediately above the structure selection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
showHeader (Show Header)	Flag indicating whether selection toggle from data only to header only to both is enabled. `False` means disabled. The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description
start (Start)	The line the structure selection begins (zero-based). This is the line clicked to initiate the structure selection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PivotSelection">
   <sequence>
      <element name="pivotArea" type="CT_PivotArea"/>
   </sequence>
   <attribute name="pane" type="ST_Pane" use="optional" default="topLeft"/>
   <attribute name="showHeader" type="xsd:boolean" default="false"/>
   <attribute name="label" type="xsd:boolean" default="false"/>
   <attribute name="data" type="xsd:boolean" default="false"/>
   <attribute name="extendable" type="xsd:boolean" default="false"/>
   <attribute name="count" type="xsd:unsignedInt" default="0"/>
   <attribute name="axis" type="ST_Axis" use="optional"/>
   <attribute name="dimension" type="xsd:unsignedInt" default="0"/>
   <attribute name="start" type="xsd:unsignedInt" default="0"/>
   <attribute name="min" type="xsd:unsignedInt" default="0"/>
   <attribute name="max" type="xsd:unsignedInt" default="0"/>
   <attribute name="activeRow" type="xsd:unsignedInt" default="0"/>
   <attribute name="activeCol" type="xsd:unsignedInt" default="0"/>
   <attribute name="previousRow" type="xsd:unsignedInt" default="0"/>
   <attribute name="previousCol" type="xsd:unsignedInt" default="0"/>
   <attribute name="click" type="xsd:unsignedInt" default="0"/>
   <attribute ref="r:id" use="optional"/>
</complexType>
```

3.3.1.68 printOptions (Print Options)

Print options for the sheet. Printer-specific settings are stored separately in the Printer Settings part as defined in Part 1.

Parent Elements
customSheetView (§3.3.1.23); dialogsheet (§3.3.1.32); worksheet (§3.3.1.96)

Attributes	Description
gridLines (Print Grid Lines)	Used in conjunction with gridLinesSet. If both gridLines and gridlinesSet are `true`, then grid lines shall print. Otherwise, they shall not (i.e., one or both have `false` values). The possible values for this attribute are defined by the XML Schema boolean datatype.
gridLinesSet (Grid Lines Set)	Used in conjunction with gridLines. If both gridLines and gridLinesSet are `true`, then grid lines shall print. Otherwise, they shall not (i.e., one or both have `false` values).

Attributes	Description
	The possible values for this attribute are defined by the XML Schema boolean datatype.
headings (Print Headings)	Print row and column headings. The possible values for this attribute are defined by the XML Schema boolean datatype.
horizontalCentered (Horizontal Centered)	Center on page horizontally when printing. The possible values for this attribute are defined by the XML Schema boolean datatype.
verticalCentered (Vertical Centered)	Center on page vertically when printing. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PrintOptions">
    <attribute name="horizontalCentered" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="verticalCentered" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="headings" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="gridLines" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="gridLinesSet" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

3.3.1.69 protectedRange (Protected Range)

A specified range to be protected. Ranges listed here are protected only when the sheet protection is ON and the cell is flagged as being locked. If no password is specified here, then read/write permissions are automatically given to all users, regardless of additional security descriptor information. In other words, the security descriptor information (specific types of access) at the user level is only applied if a password for this range is specified.

When a password is specified, then users not listed specifically as having access should be prompted with a password. If that user supplies the correct password, then they may edit the range or cell in question. This protection is optional and may be ignored by applications who choose not to support this functionality.

Parent Elements
protectedRanges (§3.3.1.70)

Attributes	Description
name (Name)	Range title. This is used as a descriptor, not as a named range definition. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
password (Password)	Specifies the hash of the password required for editing this range. The hash is generated from an 8-bit wide character. 16-bit Unicode characters must be converted down to 8 bits before the hash is computed, using the logic defined in the revisionsPassword

Attributes	Description
	attribute of §3.2.29. The resulting value is hashed using the algorithm defined below. [*Note*: An example algorithm to hash the user input into the value stored is as follows: `// Function Input:` `//  szPassword: NULL terminated C-Style string` `//  cchPassword: The number of characters in szPassword (not including the NULL terminator)` `WORD GetPasswordHash(const CHAR *szPassword, int cchPassword) {` `  WORD wPasswordHash;` `  const CHAR *pch;` `  wPasswordHash = 0;` `  if (cchPassword > 0)` `  {` `  pch = &szPassword[cchPassword];` `  while (pch-- != szPassword)` `  {` ` wPasswordHash = ((wPasswordHash >> 14) & 0x01)	((wPasswordHash << 1) & 0x7fff);` ` wPasswordHash ^= *pch;` ` }` ` wPasswordHash ^= (0x8000	('N' << 8)	'K');` ` }`  ` return(wPasswordHash);` `}` *end note*] The possible values for this attribute are defined by the ST_UnsignedShortHex simple type (§3.18.87).
securityDescriptor (Security Descriptor)	Optional setting to specify the relative security descriptor. The security descriptor defines user accounts who may edit this range without providing a password to access the range. Removing this attribute shall remove all permissions granted or denied to users for this range. The possible values for this attribute are defined by the XML Schema string datatype.
sqref (Sequence of References)	The range to be protected. The possible values for this attribute are defined by the ST_Sqref simple type (§3.18.78).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ProtectedRange">
   <attribute name="password" type="ST_UnsignedShortHex" use="optional"/>
   <attribute name="sqref" type="ST_Sqref" use="required"/>
   <attribute name="name" type="ST_Xstring" use="required"/>
   <attribute name="securityDescriptor" type="xsd:string" use="optional"/>
</complexType>
```

3.3.1.70 protectedRanges (Protected Ranges)

This collection specifies all protected ranges on this worksheet.

[*Example*:

This example demonstrates that A1:C5 have been protected, with no password specified.

```
<protectedRanges>
  <protectedRange sqref="A1:C5" name="Range1"/>
</protectedRanges>
```

end example]

Parent Elements
worksheet (§3.3.1.96)

Child Elements	Subclause
protectedRange (Protected Range)	§3.3.1.69

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ProtectedRanges">
   <sequence>
      <element name="protectedRange" type="CT_ProtectedRange" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.3.1.71 row (Row)

The element expresses information about an entire row of a worksheet, and contains all cell definitions for a particular row in the worksheet.

[*Example*:

This row expresses information about row 2 in the worksheet, and contains 3 cell definitions.

```
<row r="2" spans="2:12">
  <c r="C2" s="1">
    <f>PMT(B3/12,B4,-B5)</f>
    <v>672.68336574300008</v>
  </c>
  <c r="D2">
    <v>180</v>
  </c>
  <c r="E2">
    <v>360</v>
  </c>
</row>
```

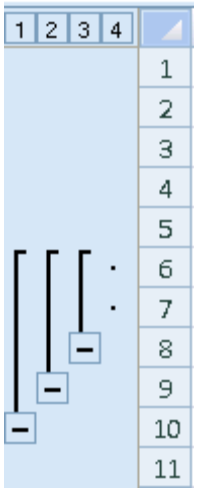
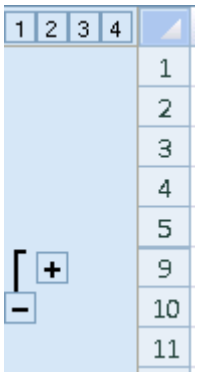
end example]

Parent Elements
sheetData (§3.3.1.77)

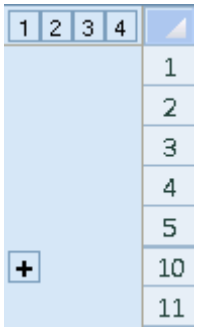
Child Elements	Subclause
c (Cell)	§3.3.1.3
extLst (Future Feature Data Storage Area)	§3.2.10

Attributes	Description
collapsed (Collapsed)	'1' if the rows 1 level of outlining deeper than the current row are in the collapsed outline state. It means that the rows which are 1 outline level deeper (numerically higher value) than the current row are currently hidden due to a collapsed outline state. It is possible for collapsed to be `false` and yet still have the rows in question hidden. This can be achieved by having a lower outline level collapsed, thus hiding all the child rows. [*Example*: This example shows 3 levels of outlining:

Attributes	Description

	
	
	In the XML shall be:
	`<sheetData>`
	`<row r="6" outlineLevel="3"/>`
	`<row r="7" outlineLevel="3"/>`
	`<row r="8" outlineLevel="2"/>`
	`<row r="9" outlineLevel="1"/>`
	`</sheetData>`
	
	end example]
	
	[*Example*:
	This example shows the same outline feature, with the middle level collapsed:
	
	 
	
	In the XML shall be:
	`<sheetData>`
	`<row r="6" hidden="1" outlineLevel="3"/>`
	`<row r="7" hidden="1" outlineLevel="3"/>`
	`<row r="8" hidden="1" outlineLevel="2"/>`
	`<row r="9" outlineLevel="1" collapsed="1"/>`
	`</sheetData>`

Attributes	Description

	end example]
	
	[*Example*:
	This example shows the same outline feature as above, where both the middle and lowest level are collapsed:
	
	 
	
	In the XML shall be:
	
	`<sheetData>`
	`  <row r="6" hidden="1" outlineLevel="3"/>`
	`  <row r="7" hidden="1" outlineLevel="3"/>`
	`  <row r="8" hidden="1" outlineLevel="2"/>`
	`  <row r="9" hidden="1" outlineLevel="1" collapsed="1"/>`
	`  <row r="10" collapsed="1"/>`
	`</sheetData>`
	
	Note that in this case, if the lowest level were expanded, the middle level would remain collapsed due to collapsed being `true` on row 9.
	
	end example]
	
	See description of outlinePr element's summaryBelow and summaryRight attributes for detailed information.
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

customFormat (Custom Format)	'1' if the row style should be applied. The possible values for this attribute are defined by the XML Schema boolean datatype.
customHeight (Custom Height)	'1' if the row height has been manually set. The possible values for this attribute are defined by the XML Schema boolean datatype.
hidden (Hidden)	'1' if the row is hidden, e.g., due to a collapsed outline or by manually selecting and hiding a row. The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description
ht (Row Height)	Row height measured in point size. There is no margin padding on row height. The possible values for this attribute are defined by the XML Schema double datatype.
outlineLevel (Outline Level)	Outlining level of the row, when outlining is on. See description of outlinePr element's summaryBelow and summaryRight attributes for detailed information. The possible values for this attribute are defined by the XML Schema unsignedByte datatype.
ph (Show Phonetic)	'1' if the row should show phonetic. The possible values for this attribute are defined by the XML Schema boolean datatype.
r (Row Index)	Row index. Indicates to which row in the sheet this <row> definition corresponds. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
s (Style Index)	Index to style record for the row (only applied if customFormat attribute is '1') The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

spans (Spans)	Optimization only, and not required. Specifies the range of non-empty columns (in the format X:Y) for the block of rows to which the current row belongs. To achieve the optimization, span attribute values in a single block should be the same.
	
	There are 16 rows per block, beginning with the first row.
	
	Note: this is an optimization, and is purely optional. Different span values within the same row block is allowed. Not writing the span value at all is also allowed.
	
	Blank rows are not required to write out span values.
	
	For example, if cells F8, E9, and D10 have data in them and the rest of the sheet is empty, then for those three rows (8,9, and 10), the spans value should each be "4:6":
	
	`<sheetData>`
	`<row r="8" spans="4:6">`
	`<c r="F8">`
	`<v>1</v>`
	`</c>`
	`</row>`
	`<row r="9" spans="4:6">`
	`<c r="E9">`
	`<v>2</v>`
	`</c>`
	`</row>`
	`<row r="10" spans="4:6">`

Attributes	Description
	<c r="D10"> <v>3</v> </c> </row> </sheetData> For example, if cells A1 and J10 have data in them and the rest of the sheet is empty, then the rows should be written like this: <sheetData> <row r="1" spans="1:10"> <c r="A1"> <v>1</v> </c> </row> <row r="10" spans="1:10"> <c r="J10"> <v>2</v> </c> </row> </sheetData> The possible values for this attribute are defined by the ST_CellSpans simple type (§3.18.10).
thickBot (Thick Bottom)	'1' if any cell in the row has a medium or thick bottom border, or if any cell in the row directly below the current row has a thick top border. When `true` and customHeight is `false`, this flag means that the row height has been adjusted higher by .75 points of the normal style font height. This also means that if the row no longer contains these borders, then the height is automatically re-adjusted down. This adjustment is in addition to any adjustment of height due to thickTop. Medium borders are these enumeration values from the Styles Part: • mediumDashDotDot • slantDashDot • mediumDashDot • mediumDashed • medium Thick borders are these enumeration values from the Styles Part: • thick • double The possible values for this attribute are defined by the XML Schema boolean datatype.
thickTop (Thick Top Border)	`True` if the row has a medium or thick top border, or if any cell in the row directly above the current row has a thick bottom border. When `true` and customHeight is `false`, this

Attributes	Description
	flag means that the row height has been adjusted higher by .75 points of the normal style font height. This also means that if the row no longer contains these borders, then the height is automatically re-adjusted down. This adjustment is in addition to any adjustment of height due to thickBot. Medium borders are these enumeration values from the Styles Part: • mediumDashDotDot • slantDashDot • mediumDashDot • mediumDashed • medium Thick borders are these enumeration values from the Styles Part: • thick • double The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Row">
   <sequence>
      <element name="c" type="CT_Cell" minOccurs="0" maxOccurs="unbounded"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="r" type="xsd:unsignedInt" use="optional"/>
   <attribute name="spans" type="ST_CellSpans" use="optional"/>
   <attribute name="s" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="customFormat" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="ht" type="xsd:double" use="optional"/>
   <attribute name="hidden" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="customHeight" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="outlineLevel" type="xsd:unsignedByte" use="optional" default="0"/>
   <attribute name="collapsed" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="thickTop" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="thickBot" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="ph" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.3.1.72 rowBreaks (Horizontal Page Breaks (Row))

Horizontal page break information used for print layout view, page layout view, drawing print breaks in normal view, and for printing the worksheet.

[*Example*: This example shows a break inserted at cell B25:

```
<rowBreaks count="1" manualBreakCount="1">
  <brk id="24" max="16383" man="1"/>
</rowBreaks>
```

end example]

Parent Elements
customSheetView (§3.3.1.23); worksheet (§3.3.1.96)

Child Elements	Subclause
brk (Break)	§3.3.1.2

Attributes	Description
count (Page Break Count)	Number of breaks in the collection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
manualBreakCount (Manual Break Count)	Number of manual breaks in the collection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PageBreak">
    <sequence>
        <element name="brk" type="CT_Break" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="count" type="xsd:unsignedInt" use="optional" default="0"/>
    <attribute name="manualBreakCount" type="xsd:unsignedInt" use="optional" default="0"/>
</complexType>
```

3.3.1.73 scenario (Scenario)

An individual scenario description. See parent element for example.

Parent Elements
scenarios (§3.3.1.74)

Child Elements	Subclause
inputCells (Input Cells)	§3.3.1.49

Attributes	Description
comment (Scenario Comment)	Comment for this scenario, rich text not supported. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
count (Changing Cell Count)	Number of input cells. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
hidden (Hidden Scenario)	Scenario is hidden when the sheet is protected and 'edit scenarios' is not enabled in sheet protection options. If the scenario is marked as hidden but sheet protection options specify to allow editing scenarios, then the scenario shall not be hidden. The possible values for this attribute are defined by the XML Schema boolean datatype.
locked (Scenario Locked)	Scenario is locked for editing when the sheet is protected. If sheet is protected and "edit scenarios" is enabled, then this setting is ignored. The possible values for this attribute are defined by the XML Schema boolean datatype.
name (Scenario Name)	Scenario's name (user input). Must be unique for the workbook. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
user (User Name)	Name of user who last changed the scenario. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Scenario">
   <sequence>
      <element name="inputCells" type="CT_InputCells" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="name" type="ST_Xstring" use="required"/>
   <attribute name="locked" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="hidden" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
   <attribute name="user" type="ST_Xstring" use="optional"/>
   <attribute name="comment" type="ST_Xstring" use="optional"/>
</complexType>
```

3.3.1.74 scenarios (Scenarios)

A collection of Scenarios. A scenario is a named what-if model that includes variable cells linked together by one or more formulas. For example, you might want to compare best-case and worst-case scenarios for sales in a coffee shop, based on the number of cups of coffee sold in a week.

[*Example*:

```
<scenarios current="1" show="0" sqref="G4 G6 G7 G8">
  <scenario name="Best Case" locked="1" count="3" user="anonymous"
    comment="Created on 6/9/2006_x000a_Modified on 6/9/2006">
    <inputCells r="D5" val="151" numFmtId="37"/>
    <inputCells r="D9" val="226"/>
    <inputCells r="D13" val="126"/>
  </scenario>
  <scenario name="Worst Case" locked="1" count="3" user="anonymous"
    comment="Created on 6/9/2006">
    <inputCells r="D5" val="50" numFmtId="37"/>
    <inputCells r="D9" val="40"/>
    <inputCells r="D13" val="30"/>
  </scenario>
</scenarios>
```

end example]

Parent Elements
worksheet (§3.3.1.96)

Child Elements	Subclause
scenario (Scenario)	§3.3.1.73

Attributes	Description
current (Current Scenario)	Zero-based index to current scenario selected. Can correspond to selection UI. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
show (Last Shown Scenario)	Zero-based index to last shown scenario. Indicates which scenario was last selected by the user to be run/shown. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
sqref (Sequence of References)	Range or sequence of cells used for scenario results summary. The possible values for this attribute are defined by the ST_Sqref simple type (§3.18.78).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Scenarios">
   <sequence>
      <element name="scenario" type="CT_Scenario" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="current" type="xsd:unsignedInt" use="optional"/>
   <attribute name="show" type="xsd:unsignedInt" use="optional"/>
   <attribute name="sqref" type="ST_Sqref" use="optional"/>
</complexType>
```

3.3.1.75 selection (Selection)

Worksheet view selection.

Parent Elements
customSheetView (§3.3.1.23); sheetView (§3.3.1.83)

Attributes	Description
activeCell (Active Cell Location)	Location of the active cell. The possible values for this attribute are defined by the ST_CellRef simple type (§3.18.8).
activeCellId (Active Cell Index)	0-based index of the range reference (in the array of references listed in sqref) containing the active cell. Only used when the selection in sqref is not contiguous. Therefore, this value needs to be aware of the order in which the range references are written in sqref. When this value is out of range then activeCell can be used. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
pane (Pane)	The pane to which this selection belongs. The possible values for this attribute are defined by the ST_Pane simple type (§3.18.54).
sqref (Sequence of References)	Range of the selection. Can be non-contiguous set of ranges. The possible values for this attribute are defined by the ST_Sqref simple type (§3.18.78).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Selection">
   <attribute name="pane" type="ST_Pane" use="optional" default="topLeft"/>
   <attribute name="activeCell" type="ST_CellRef" use="optional"/>
   <attribute name="activeCellId" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="sqref" type="ST_Sqref" use="optional" default="A1"/>
</complexType>
```

3.3.1.76 sheetCalcPr (Sheet Calculation Properties)

This element contains calculation properties for the worksheet.

Parent Elements
worksheet (§3.3.1.96)

Attributes	Description
fullCalcOnLoad (Full Calculation On Load)	Indicates whether the application should do a full calculate on load due to contents on this sheet. After load and successful calc, the application shall set this value to false. Set this to true when the application should calculate the workbook on load. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SheetCalcPr">
   <attribute name="fullCalcOnLoad" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.3.1.77 sheetData (Sheet Data)

This collection represents the cell table itself. This collection expresses information about each cell, grouped together by rows in the worksheet.

Parent Elements
worksheet (§3.3.1.96)

Child Elements	Subclause
row (Row)	§3.3.1.71

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SheetData">
   <sequence>
      <element name="row" type="CT_Row" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.3.1.78 sheetFormatPr (Sheet Format Properties)

Sheet formatting properties.

Parent Elements
dialogsheet (§3.3.1.32); worksheet (§3.3.1.96)

Attributes	Description
baseColWidth (Base Column Width)	Specifies the number of characters of the maximum digit width of the normal style's font. This value does not include margin padding or extra padding for gridlines. It is only the number of characters. See defaultColWidth description in this section for details on calculating this value. See the col element description, particularly the width attribute description, for more information on what is meant by "maximum digit width". The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
customHeight (Custom Height)	'True' if defaultRowHeight value has been manually set, or is different from the default value. The possible values for this attribute are defined by the XML Schema boolean datatype.
defaultColWidth (Default Column Width)	Default column width measured as the number of characters of the maximum digit width of the normal style's font. If the user has not set this manually, then it can be calculated: defaultColWidth = baseColumnWidth + {margin padding (2 pixels on each side, totalling 4 pixels)} + {gridline (1pixel)} If the user has set this manually, then there is no calculation, and simply a value is specified. The possible values for this attribute are defined by the XML Schema double datatype.
defaultRowHeight (Default Row Height)	Default row height measured in point size. Optimization so we don't have to write the height on all rows. This can be written out if most rows have custom height, to achieve the optimization. When the row height of all rows in a sheet is the default value, then that value is written here, and customHeight is not set. If a few rows have a different height, that information is written directly on each row. However, if most or all of the rows in the sheet have the same height, but that height isn't the default height, then that height value should be written here (as an optimization), and the customHeight flag should also be set. In this case, all rows having this height do not need to express the height, only rows whose height differs from this value need to be explicitly expressed. The possible values for this attribute are defined by the XML Schema double datatype.
outlineLevelCol (Column Outline Level)	Highest number of outline levels for columns in this sheet. These values must be in synch with the actual sheet outline levels. The possible values for this attribute are defined by the XML Schema unsignedByte datatype.

Attributes	Description
outlineLevelRow (Maximum Outline Row)	Highest number of outline level for rows in this sheet. These values must be in synch with the actual sheet outline levels. The possible values for this attribute are defined by the XML Schema unsignedByte datatype.
thickBottom (Thick Bottom Border)	'True' if rows have a thick bottom border by default. The possible values for this attribute are defined by the XML Schema boolean datatype.
thickTop (Thick Top Border)	'True' if rows have a thick top border by default. The possible values for this attribute are defined by the XML Schema boolean datatype.
zeroHeight (Hidden By Default)	'True' if rows are hidden by default. This setting is an optimization used when most rows of the sheet are hidden. In this case, instead of writing out every row and specifying hidden, it is much shorter to only write out the rows that are not hidden, and specify here that rows are hidden by default, and only not hidden if specified. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SheetFormatPr">
    <attribute name="baseColWidth" type="xsd:unsignedInt" use="optional" default="8"/>
    <attribute name="defaultColWidth" type="xsd:double" use="optional"/>
    <attribute name="defaultRowHeight" type="xsd:double" use="required"/>
    <attribute name="customHeight" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="zeroHeight" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="thickTop" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="thickBottom" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="outlineLevelRow" type="xsd:unsignedByte" use="optional" default="0"/>
    <attribute name="outlineLevelCol" type="xsd:unsignedByte" use="optional" default="0"/>
</complexType>
```

3.3.1.79 sheetPr (Sheet Properties)

Sheet-level properties.

Parent Elements
dialogsheet (§3.3.1.32); worksheet (§3.3.1.96)

Child Elements	Subclause
outlinePr (Outline Properties)	§3.3.1.59
pageSetUpPr (Page Setup Properties)	§3.3.1.63
tabColor (Sheet Tab Color)	§3.3.1.90

Attributes	Description
codeName (Code Name)	Specifies a stable name of the sheet, which should not change over time, and does not change from user input. This name should be used by code to reference a particular sheet. The possible values for this attribute are defined by the XML Schema string datatype.
enableFormatConditionsCalculation (Enable Conditional Formatting Calculations)	Flag indicating whether the conditional formatting calculations shall be evaluated. If set to `false`, then the min/max values of color scales or databars or threshold values in Top N rules shall not be updated. Essentially the conditional formatting "calc" is off. This is useful when conditional formats are being set programmatically at runtime, recalculation of the conditional formatting does not need to be done until the program execution has finished setting all the conditional formatting properties. The possible values for this attribute are defined by the XML Schema boolean datatype.
filterMode (Filter Mode)	Flag indicating whether the worksheet has one or more autofilters or advanced filters on. The possible values for this attribute are defined by the XML Schema boolean datatype.
published (Published)	Flag indicating whether the worksheet is published. The possible values for this attribute are defined by the XML Schema boolean datatype.
syncHorizontal (Synch Horizontal)	Flag indicating whether this worksheet is horizontally synced to the synchRef anchor point. When true and scroll location is missing from the window properties, the window view shall be scrolled to the horizontal (row) aspect of the synchRef value. The possible values for this attribute are defined by the XML Schema boolean datatype.
syncRef (Synch Reference)	Anchor point for worksheet's window. The possible values for this attribute are defined by the ST_Ref simple type (§3.18.64).
syncVertical (Synch Vertical)	Flag indicating whether this worksheet is vertically synced to the synchRef anchor point. When true and scroll location is missing from the window properties, the window view shall be scrolled to the vertical (column) aspect of the synchRef value. The possible values for this attribute are defined by the XML Schema boolean datatype.
transitionEntry (Transition Formula Entry)	Flag indicating whether the Transition Formula Entry (Lotus compatibility) option is enabled. The possible values for this attribute are defined by the XML Schema boolean datatype.
transitionEvaluation (Transition Formula Evaluation)	Flag indicating whether the Transition Formula Evaluation (Lotus compatibility) option is enabled. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SheetPr">
   <sequence>
      <element name="tabColor" type="CT_Color" minOccurs="0" maxOccurs="1"/>
      <element name="outlinePr" type="CT_OutlinePr" minOccurs="0" maxOccurs="1"/>
      <element name="pageSetUpPr" type="CT_PageSetUpPr" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="syncHorizontal" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="syncVertical" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="syncRef" type="ST_Ref" use="optional"/>
   <attribute name="transitionEvaluation" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="transitionEntry" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="published" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="codeName" type="xsd:string" use="optional"/>
   <attribute name="filterMode" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="enableFormatConditionsCalculation" type="xsd:boolean" use="optional"
     default="true"/>
</complexType>
```

3.3.1.80 sheetPr (Chart Sheet Properties)

This element specifies chart sheet properties.

Parent Elements
chartsheet (§3.3.1.11)

Child Elements	Subclause
tabColor (Sheet Tab Color)	§3.3.1.90

Attributes	Description
codeName (Code Name)	Specifies a stable name of the sheet, which should not change over time, and does not change from user input. This name should be used by code to reference a particular sheet. The possible values for this attribute are defined by the XML Schema string datatype.
published (Published)	Flag indicating whether the chart sheet is published. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ChartsheetPr">
   <sequence>
      <element name="tabColor" type="CT_Color" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="published" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="codeName" type="xsd:string" use="optional"/>
</complexType>
```

3.3.1.81 sheetProtection (Sheet Protection Options)

This collection expresses the sheet protection options to enforce when the sheet is protected.

[*Example*:

This example demonstrates that the sheet is protected, objects and scenarios may be edited, and

```
<sheetProtection sheet="1" objects="1" scenarios="1" formatCells="0"
  selectLockedCells="1"/>
```

end example]

Parent Elements
dialogsheet (§3.3.1.32); worksheet (§3.3.1.96)

Attributes	Description
autoFilter (AutoFilter Locked)	Autofilters are locked when the sheet is protected. The possible values for this attribute are defined by the XML Schema boolean datatype.
deleteColumns (Delete Columns Locked)	Deleting columns is locked when the sheet is protected. The possible values for this attribute are defined by the XML Schema boolean datatype.
deleteRows (Delete Rows Locked)	Deleting rows is locked when the sheet is protected. The possible values for this attribute are defined by the XML Schema boolean datatype.
formatCells (Format Cells Locked)	Formatting cells is locked when the sheet is protected. The possible values for this attribute are defined by the XML Schema boolean datatype.
formatColumns (Format Columns Locked)	Formatting columns is locked when the sheet is protected. The possible values for this attribute are defined by the XML Schema boolean datatype.
formatRows (Format Rows Locked)	Formatting rows is locked when the sheet is protected. The possible values for this attribute are defined by the XML Schema boolean datatype.
insertColumns	Inserting columns is locked when the sheet is protected.

Attributes	Description
(Insert Columns Locked)	The possible values for this attribute are defined by the XML Schema boolean datatype.
insertHyperlinks (Insert Hyperlinks Locked)	Inserting hyperlinks is locked when the sheet is protected. The possible values for this attribute are defined by the XML Schema boolean datatype.
insertRows (Insert Rows Locked)	Inserting rows is locked when the sheet is protected. The possible values for this attribute are defined by the XML Schema boolean datatype.
objects (Objects Locked)	Objects are locked when the sheet is protected. The possible values for this attribute are defined by the XML Schema boolean datatype.
password (Password)	Specifies the hash of the password required for editing this worksheet. This protection is optional and may be ignored by applications that choose not to support this functionality. The hash is generated from an 8-bit wide character. 16-bit Unicode characters must be converted down to 8 bits before the hash is computed, using the logic defined in the revisionsPassword attribute of §3.2.29. The resulting value is hashed using the algorithm defined below. [*Note*: An example algorithm to hash the user input into the value stored is as follows: `// Function Input:` `//  szPassword: NULL terminated C-Style string` `//  cchPassword: The number of characters in szPassword (not including the NULL terminator)` `WORD GetPasswordHash(const CHAR *szPassword, int cchPassword) {` `  WORD wPasswordHash;` `  const CHAR *pch;` `  wPasswordHash = 0;` `  if (cchPassword > 0)` `  {` `  pch = &szPassword[cchPassword];` `  while (pch-- != szPassword)` `  {` ` wPasswordHash = ((wPasswordHash >> 14) & 0x01)	((wPasswordHash << 1) & 0x7fff);` ` wPasswordHash ^= *pch;` ` }` ` wPasswordHash ^= (0x8000	('N' << 8)	'K');` ` }`  ` return(wPasswordHash);` `}` *end note*]

Attributes	Description
	The possible values for this attribute are defined by the ST_UnsignedShortHex simple type (§3.18.87).
pivotTables (Pivot Tables Locked)	Pivot tables are locked when the sheet is protected. The possible values for this attribute are defined by the XML Schema boolean datatype.
scenarios (Scenarios Locked)	Scenarios are locked when the sheet is protected. The possible values for this attribute are defined by the XML Schema boolean datatype.
selectLockedCells (Select Locked Cells Locked)	Selection of locked cells is locked when the sheet is protected. The possible values for this attribute are defined by the XML Schema boolean datatype.
selectUnlockedCells (Select Unlocked Cells Locked)	Selection of unlocked cells is locked when the sheet is protected. The possible values for this attribute are defined by the XML Schema boolean datatype.
sheet (Sheet Locked)	Sheet is locked when the sheet is protected. The possible values for this attribute are defined by the XML Schema boolean datatype.
sort (Sort Locked)	Sorting is locked when the sheet is protected. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SheetProtection">
    <attribute name="password" type="ST_UnsignedShortHex" use="optional"/>
    <attribute name="sheet" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="objects" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="scenarios" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="formatCells" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="formatColumns" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="formatRows" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="insertColumns" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="insertRows" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="insertHyperlinks" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="deleteColumns" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="deleteRows" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="selectLockedCells" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="sort" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="autoFilter" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="pivotTables" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="selectUnlockedCells" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.3.1.82 sheetProtection (Chart Sheet Protection)

This collection expresses the chart sheet protection options to enforce when the chart sheet is protected.

Parent Elements
chartsheet (§3.3.1.11)

Attributes	Description
content (Contents)	When true prevents users from making changes to items that are part of the chart, such as data series, axes, and legends. The chart continues to reflect changes made to its source data. The possible values for this attribute are defined by the XML Schema boolean datatype.
objects (Objects Locked)	When true prevents users from making changes to graphic objects— including shapes, text boxes, and controls— unless you unlock the specific objects before you protect the chart sheet. The possible values for this attribute are defined by the XML Schema boolean datatype.
password (Password)	Specifies the hash of the password required for editing this chart sheet. This protection is optional and may be ignored by applications that choose not to support this functionality. The hash is generated from an 8-bit wide character. 16-bit Unicode characters must be converted down to 8 bits before the hash is computed, using the logic defined in the revisionPassword attribute of §3.2.29. The resulting value is hashed using the algorithm defined below. [*Note*: An example algorithm to hash the user input into the value stored is as follows: `// Function Input:` `//  szPassword: NULL terminated C-Style string` `//  cchPassword: The number of characters in szPassword (not including the NULL terminator)` `WORD GetPasswordHash(const CHAR *szPassword, int cchPassword) {` `  WORD wPasswordHash;` `  const CHAR *pch;` `  wPasswordHash = 0;` `  if (cchPassword > 0)` `  {` `  pch = &szPassword[cchPassword];` `  while (pch-- != szPassword)` `  {` ` wPasswordHash = ((wPasswordHash >> 14) & 0x01)	((wPasswordHash << 1) & 0x7fff);` ` wPasswordHash ^= *pch;` ` }` ` wPasswordHash ^= (0x8000	('N' << 8)	'K');` ` }`

Attributes	Description

	return(wPasswordHash);
	}
	end note]
	
	The possible values for this attribute are defined by the ST_UnsignedShortHex simple type (§3.18.87).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ChartsheetProtection">
   <attribute name="password" type="ST_UnsignedShortHex" use="optional"/>
   <attribute name="content" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="objects" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.3.1.83 sheetView (Worksheet View)

A single sheet view definition. When more than 1 sheet view is defined in the file, it means that when opening the workbook, each sheet view corresponds to a separate window within the spreadsheet application, where each window is showing the particular sheet. containing the same workbookViewId value, the last sheetView definition is loaded, and the others are discarded. When multiple windows are viewing the same sheet, multiple sheetView elements (with corresponding workbookView entries) are saved.

Parent Elements
sheetViews (§3.3.1.85)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
pane (View Pane)	§3.3.1.64
pivotSelection (PivotTable Selection)	§3.3.1.67
selection (Selection)	§3.3.1.75

Attributes	Description
colorId (Color Id)	Index to the color value for row/column text headings and gridlines. This is an 'index color value' (ICV) rather than rgb value. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
defaultGridColor (Default Grid Color)	Flag indicating that the consuming application should use the default grid lines color (system dependent). Overrides any color specified in colorId. The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description
rightToLeft (Right To Left)	Flag indicating whether the sheet is in 'right to left' display mode. When in this mode, Column A is on the far right, Column B ;is one column left of Column A, and so on. Also, information in cells is displayed in the Right to Left format. The possible values for this attribute are defined by the XML Schema boolean datatype.
showFormulas (Show Formulas)	Flag indicating whether this sheet should display formulas. The possible values for this attribute are defined by the XML Schema boolean datatype.
showGridLines (Show Grid Lines)	Flag indicating whether this sheet should display gridlines. The possible values for this attribute are defined by the XML Schema boolean datatype.
showOutlineSymbols (Show Outline Symbols)	Flag indicating whether the sheet has outline symbols visible. This flag shall always override SheetPr element's outlinePr child element whose attribute is named showOutlineSymbols when there is a conflict. The possible values for this attribute are defined by the XML Schema boolean datatype.
showRowColHeaders (Show Headers)	Flag indicating whether the sheet should display row and column headings. The possible values for this attribute are defined by the XML Schema boolean datatype.
showRuler (Show Ruler)	Show the ruler in Page Layout View. The possible values for this attribute are defined by the XML Schema boolean datatype.
showWhiteSpace (Show White Space)	Flag indicating whether page layout view shall display margins. False means do not display left, right, top (header), and bottom (footer) margins (even when there is data in the header or footer). The possible values for this attribute are defined by the XML Schema boolean datatype.
showZeros (Show Zero Values)	Flag indicating whether the window should show 0 (zero) in cells containing zero value. When false, cells with zero value appear blank instead of showing the number zero. The possible values for this attribute are defined by the XML Schema boolean datatype.
tabSelected (Sheet Tab Selected)	Flag indicating whether this sheet is selected. When only 1 sheet is selected and active, this value should be in synch with the activeTab value. In case of a conflict, the Start Part setting wins and sets the active sheet tab. Note: multiple sheets can be selected, but only one sheet can be active at one time. The possible values for this attribute are defined by the XML Schema boolean datatype.
topLeftCell (Top Left Visible Cell)	Location of the top left visible cell Location of the top left visible cell in the bottom right pane (when in Left-to-Right mode). The possible values for this attribute are defined by the ST_CellRef simple type (§3.18.8).

Attributes	Description
view (View Type)	Indicates view type. The possible values for this attribute are defined by the ST_SheetViewType simple type (§3.18.71).
windowProtection (Window Protection)	Flag indicating whether the panes in the window are locked due to workbook protection. This is an option when the workbook structure is protected. The possible values for this attribute are defined by the XML Schema boolean datatype.
workbookViewId (Workbook View Index)	Zero-based index of this workbook view, pointing to a workbookView element in the bookViews collection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
zoomScale (Zoom Scale)	Window zoom magnification for current view representing percent values. Valid values range from 10 to 400. Horizontal & Vertical scale together. For example: 10 - 10% 20 - 20% … 100 - 100% … 400 - 400% Current view can be Normal, Page Layout, or Page Break Preview. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
zoomScaleNormal (Zoom Scale Normal View)	Zoom magnification to use when in normal view, representing percent values. Valid values range from 10 to 400. Horizontal & Vertical scale together. For example: 10 - 10% 20 - 20% … 100 - 100% … 400 - 400% Applies for worksheet sheet type only; zero implies the automatic setting. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

Attributes	Description
zoomScalePageLayoutView (Zoom Scale Page Layout View)	Zoom magnification to use when in page layout view, representing percent values. Valid values range from 10 to 400. Horizontal & Vertical scale together. For example: 10 - 10% 20 - 20% … 100 - 100% … 400 - 400% Applies for worksheet sheet type only; zero implies the automatic setting. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
zoomScaleSheetLayoutView (Zoom Scale Page Break Preview)	Zoom magnification to use when in page break preview, representing percent values. Valid values range from 10 to 400. Horizontal & Vertical scale together. For example: 10 - 10% 20 - 20% … 100 - 100% … 400 - 400% Applies for worksheet only; zero implies the automatic setting. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SheetView">
   <sequence>
      <element name="pane" type="CT_Pane" minOccurs="0" maxOccurs="1"/>
      <element name="selection" type="CT_Selection" minOccurs="0" maxOccurs="4"/>
      <element name="pivotSelection" type="CT_PivotSelection" minOccurs="0" maxOccurs="4"/>
      <element name="extLst" minOccurs="0" maxOccurs="1" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="windowProtection" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="showFormulas" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="showGridLines" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="showRowColHeaders" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="showZeros" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="rightToLeft" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="tabSelected" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="showRuler" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="showOutlineSymbols" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="defaultGridColor" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="showWhiteSpace" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="view" type="ST_SheetViewType" use="optional" default="normal"/>
   <attribute name="topLeftCell" type="ST_CellRef" use="optional"/>
   <attribute name="colorId" type="xsd:unsignedInt" use="optional" default="64"/>
   <attribute name="zoomScale" type="xsd:unsignedInt" use="optional" default="100"/>
   <attribute name="zoomScaleNormal" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="zoomScaleSheetLayoutView" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="zoomScalePageLayoutView" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="workbookViewId" type="xsd:unsignedInt" use="required"/>
</complexType>
```

3.3.1.84 sheetView (Chart Sheet View)

This element specifies a chart sheet view. See sheetView (§3.3.1.83) for an example.

Parent Elements
sheetViews (§3.3.1.86)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10

Attributes	Description
tabSelected (Sheet Tab Selected)	Flag indicating whether the sheet tab is selected. The possible values for this attribute are defined by the XML Schema boolean datatype.
workbookViewId (Workbook View Id)	Zero-based index of this workbook view, pointing to a workbookView element in the bookViews collection.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
zoomScale (Window Zoom Scale)	Window zoom magnification, representing percent values. Valid values range from 10 to 400. Horizontal & Vertical scale together. For example: 10 - 10% 20 - 20% … 100 - 100% … 400 - 400% The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
zoomToFit (Zoom To Fit)	Flag indicating whether chart sheet is zoom to fit window. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ChartsheetView">
   <sequence>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="tabSelected" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="zoomScale" type="xsd:unsignedInt" default="100" use="optional"/>
   <attribute name="workbookViewId" type="xsd:unsignedInt" use="required"/>
   <attribute name="zoomToFit" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.3.1.85 sheetViews (Sheet Views)

Worksheet views collection.

[*Example*:

This example shows one sheet view definition. The definition indicates that the current sheet is the active/selected sheet, and that there is a split pane applied to the view. This definition also indicates for each of the four window panes of the split which cell is the active cell for that pane.

```
<sheetViews>
  <sheetView tabSelected="1" workbookViewId="0">
    <pane xSplit="2310" ySplit="2070" topLeftCell="C1"
      activePane="bottomRight"/>
    <selection/>
    <selection pane="bottomLeft" activeCell="A6" sqref="A6"/>
    <selection pane="topRight" activeCell="C1" sqref="C1"/>
    <selection pane="bottomRight" activeCell="E13" sqref="E13"/>
  </sheetView>
</sheetViews>
```

end example]

Parent Elements
dialogsheet (§3.3.1.32); worksheet (§3.3.1.96)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
sheetView (Worksheet View)	§3.3.1.83

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SheetViews">
   <sequence>
      <element name="sheetView" type="CT_SheetView" minOccurs="1" maxOccurs="unbounded"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

3.3.1.86 sheetViews (Chart Sheet Views)

This element specifies chart sheet views.

Parent Elements
chartsheet (§3.3.1.11)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
sheetView (Chart Sheet View)	§3.3.1.84

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ChartsheetViews">
   <sequence>
      <element name="sheetView" type="CT_ChartsheetView" minOccurs="1" maxOccurs="unbounded"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

3.3.1.87 smartTags (Smart Tags)

This collection expresses all smart tags associated with cells on this sheet. There can be multiple smart tags associated with a particular cell, and many cells with smart tags for a given worksheet.

[*Example*:

This example shows three smart tags, each one associated with a unique cell on the worksheet.

```
<smartTags>
  <cellSmartTags r="A1">
    <cellSmartTag type="0"/>
  </cellSmartTags>
  <cellSmartTags r="B1">
    <cellSmartTag type="0"/>
  </cellSmartTags>
  <cellSmartTags r="B2">
    <cellSmartTag type="0"/>
  </cellSmartTags>
</smartTags>
```

end example]

Parent Elements
worksheet (§3.3.1.96)

Child Elements	Subclause
cellSmartTags (Cell Smart Tags)	§3.3.1.6

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SmartTags">
   <sequence>
      <element name="cellSmartTags" type="CT_CellSmartTags" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.3.1.88 sortCondition (Sort Condition)

Sort condition. When more than one sortCondition is specified, the first condition is applied first, then the second condition is applied, and so on.

Parent Elements
sortState (§3.3.1.89)

Attributes	Description
customList (Custom List)	Sort by a custom list. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
descending (Descending)	Sort descending. The possible values for this attribute are defined by the XML Schema boolean datatype.
dxfId (Format Id)	Format Id when sortBy=cellColor or fontColor The possible values for this attribute are defined by the ST_DxfId simple type (§3.18.26).
iconId (Icon Id)	Zero-based index of an icon in an icon set. The absence of this attribute means "no icon" The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
iconSet (Icon Set)	Icon set index when sortBy=icon. The possible values for this attribute are defined by the ST_IconSetType simple type (§3.18.44).
ref (Reference)	Column/Row that this sort condition applies to. This must be contained within the ref in CT_SortState. The possible values for this attribute are defined by the ST_Ref simple type (§3.18.64).
sortBy (Sort By)	Type of sort. The possible values for this attribute are defined by the ST_SortBy simple type (§3.18.74).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SortCondition">
    <attribute name="descending" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="sortBy" type="ST_SortBy" use="optional" default="value"/>
    <attribute name="ref" type="ST_Ref" use="required"/>
    <attribute name="customList" type="ST_Xstring" use="optional"/>
    <attribute name="dxfId" type="ST_DxfId" use="optional"/>
    <attribute name="iconSet" type="ST_IconSetType" use="optional" default="3Arrows"/>
    <attribute name="iconId" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.3.1.89 sortState (Sort State)

This collection preserves the AutoFilter sort state.

[*Example*: This example shows a sort which is case-sensitive, descending sort. While the range of data to sort is B4:E8, the range to sort by is B4:B8.

```
<sortState caseSensitive="1" ref="B4:E8">
  <sortCondition descending="1" ref="B4:B8"/>
</sortState>
```

end example]

Parent Elements
autoFilter (§3.3.1.1); queryTableRefresh (§3.12.6); table (§3.5.1.2); worksheet (§3.3.1.96)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
sortCondition (Sort Condition)	§3.3.1.88

Attributes	Description
caseSensitive (Case Sensitive)	Flag indicating whether or not the sort is case-sensitive. The possible values for this attribute are defined by the XML Schema boolean datatype.
columnSort (Sort by Columns)	Flag indicating whether or not to sort by columns. Only applies to ranges that don't have AutoFilter applied. The possible values for this attribute are defined by the XML Schema boolean datatype.
ref (Sort Range)	The whole range of data to sort (not just the sort-by column). The possible values for this attribute are defined by the ST_Ref simple type (§3.18.64).
sortMethod (Sort Method)	Strokes or PinYin sort method. Applies only to these application UI languages: • Chinese Simplified

Attributes	Description
	• Chinese Traditional • Japanese For these languages, alternate sort methods can be selected, affecting how the data is sorted. The possible values for this attribute are defined by the ST_SortMethod simple type (§3.18.75).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SortState">
   <sequence>
      <element name="sortCondition" minOccurs="0" maxOccurs="64" type="CT_SortCondition"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="columnSort" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="caseSensitive" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="sortMethod" type="ST_SortMethod" use="optional" default="none"/>
   <attribute name="ref" type="ST_Ref" use="required"/>
</complexType>
```

3.3.1.90 tabColor (Sheet Tab Color)

Background color of the sheet tab.

Parent Elements
sheetPr (§3.3.1.79); sheetPr (§3.3.1.80)

Attributes	Description
auto (Automatic)	A boolean value indicating the color is automatic and system color dependent. The possible values for this attribute are defined by the XML Schema boolean datatype.
indexed (Index)	Indexed color value. Only used for backwards compatibility. References a color in indexedColors. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
rgb (Alpha Red Green Blue Color Value)	Standard Alpha Red Green Blue color value (ARGB). The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§3.18.86).
theme (Theme Color)	Index into the <clrScheme> collection, referencing a particular <sysClr> or <srgbClr> value expressed in the Theme part.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
tint (Tint)	Specifies the tint value applied to the color. If tint is supplied, then it is applied to the RGB value of the color to determine the final color applied. The tint value is stored as a double from -1.0 .. 1.0, where -1.0 means 100% darken and 1.0 means 100% lighten. Also, 0.0 means no change. In loading the RGB value, it is converted to HLS where HLS values are (0..HLSMAX), where HLSMAX is currently 255. [*Example*: Here are some examples of how to apply tint to color: **If (tint < 0)** Lum' = Lum * (1.0 + tint) For example: Lum = 200; tint = -0.5; Darken 50% Lum‘ = 200 * (0.5) => 100 For example: Lum = 200; tint = -1.0; Darken 100% (make black) Lum‘ = 200 * (1.0-1.0) => 0 **If (tint > 0)** Lum‘ = Lum * (1.0-tint) + (HLSMAX – HLSMAX * (1.0-tint)) For example: Lum = 100; tint = 0.75; Lighten 75% Lum‘ = 100 * (1-.75) + (HLSMAX – HLSMAX*(1-.75)) = 100 * .25 + (255 – 255 * .25) = 25 + (255 – 63) = 25 + 192 = 217 For example: Lum = 100; tint = 1.0; Lighten 100% (make white) Lum‘ = 100 * (1-1) + (HLSMAX – HLSMAX*(1-1)) = 100 * 0 + (255 – 255 * 0) = 0 + (255 – 0) = 255 *end example*] The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <attribute name="auto" type="xsd:boolean" use="optional"/>
   <attribute name="indexed" type="xsd:unsignedInt" use="optional"/>
   <attribute name="rgb" type="ST_UnsignedIntHex" use="optional"/>
   <attribute name="theme" type="xsd:unsignedInt" use="optional"/>
   <attribute name="tint" type="xsd:double" use="optional" default="0.0"/>
</complexType>
```

3.3.1.91 tablePart (Table Part)

A single Table Part reference.

Parent Elements
tableParts (§3.3.1.92)

Attributes	Description
id (Relationship Id) Namespace: .../officeDocument/2006/relationships	This relationship Id is used to locate a particular table definition part. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TablePart">
   <attribute ref="r:id" use="required"/>
</complexType>
```

3.3.1.92 tableParts (Table Parts)

This collection expresses a relationship Id pointing to every table on this sheet.

[*Example*: This example indicates that the current sheet has two tables, and their definitions can be found by locating the appropriate relationships from the sheet:

```
<tableParts count="2">
  <tablePart r:id="rId1"/>
  <tablePart r:id="rId2"/>
</tableParts>
```

end example]

Parent Elements
worksheet (§3.3.1.96)

Child Elements	Subclause
tablePart (Table Part)	§3.3.1.91

Attributes	Description
count (Count)	A count of table elements in the collection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TableParts">
   <sequence>
      <element name="tablePart" type="CT_TablePart" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.3.1.93 v (Cell Value)

This element expresses the value contained in a cell. If the cell contains a string, then this value is an index into the shared string table, pointing to the actual string value. Otherwise, the value of the cell is expressed directly in this element. Cells containing formulas express the last calculated result of the formula in this element.

For applications not wanting to implement the shared string table, an 'inline string' may be expressed in an <is> element under <c> (instead of a <v> element under <c>),in the same way a string would be expressed in the shared string table. See <is> for an example.

[*Example*: In this example cell B4 contains the number "360".

```
<c r="B4">
  <v>360</v>
</c>
```

end example]

The possible values for this element are defined by the ST_Xstring simple type (§3.18.96).

Parent Elements
c (§3.3.1.3); cell (§3.14.1); nc (§3.11.1.3); oc (§3.11.1.5); tp (§3.15.3)

3.3.1.94 webPublishItem (Web Publishing Item)

This element represents information for a single item or object which can be published to HTML.

Parent Elements
webPublishItems (§3.3.1.95)

Attributes	Description
autoRepublish (Automatically Publish)	Automatically publish this item every time the workbook is saved. The possible values for this attribute are defined by the XML Schema boolean datatype.
destinationFile (Destination File Name)	Destination file name. Indicates where to save the HTML publish file. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
divId (Destination Bookmark)	Destination bookmark. Identifies a specific section in the published HTML file when a subset of the workbook is published to HTML. Each item that has been published from a workbook is written to a unique element in HTML. On re-publishing a particular item from the workbook, only that item's corresponding content is updated. Therefore each publish item corresponds to a unique element. It is possible to add new publish items to an existing published page, and it is possible to re-publish individual items without republishing the entire workbook. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
id (Id)	This is a unique number "nnnnn" of the webPublishItem. This value is used to generate the divId and styleId values. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
sourceObject (Source Object Name)	Source object name (required for sourceType = pivotTable, query, or label). The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
sourceRef (Source Id)	Source range (required for sourceType = 'range'). The possible values for this attribute are defined by the ST_Ref simple type (§3.18.64).
sourceType (Web Source Type)	Type of web source (or objects to publish). The possible values for this attribute are defined by the ST_WebSourceType simple type (§3.18.94).
title (Title)	HTML title of published item. For example, this value can appear in the web browser window's title bar. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_WebPublishItem">
    <attribute name="id" type="xsd:unsignedInt" use="required"/>
    <attribute name="divId" type="ST_Xstring" use="required"/>
    <attribute name="sourceType" type="ST_WebSourceType" use="required"/>
    <attribute name="sourceRef" type="ST_Ref" use="optional"/>
    <attribute name="sourceObject" type="ST_Xstring" use="optional"/>
    <attribute name="destinationFile" type="ST_Xstring" use="required"/>
    <attribute name="title" type="ST_Xstring" use="optional"/>
    <attribute name="autoRepublish" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.3.1.95 webPublishItems (Web Publishing Items)

This represents a listing of individual objects in this workbook that have been published (to HTML).

Note:When one of these objects is selected to be published, just the object is published to HTML, not the entire workbook contents.

[*Example*: This example shows two items which have been previously selected for publishing. One is a range (A6:C6), the other is a chart, named "Chart 1".

```
<webPublishItems count="2">
  <webPublishItem id="11289" divId="Views_11289" sourceType="range"
    sourceRef="A6:C6" destinationFile="D:\Publish.htm" published="0"/>
  <webPublishItem id="6433" divId="Views_6433" sourceType="chart"
    sourceObject="Chart 1" destinationFile="D:\Publish.mht" published="0"/>
</webPublishItems>
```

end example]

Parent Elements
chartsheet (§3.3.1.11); worksheet (§3.3.1.96)

Child Elements	Subclause
webPublishItem (Web Publishing Item)	§3.3.1.94

Attributes	Description
count (Web Publishing Items Count)	Number of items. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_WebPublishItems">
   <sequence>
      <element name="webPublishItem" type="CT_WebPublishItem" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.3.1.96 worksheet (Worksheet)

This is the root element of Sheet Parts that are of type 'worksheet'.

Parent Elements
Root element of SpreadsheetML Worksheet part

Child Elements	Subclause
autoFilter (AutoFilter Settings)	§3.3.1.1
cellWatches (Cell Watch Items)	§3.3.1.8
colBreaks (Vertical Page Breaks)	§3.3.1.13
cols (Column Information)	§3.3.1.16
conditionalFormatting (Conditional Formatting)	§3.3.1.17
controls (Embedded Controls)	§3.3.1.19
customProperties (Custom Properties)	§3.3.1.21
customSheetViews (Custom Sheet Views)	§3.3.1.25
dataConsolidate (Data Consolidate)	§3.3.1.27
dataValidations (Data Validations)	§3.3.1.31
dimension (Worksheet Dimensions)	§3.3.1.33
drawing (Drawing)	§3.3.1.34
extLst (Future Feature Data Storage Area)	§3.2.10
headerFooter (Header Footer Settings)	§3.3.1.43
hyperlinks (Hyperlinks)	§3.3.1.45
ignoredErrors (Ignored Errors)	§3.3.1.48
legacyDrawing (Legacy Drawing Reference)	§3.3.1.51
legacyDrawingHF (Legacy Drawing Reference in Header Footer)	§3.3.1.52
mergeCells (Merge Cells)	§3.3.1.54
oleObjects (Embedded Objects)	§3.3.1.58
pageMargins (Page Margins)	§3.3.1.60
pageSetup (Page Setup Settings)	§3.3.1.61
phoneticPr (Phonetic Properties)	§3.4.3

Child Elements	Subclause
picture (Background Image)	§3.3.1.65
printOptions (Print Options)	§3.3.1.68
protectedRanges (Protected Ranges)	§3.3.1.70
rowBreaks (Horizontal Page Breaks (Row))	§3.3.1.72
scenarios (Scenarios)	§3.3.1.74
sheetCalcPr (Sheet Calculation Properties)	§3.3.1.76
sheetData (Sheet Data)	§3.3.1.77
sheetFormatPr (Sheet Format Properties)	§3.3.1.78
sheetPr (Sheet Properties)	§3.3.1.79
sheetProtection (Sheet Protection Options)	§3.3.1.81
sheetViews (Sheet Views)	§3.3.1.85
smartTags (Smart Tags)	§3.3.1.87
sortState (Sort State)	§3.3.1.89
tableParts (Table Parts)	§3.3.1.92
webPublishItems (Web Publishing Items)	§3.3.1.95

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Worksheet">
    <sequence>
        <element name="sheetPr" type="CT_SheetPr" minOccurs="0" maxOccurs="1"/>
        <element name="dimension" type="CT_SheetDimension" minOccurs="0" maxOccurs="1"/>
        <element name="sheetViews" type="CT_SheetViews" minOccurs="0" maxOccurs="1"/>
        <element name="sheetFormatPr" type="CT_SheetFormatPr" minOccurs="0" maxOccurs="1"/>
        <element name="cols" type="CT_Cols" minOccurs="0" maxOccurs="unbounded"/>
        <element name="sheetData" type="CT_SheetData" minOccurs="1" maxOccurs="1"/>
        <element name="sheetCalcPr" type="CT_SheetCalcPr" minOccurs="0" maxOccurs="1"/>
        <element name="sheetProtection" type="CT_SheetProtection" minOccurs="0" maxOccurs="1"/>
        <element name="protectedRanges" type="CT_ProtectedRanges" minOccurs="0" maxOccurs="1"/>
        <element name="scenarios" type="CT_Scenarios" minOccurs="0" maxOccurs="1"/>
        <element name="autoFilter" type="CT_AutoFilter" minOccurs="0" maxOccurs="1"/>
        <element name="sortState" type="CT_SortState" minOccurs="0" maxOccurs="1"/>
        <element name="dataConsolidate" type="CT_DataConsolidate" minOccurs="0" maxOccurs="1"/>
        <element name="customSheetViews" type="CT_CustomSheetViews" minOccurs="0" maxOccurs="1"/>
        <element name="mergeCells" type="CT_MergeCells" minOccurs="0" maxOccurs="1"/>
        <element name="phoneticPr" type="CT_PhoneticPr" minOccurs="0" maxOccurs="1"/>
        <element name="conditionalFormatting" type="CT_ConditionalFormatting" minOccurs="0"
          maxOccurs="unbounded"/>
        <element name="dataValidations" type="CT_DataValidations" minOccurs="0" maxOccurs="1"/>
        <element name="hyperlinks" type="CT_Hyperlinks" minOccurs="0" maxOccurs="1"/>
        <element name="printOptions" type="CT_PrintOptions" minOccurs="0" maxOccurs="1"/>
        <element name="pageMargins" type="CT_PageMargins" minOccurs="0" maxOccurs="1"/>
        <element name="pageSetup" type="CT_PageSetup" minOccurs="0" maxOccurs="1"/>
        <element name="headerFooter" type="CT_HeaderFooter" minOccurs="0" maxOccurs="1"/>
        <element name="rowBreaks" type="CT_PageBreak" minOccurs="0" maxOccurs="1"/>
        <element name="colBreaks" type="CT_PageBreak" minOccurs="0" maxOccurs="1"/>
        <element name="customProperties" type="CT_CustomProperties" minOccurs="0" maxOccurs="1"/>
        <element name="cellWatches" type="CT_CellWatches" minOccurs="0" maxOccurs="1"/>
        <element name="ignoredErrors" type="CT_IgnoredErrors" minOccurs="0" maxOccurs="1"/>
        <element name="smartTags" type="CT_SmartTags" minOccurs="0" maxOccurs="1"/>
        <element name="drawing" type="CT_Drawing" minOccurs="0" maxOccurs="1"/>
        <element name="legacyDrawing" type="CT_LegacyDrawing" minOccurs="0" maxOccurs="1"/>
        <element name="legacyDrawingHF" type="CT_LegacyDrawing" minOccurs="0" maxOccurs="1"/>
        <element name="picture" type="CT_SheetBackgroundPicture" minOccurs="0" maxOccurs="1"/>
        <element name="oleObjects" type="CT_OleObjects" minOccurs="0" maxOccurs="1"/>
        <element name="controls" type="CT_Controls" minOccurs="0" maxOccurs="1"/>
        <element name="webPublishItems" type="CT_WebPublishItems" minOccurs="0" maxOccurs="1"/>
        <element name="tableParts" type="CT_TableParts" minOccurs="0" maxOccurs="1"/>
        <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
    </sequence>
</complexType>
```

3.3.2 AutoFilter Settings

The following subclause defines the settings which can be specified as part of an AutoFilter definition. An *AutoFilter* temporarily hides rows based on a filter criteria, which is applied column by column to a table of data in the worksheet.

3.3.2.1 colorFilter (Color Filter Criteria)

This element specifies the color to filter by and whether to use the cell's fill or font color in the filter criteria. If the cell's font or fill color does not match the color specified in the criteria, the rows corresponding to those cells are hidden from view.

[*Example*:

```
<filterColumn colId="1">
  <colorFilter dxfId="0" cellColor="0"/>
</filterColumn>
```

end example]

Parent Elements
filterColumn (§3.3.2.7)

Attributes	Description
cellColor (Filter By Cell Color)	Flag indicating whether or not to filter by the cell's fill color. '1' indicates to filter by cell fill. '0' indicates to filter by the cell's font color. For rich text in cells, if the color specified appears in the cell at all, it shall be included in the filter. The possible values for this attribute are defined by the XML Schema boolean datatype.
dxfId (Differential Format Record Id)	Id of differential format record (dxf) in the Styles Part which expresses the color value to filter by. The possible values for this attribute are defined by the ST_DxfId simple type (§3.18.26).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ColorFilter">
   <attribute name="dxfId" type="ST_DxfId" use="optional"/>
   <attribute name="cellColor" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

3.3.2.2 customFilter (Custom Filter Criteria)

A custom AutoFilter specifies an operator and a value. There can be at most two customFilters specified, and in that case the parent element specifies whether the two conditions are joined by 'and' or 'or'. For any cells whose values do not meet the specified criteria, the corresponding rows shall be hidden from view when the fitler is applied.

[*Example*:

```
<customFilters and="1">
  <customFilter operator="greaterThanOrEqual" val="0.2"/>
  <customFilter operator="lessThanOrEqual" val="0.5"/>
</customFilters>
```

end example]

Parent Elements
customFilters (§3.3.2.3)

Attributes	Description
operator (Filter Comparison Operator)	Operator used by the filter comparison. The possible values for this attribute are defined by the ST_FilterOperator simple type (§3.18.32).
val (Top or Bottom Value)	Top or bottom value used in the filter criteria. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustomFilter">
   <attribute name="operator" type="ST_FilterOperator" default="equal" use="optional"/>
   <attribute name="val" type="ST_Xstring"/>
</complexType>
```

3.3.2.3 customFilters (Custom Filters)

When there is more than one custom filter criteria to apply (an 'and' or 'or' joining two criteria), then this element groups the customFilter elements together.

Parent Elements
filterColumn (§3.3.2.7)

Child Elements	Subclause
customFilter (Custom Filter Criteria)	§3.3.2.2

Attributes	Description
and (And)	Flag indicating whether the two criterias have an "and" relationship. '1' indicates "and", '0' indicates "or". The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustomFilters">
   <sequence>
      <element name="customFilter" type="CT_CustomFilter" minOccurs="1" maxOccurs="2"/>
   </sequence>
   <attribute name="and" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.3.2.4 dateGroupItem (Date Grouping)

This collection is used to express a group of dates or times which are used in an AutoFilter criteria. See parent element for an example. Values are always written in the calendar type of the first date encountered in the filter range, so that all subsequent dates, even when formatted or represented by other calendar types, can be correctly compared for the purposes of filtering.

Parent Elements
filters (§3.3.2.8)

Attributes	Description
dateTimeGrouping (Date Time Grouping)	Grouping level. The possible values for this attribute are defined by the ST_DateTimeGrouping simple type (§3.18.23).
day (Day)	Day (1-31) The possible values for this attribute are defined by the XML Schema unsignedShort datatype.
hour (Hour)	Hour (0-23) The possible values for this attribute are defined by the XML Schema unsignedShort datatype.
minute (Minute)	Minute (0-59) The possible values for this attribute are defined by the XML Schema unsignedShort datatype.
month (Month)	Month (1-12) The possible values for this attribute are defined by the XML Schema unsignedShort datatype.
second (Second)	Second (0-59) The possible values for this attribute are defined by the XML Schema unsignedShort datatype.
year (Year)	Year (4 digits)

Attributes	Description
	The possible values for this attribute are defined by the XML Schema unsignedShort datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DateGroupItem">
   <attribute name="year" type="xsd:unsignedShort" use="required"/>
   <attribute name="month" type="xsd:unsignedShort" use="optional"/>
   <attribute name="day" type="xsd:unsignedShort" use="optional"/>
   <attribute name="hour" type="xsd:unsignedShort" use="optional"/>
   <attribute name="minute" type="xsd:unsignedShort" use="optional"/>
   <attribute name="second" type="xsd:unsignedShort" use="optional"/>
   <attribute name="dateTimeGrouping" type="ST_DateTimeGrouping" use="required"/>
</complexType>
```

3.3.2.5 dynamicFilter (Dynamic Filter)

This collection specifies dynamic filter criteria. These criteria are considered dynamic because they can change, either with the data itself (e.g., "above average") or with the current system date (e.g., show values for "today"). For any cells whose values do not meet the specified criteria, the corresponding rows shall be hidden from view when the fitler is applied.

[*Example*:

```
<filterColumn colId="0">
  <dynamicFilter type="today"/>
</filterColumn>
```

end example]

Parent Elements
filterColumn (§3.3.2.7)

Attributes	Description
maxVal (Max Value)	A maximum value for dynamic filter. It shall be required for today, yesterday, tomorrow, nextWeek, thisWeek, lastWeek, nextMonth, thisMonth, lastMonth, nextQuarter, thisQuarter, lastQuarter, nextYear, thisYear, lastYear, and yearToDate. The above criteria are based on a value range. For example, if today's date is September 22nd, then the range for thisWeek is the values greater than or equal to September 17 and less than September 24. In the thisWeek range, the lower value is expressed using val. The higher value is expressed using maxVal. These dynamic filter shall not require val / maxVal: Q1, Q2, Q3, Q4,

Attributes	Description
	M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, M11 and M12. The above criteria shall not specify the range using val and maxVal because Q1 always starts from M1 to M3, and M1 is always January. These types of dynamic filters shall use val and shall not use maxVal: aboveAverage and belowAverage The possible values for this attribute are defined by the XML Schema double datatype.
type (Dynamic filter type)	Dynamic filter type, e.g., “today” or “nextWeek”. The possible values for this attribute are defined by the ST_DynamicFilterType simple type (§3.18.27).
val (Value)	A minimum value for dynamic filter. See description of maxVal to understand when val is required. The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DynamicFilter">
   <attribute name="type" type="ST_DynamicFilterType" use="required"/>
   <attribute name="val" type="xsd:double" use="optional"/>
   <attribute name="maxVal" type="xsd:double" use="optional"/>
</complexType>
```

3.3.2.6 filter (Filter)

This element expresses a filter criteria value.

[*Example*:

```
<filters>
  <filter val="0.316588716"/>
  <filter val="0.667439395"/>
  <filter val="0.823086999"/>
</filters>
```

end example]

Parent Elements
filters (§3.3.2.8)

Attributes	Description
val (Filter Value)	Filter value used in the criteria.

Attributes	Description
	The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Filter">
   <attribute name="val" type="ST_Xstring"/>
</complexType>
```

3.3.2.7 filterColumn (AutoFilter Column)

The filterColumn collection identifies a particular column in the AutoFilter range and specifies filter information that has been applied to this column. If a column in the AutoFilter range has no criteria specified, then there is no corresponding filterColumn collection expressed for that column.

Parent Elements
autoFilter (§3.3.1.1)

Child Elements	Subclause
colorFilter (Color Filter Criteria)	§3.3.2.1
customFilters (Custom Filters)	§3.3.2.3
dynamicFilter (Dynamic Filter)	§3.3.2.5
extLst (Future Feature Data Storage Area)	§3.2.10
filters (Filter Criteria)	§3.3.2.8
iconFilter (Icon Filter)	§3.3.2.9
top10 (Top 10)	§3.3.2.10

Attributes	Description
colId (Filter Column Data)	Zero-based index indicating the AutoFilter column to which this filter information applies. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
hiddenButton (Hidden AutoFilter Button)	Flag indicating whether the AutoFilter button for this column is hidden. The possible values for this attribute are defined by the XML Schema boolean datatype.
showButton (Show Filter Button)	Flag indicating whether the filter button is visible. For example, when the cell containing the filter button is merged with another cell, the filter button may be hidden, and not drawn. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FilterColumn">
   <choice minOccurs="0" maxOccurs="1">
      <element name="filters" type="CT_Filters" minOccurs="0" maxOccurs="1"/>
      <element name="top10" type="CT_Top10" minOccurs="0" maxOccurs="1"/>
      <element name="customFilters" type="CT_CustomFilters" minOccurs="0" maxOccurs="1"/>
      <element name="dynamicFilter" type="CT_DynamicFilter" minOccurs="0" maxOccurs="1"/>
      <element name="colorFilter" type="CT_ColorFilter" minOccurs="0" maxOccurs="1"/>
      <element name="iconFilter" minOccurs="0" maxOccurs="1" type="CT_IconFilter"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </choice>
   <attribute name="colId" type="xsd:unsignedInt" use="required"/>
   <attribute name="hiddenButton" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="showButton" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

3.3.2.8 filters (Filter Criteria)

When multiple values are chosen to filter by, or when a group of date values are chosen to filter by, this element groups those criteria together.

[*Example*:

```
<filters>
  <dateGroupItem year="2006" month="1" day="2" dateTimeGrouping="day"/>
  <dateGroupItem year="2005" month="1" day="2" dateTimeGrouping="day"/>
</filters>
```

end example]

Parent Elements
filterColumn (§3.3.2.7)

Child Elements	Subclause
dateGroupItem (Date Grouping)	§3.3.2.4
filter (Filter)	§3.3.2.6

Attributes	Description
blank (Filter by Blank)	Flag indicating whether to filter by blank. The possible values for this attribute are defined by the XML Schema boolean datatype.
calendarType (Calendar Type)	Calendar type for date grouped items. Used to interpret the values in dateGroupItem. This is the calendar type used to evaluate all dates in the filter column, even when those dates are not using the same calendar system / date formatting.

Attributes	Description
	The possible values for this attribute are defined by the ST_CalendarType simple type (§3.18.5).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Filters">
   <sequence>
      <element name="filter" type="CT_Filter" minOccurs="0" maxOccurs="unbounded"/>
      <element name="dateGroupItem" type="CT_DateGroupItem" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="blank" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="calendarType" type="ST_CalendarType" use="optional" default="none"/>
</complexType>
```

3.3.2.9 iconFilter (Icon Filter)

This element specifies the icon set and particular icon within that set to filter by. For any cells whose icon does not match the specified criteria, the corresponding rows shall be hidden from view when the filter is applied.

[*Example*:

```
<filterColumn colId="3">
  <iconFilter iconSet="3Arrows" iconId="0"/>
</filterColumn>
```

end example]

Parent Elements
filterColumn (§3.3.2.7)

Attributes	Description
iconId (Icon Id)	Zero-based index of an icon in an icon set. The absence of this attribute means "no icon" The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
iconSet (Icon Set)	Specifies which icon set is used in the filter criteria. The possible values for this attribute are defined by the ST_IconSetType simple type (§3.18.44).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_IconFilter">
   <attribute name="iconSet" type="ST_IconSetType" use="required"/>
   <attribute name="iconId" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.3.2.10 top10 (Top 10)

This element specifies the top N (percent or number of items) to filter by.

[*Example*: This example filters the first column by the top 10 percent of the values in that column. For all cells in the column whose value falls outside the top 10 percent of the value in that column, the rows corresponding to those cells are hidden from the view. In this example, there are 6 cells in the range, containing 1, 2, 3, 4, 5, 6 repsectively.

```
<filterColumn colId="0">
  <top10 percent="1" val="5" filterVal="6"/>
</filterColumn
```

end example]

Parent Elements
filterColumn (§3.3.2.7)

Attributes	Description
filterVal (Filter Value)	The actual cell value in the range which is used to perform the comparison for this filter. The possible values for this attribute are defined by the XML Schema double datatype.
percent (Filter by Percent)	Flag indicating whether or not to filter by percent value of the column. A false value filters by number of items. The possible values for this attribute are defined by the XML Schema boolean datatype.
top (Top)	Flag indicating whether or not to filter by top order. A false value filters by bottom order. The possible values for this attribute are defined by the XML Schema boolean datatype.
val (Top or Bottom Value)	Top or bottom value to use as the filter criteria. For example "Filter by Top 10 Percent" or "Filter by Top 5 Items". The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Top10">
   <attribute name="top" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="percent" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="val" type="xsd:double" use="required"/>
   <attribute name="filterVal" type="xsd:double" use="optional"/>
</complexType>
```

3.4 Shared String Table

A workbook may contain thousands of cells containing string (non-numeric) data. Furthermore this data is very likely to be repeated across many rows or columns. The goal of implementing a single string table that is shared across the workbook is to improve performance in opening and saving the file by only reading and writing the repetitive information once.

Consider for example a workbook summarizing information for cities within various countries. There may be a column for the name of the country, a column for the name of each city in that country, and a column containing the data for each city. In this case the country name is repetitive, being duplicated in many cells. In many cases the repetition is extensive, and a tremendous savings is realized by making use of a shared string table when saving the workbook. When displaying text in the spreadsheet, the cell table will just contain an index into the string table as the value of a cell, instead of the full string.

The shared string table contains all the necessary information for displaying the string: the text, formatting properties, and phonetic properties (for East Asian languages).

Most strings in a workbook have formatting applied at the cell level, that is, the entire string in the cell has the same formatting applied. In these cases, the formatting for the cell is stored in the styles part, and the string for the cell can be stored in the shared strings table. In this case, the strings stored in the shared strings table are very simple text elements, and the following xml illustrates the example.

[*Example*:

```
<sst xmlns=http://schemas.openxmlformats.org/spreadsheetml/2006/5/main
  count="8" uniqueCount="4">
  <si>
    <t>United States</t>
  </si>
  <si>
    <t>Seattle</t>
  </si>
  <si>
    <t>Denver</t>
  </si>
  <si>
    <t>New York</t>
  </si>
</sst>
```

end example]

In the above example we can see that the string table is just a collection of string items that consist of simple text elements. Note that any numeric data in the workbook is not shown in the shared string table.

Some strings in the workbook may have formatting applied at a level that is more granular than the cell level. For instance, specific characters within the string may be bolded, have coloring, italicizing, etc. In these cases, the formatting is stored along with the text in the string table, and is treated as a unique entry in the table. The following xml illustrates this.

[*Example*:

```
<sst xmlns=http://schemas.openxmlformats.org/spreadsheetml/2006/5/main
  count="8" uniqueCount="4">
  <si>
    <r>
      <t xml:space="preserve">United </t>
    </r>
    <r>
      <rPr>
        <sz val="11"/>
        <color rgb="FFFF0000"/>
        <rFont val="Calibri"/>
        <family val="2"/>
        <scheme val="minor"/>
      </rPr>
      <t>States</t>
    </r>
  </si>
  <si>
    <t>Seattle</t>
  </si>
  <si>
    <t>Denver</t>
  </si>
  <si>
    <t>New York</t>
  </si>
</sst>
```

In the above example you can see that this time, the text "United States" has specific, colored, formatting applied to the text, "States." *end example*]

3.4.1 charset (Character Set)

This element defines the font character set of this font.

This field is used in font creation and selection if a font of the given facename is not available on the system. Although it is not required to have around when resolving font facename, the information can be stored for when needed to help resolve which font face to use of all available fonts on a system.

Charset represents the basic set of characters associated with a font (that it can display), and roughly corresponds to the ANSI codepage (8-bit or DBCS) of that character set used by a given language. Given more common use of Unicode where many fonts support more than one of the traditional charset categories, and the use of font linking, using charset to resolve font name is less and less common, but still can be useful.

These are operating-system-dependent values.

[*Note*: The following are some of the possible the character sets:

INT Value	Character Set
0	ANSI_CHARSET
1	DEFAULT_CHARSET
2	SYMBOL_CHARSET
77	MAC_CHARSET
128	SHIFTJIS_CHARSET
129	HANGEUL_CHARSET
129	HANGUL_CHARSET
130	JOHAB_CHARSET
134	GB2312_CHARSET
136	CHINESEBIG5_CHARSET
161	GREEK_CHARSET
162	TURKISH_CHARSET
163	VIETNAMESE_CHARSET
177	HEBREW_CHARSET
178	ARABIC_CHARSET
186	BALTIC_CHARSET
204	RUSSIAN_CHARSET
222	THAI_CHARSET
238	EASTEUROPE_CHARSET
255	OEM_CHARSET

The OEM_CHARSET value specifies a character set that is operating-system dependent. *end note*]

Fonts with other character sets may exist in the operating system. If an application uses a font with an unknown character set, it should not attempt to translate or interpret strings that are rendered with that font.

Parent Elements
font (§3.8.21); rPr (§3.4.7)

Attributes	Description
val (Value)	The value of an integer, where each value corresponds to a different character set. Valid values are 0 to 255. The possible values for this attribute are defined by the XML Schema int datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_IntProperty">
   <attribute name="val" type="xsd:int" use="required"/>
</complexType>
```

3.4.2 outline (Outline)

This element displays only the inner and outer borders of each character. This is very similar to Bold in behavior.

Parent Elements
font (§3.8.21); rPr (§3.4.7)

Attributes	Description
val (Value)	A boolean value for the property specified by the parent XML element. If omitted, the default value is `true`. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BooleanProperty">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

3.4.3 phoneticPr (Phonetic Properties)

This element represents a collection of phonetic properties that affect the display of phonetic text for this String Item (si).

Phonetic text is used to give hints as to the pronunciation of an East Asian language, and the hints are displayed as text within the spreadsheet cells across the top portion of the cell. Since the phonetic hints are text, every phonetic hint is expressed as a phonetic run (rPh), and these properties specify how to display that phonetic run.

[*Example*:

```
<si>
  <t>課く 毛</t>
  <rPh sb="0" eb="1">
    <t>カ</t>
  </rPh>
  <rPh sb="4" eb="5">
    <t>ケ</t>
  </rPh>
  <phoneticPr fontId="1"/>
</si>
```

end example]

The above example shows a String Item that displays some Japanese text "課く 毛" It also displays some phonetic text across the top of the cell. The phonetic text character, "カ' is displayed over the "課 character and the phonetic text "ゲ is displayed above the "毛 character, using the font record in the style sheet at index 1.

Parent Elements
is (§3.3.1.50); si (§3.4.8); text (§3.7.6); worksheet (§3.3.1.96)

Attributes	Description
alignment (Alignment)	Specifies how the text for the phonetic run is aligned across the top of the cells, with respect to the main text in the body of the cell. The possible values for this attribute are defined by the ST_PhoneticAlignment simple type (§3.18.58).
fontId (Font Id)	An integer that is a zero-based index into the font record in the style sheet. Represents the font to be used to display this phonetic run. If this index is out of bounds, then the default font of the Normal style should be used in its place. This default font should be at index 0. The possible values for this attribute are defined by the ST_FontId simple type (§3.18.33).
type (Character Type)	An enumeration which specifies which type of East Asian character set should be used to display the phonetic run The possible values for this attribute are defined by the ST_PhoneticType simple type (§3.18.59).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PhoneticPr">
   <attribute name="fontId" type="ST_FontId" use="required"/>
   <attribute name="type" type="ST_PhoneticType" use="optional" default="fullwidthKatakana"/>
   <attribute name="alignment" type="ST_PhoneticAlignment" use="optional" default="left"/>
</complexType>
```

3.4.4 r (Rich Text Run)

This element represents a run of rich text. A rich text run is a region of text that share a common set of properties, such as formatting properties. The properties are defined in the rPr element, and the text displayed to the user is defined in the Text (t) element.

Parent Elements
is (§3.3.1.50); si (§3.4.8); text (§3.7.6)

Child Elements	Subclause
rPr (Run Properties)	§3.4.7
t (Text)	§3.4.12

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RElt">
   <sequence>
      <element name="rPr" type="CT_RPrElt" minOccurs="0" maxOccurs="1"/>
      <element name="t" type="ST_Xstring" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

3.4.5 rFont (Font)

This element is a string representing the name of the font assigned to display this run.

Parent Elements
rPr (§3.4.7)

Attributes	Description
val (String Value)	A string representing the name of the font. If the font doesn't exist (because it isn't installed on the system), or the charset is invalid for that font, then another font should be substituted. The string length for this attribute shall be 0 to 31 characters. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FontName">
   <attribute name="val" type="ST_Xstring" use="required"/>
</complexType>
```

3.4.6 rPh (Phonetic Run)

This element represents a run of text which displays a phonetic hint for this String Item (si).

Phonetic hints are used to give information about the pronunciation of an East Asian language. The hints are displayed as text within the spreadsheet cells across the top portion of the cell.

Parent Elements
is (§3.3.1.50); si (§3.4.8); text (§3.7.6)

Child Elements	Subclause
t (Text)	§3.4.12

Attributes	Description
eb (Base Text End Index)	An integer used as a zero-based index representing the ending offset into the base text for this phonetic run. This represents the ending point in the base text the phonetic hint applies to. This value shall be between 0 and the total length of the base text. The following condition shall be true: `sb < eb`. It is recommended that the following condition also be satisfied: That for any two consecutive phonetic runs, `sb1 < eb1 <= sb2 < eb2` to avoid overlapping phonetic runs The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
sb (Base Text Start Index)	An integer used as a zero-based index representing the starting offset into the base text for this phonetic run. This represents the starting point in the base text the phonetic hint applies to. This value shall be between 0 and the total length of the base text. The following condition shall be true: `sb < eb`. It is recommended that the following condition also be satisfied: That for any two consecutive phonetic runs, `sb1 < eb1 <= sb2 < eb2` to avoid overlapping phonetic runs. The possible values for this attribute are defined by the XML Schema unsignedInt

Attributes	Description
	datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PhoneticRun">
   <sequence>
      <element name="t" type="ST_Xstring" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="sb" type="xsd:unsignedInt" use="required"/>
   <attribute name="eb" type="xsd:unsignedInt" use="required"/>
</complexType>
```

3.4.7 rPr (Run Properties)

This element represents a set of properties to apply to the contents of this rich text run.

Parent Elements
r (§3.4.4)

Child Elements	Subclause
b (Bold)	§3.8.2
charset (Character Set)	§3.4.1
color (Data Bar Color)	§3.3.1.14
condense (Condense)	§3.8.12
extend (Extend)	§3.8.16
family (Font Family)	§3.8.17
i (Italic)	§3.8.25
outline (Outline)	§3.4.2
rFont (Font)	§3.4.5
scheme (Scheme)	§3.8.36
shadow (Shadow)	§3.8.37
strike (Strike Through)	§3.4.10
sz (Font Size)	§3.4.11
u (Underline)	§3.4.13
vertAlign (Vertical Alignment)	§3.4.14

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RPrElt">
   <choice maxOccurs="unbounded">
      <element name="rFont" type="CT_FontName" minOccurs="0" maxOccurs="1"/>
      <element name="charset" type="CT_IntProperty" minOccurs="0" maxOccurs="1"/>
      <element name="family" type="CT_IntProperty" minOccurs="0" maxOccurs="1"/>
      <element name="b" type="CT_BooleanProperty" minOccurs="0" maxOccurs="1"/>
      <element name="i" type="CT_BooleanProperty" minOccurs="0" maxOccurs="1"/>
      <element name="strike" type="CT_BooleanProperty" minOccurs="0" maxOccurs="1"/>
      <element name="outline" type="CT_BooleanProperty" minOccurs="0" maxOccurs="1"/>
      <element name="shadow" type="CT_BooleanProperty" minOccurs="0" maxOccurs="1"/>
      <element name="condense" type="CT_BooleanProperty" minOccurs="0" maxOccurs="1"/>
      <element name="extend" type="CT_BooleanProperty" minOccurs="0" maxOccurs="1"/>
      <element name="color" type="CT_Color" minOccurs="0" maxOccurs="1"/>
      <element name="sz" type="CT_FontSize" minOccurs="0" maxOccurs="1"/>
      <element name="u" type="CT_UnderlineProperty" minOccurs="0" maxOccurs="1"/>
      <element name="vertAlign" type="CT_VerticalAlignFontProperty" minOccurs="0" maxOccurs="1"/>
      <element name="scheme" type="CT_FontScheme" minOccurs="0" maxOccurs="1"/>
   </choice>
</complexType>
```

3.4.8 si (String Item)

This element is the representation of an individual string in the Shared String table.

If the string is just a simple string with formatting applied at the cell level, then the String Item (si) should contain a single text element used to express the string. However, if the string in the cell is more complex - i.e., has formatting applied at the character level - then the string item shall consist of multiple rich text runs which collectively are used to express the string.

Parent Elements
sst (§3.4.9)

Child Elements	Subclause
phoneticPr (Phonetic Properties)	§3.4.3
r (Rich Text Run)	§3.4.4
rPh (Phonetic Run)	§3.4.6
t (Text)	§3.4.12

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Rst">
   <sequence>
      <element name="t" type="ST_Xstring" minOccurs="0" maxOccurs="1"/>
      <element name="r" type="CT_RElt" minOccurs="0" maxOccurs="unbounded"/>
      <element name="rPh" type="CT_PhoneticRun" minOccurs="0" maxOccurs="unbounded"/>
      <element name="phoneticPr" minOccurs="0" maxOccurs="1" type="CT_PhoneticPr"/>
   </sequence>
</complexType>
```

3.4.9 sst (Shared String Table)

This element is the root of the Shared String Table, which serves as a collection of individual String Items (si).

Parent Elements
Root element of SpreadsheetML Shared String Table part

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
si (String Item)	§3.4.8

Attributes	Description
count (String Count)	An integer representing the total count of strings in the workbook. This count does not include any numbers, it counts only the total of text strings in the workbook. This attribute is optional unless `uniqueCount` is used, in which case it is required. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
uniqueCount (Unique String Count)	An integer representing the total count of unique strings in the Shared String Table. A string is unique even if it is a copy of another string, but has different formatting applied at the character level. [*Example*: World, *World*, and *W*orld. The `count` would be 3, and the `uniqueCount` would be 2. Only one entry for "World" would show in the table because it is the same string, just with different formatting applied at the cell level (i.e., applied to the entire string in the cell). The "*W*orld" string would get a separate unique entry in the shared string table because it has different formatting applied to specific characters. *end example*] This attribute is optional unless `count` is used, in which case it is required.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Sst">
   <sequence>
      <element name="si" type="CT_Rst" minOccurs="0" maxOccurs="unbounded"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
   <attribute name="uniqueCount" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.4.10 strike (Strike Through)

This element draws a strikethrough line through the horizontal middle of the text.

Parent Elements
font (§3.8.21); rPr (§3.4.7)

Attributes	Description
val (Value)	A boolean value for the property specified by the parent XML element. If omitted, the default value is `true`. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BooleanProperty">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

3.4.11 sz (Font Size)

This element represents the point size (1/72 of an inch) of the Latin and East Asian text.

Parent Elements
font (§3.8.21); rPr (§3.4.7)

Attributes	Description
val (Value)	A double representing the value of a positive measurement in points (1/72 of an inch).

Attributes	Description
	The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FontSize">
   <attribute name="val" type="xsd:double" use="required"/>
</complexType>
```

3.4.12 t (Text)

This element represents the text content shown as part of a string.

The possible values for this element are defined by the ST_Xstring simple type (§3.18.96).

Parent Elements
is (§3.3.1.50); r (§3.4.4); rPh (§3.4.6); si (§3.4.8); text (§3.7.6)

3.4.13 u (Underline)

This element represents the underline formatting style.

Parent Elements
font (§3.8.21); rPr (§3.4.7)

Attributes	Description
val (Underline Value)	An enumeration representing the style of underlining that is used. The none style is equivalent to not using underlining at all. The possible values for this attribute are defined by the ST_UnderlineValues simple type (§3.18.85).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_UnderlineProperty">
   <attribute name="val" type="ST_UnderlineValues" use="optional" default="single"/>
</complexType>
```

3.4.14 vertAlign (Vertical Alignment)

This element adjusts the vertical position of the text relative to the text's default appearance for this run. It is used to get 'superscript' or 'subscript' texts, and shall reduce the font size (if a smaller size is available) accordingly.

Parent Elements
font (§3.8.21); rPr (§3.4.7)

Attributes	Description
val (Value)	An enumeration representing the vertical-alignment setting. Setting this to either `subscript` or `superscript` shall make the font size smaller if a smaller font size is available. The possible values for this attribute are defined by the ST_VerticalAlignRun simple type (§3.18.90).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_VerticalAlignFontProperty">
   <attribute name="val" type="ST_VerticalAlignRun" use="required"/>
</complexType>
```

3.5 Tables

A table helps organize and provide structure to lists of information in a worksheet. Tables have clearly labeled columns, rows, and data regions. Tables make it easier for users to sort, analyze, format, manage, add, and delete information.

If a region of data is designated as a Table, then special behaviors can be applied which help the user perform useful actions. For example, if the user types additional data in the row adjacent to the bottom of the table, the table can expand and automatically add that data to the data region of the table. Similarly, adding a column is as easy as typing a new column heading to the right or left of the current column headings. Filter and sort abilities can automatically be surfaced to the user via the drop down arrows. Special calculated columns can be created which summarize or calculate data in the table. These columns have the ability to expand and shrink according to size of the table, and maintain proper formula referencing.

Tables can be created from data already present in the worksheet, from an external data query, or from mapping a collection of repeating XML elements to a worksheet range.

The sheet XML stores the numeric and textual data. The table XML records the various attributes for the particular table object.

[*Example*:

```
<table xmlns=http://schemas.openxmlformats.org/spreadsheetml/2006/5/main
  id="1" name="MarginTable" displayName="MarginTable" ref="D3:G6"
  totalsRowShown="0">
```

```
    <autoFilter ref="D3:G6"/>
    <tableColumns count="4">
      <tableColumn id="1" name="Product"/>
      <tableColumn id="2" name="Wholesale"/>
      <tableColumn id="3" name="Retail"/>
      <tableColumn id="4" name="Margin" dataDxfId="0">
        <calculatedColumnFormula d="1">[Retail]-
          [Wholesale]</calculatedColumnFormula>
      </tableColumn>
    </tableColumns>
    <tableStyleInfo name="TableStyleMedium9" showFirstColumn="0"
      showLastColumn="0" showRowStripes="1" showColumnStripes="0"/>
  </table>
```

end example]

The above xml example shows a table that spans cells D3 through G6, and has four columns: Product, Wholesale, Retail, and Margin. Margin is a column where each cell has its values calculated based on the formula (Retail - Wholesale), where those values are taken from the cells in the table columns on the corresponding row. The table has a style applied, "TableStyleMedium9", but the styles formatting isn't applied to the first column and the column striping isn't shown. Note that all the data and text values are stored in the sheet xml; the table xml just stores the properties that are specific to this table, and it is referenced by the sheet.

3.5.1 Tables

Tables are ranges of data in the worksheet that have special behavior applied which allow users to better sort, analyze, format, manage, add, and delete data. Tables and table columns can also be referenced through formulas by the spreadsheet application using friendly names, making formula calculations that use tables much easier to understand and maintain. Tables provide a natural way for working with large sets of tabular data.

The tables described in this section are of the multi cell variety, as opposed to single cell tables created from XML mappings.

Each table gets its own xml part, and the relationship between a table part and the sheet is defined in the sheet's _rels directory. The sheet xml also references this id since there can be more than one table on a sheet. The sheet xml contains all the numeric and textual data, and the table xml records properties of the table as well as some formatting rules for data and text displayed in the table cells.

3.5.1.1 calculatedColumnFormula (Calculated Column Formula)

Columns in a table can have cells that are calculated, usually based on values in other cells in the table. This element stores the formula that is used to perform the calculation for each cell in this column.

It shall be understood that formulas which reference columns of this table, shall be calculated using the cells in those columns on the same row of the table as the cell that the formula resides in.

See §3.17 for details on valid/invalid formulas.

Parent Elements
tableColumn (§3.5.1.3)

Attributes	Description
array (Array)	A Boolean value that indicates whether this formula is an array style formula. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TableFormula">
   <simpleContent>
      <extension base="ST_Formula">
         <attribute name="array" type="xsd:boolean" default="false"/>
      </extension>
   </simpleContent>
</complexType>
```

3.5.1.2 table (Table)

This element is the root element for a table that is not a single cell XML table.

Parent Elements
Root element of SpreadsheetML Table Definitions part

Child Elements	Subclause
autoFilter (AutoFilter Settings)	§3.3.1.1
extLst (Future Feature Data Storage Area)	§3.2.10
sortState (Sort State)	§3.3.1.89
tableColumns (Table Columns)	§3.5.1.4
tableStyleInfo (Table Style)	§3.5.1.5

Attributes	Description
comment (Table Comment)	A string representing a textual comment about the table. Note: this can be used by the spreadsheet application in other UI. For example, there may be name UI that is used to organize defined names and function references, if tables are listed in that UI the comment can give more information about the table. The maximum length of this string should be 32767 characters.

Attributes	Description
	The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
connectionId (Connection ID)	An integer representing an ID to indicate which connection from the connections collection is used by this table. This shall only be used for tables that are based off of xml maps. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
dataCellStyle (Data Style Name)	A string representing the name of the cell style that is applied to the data area cells of the table. If this string is missing or invalid, then the data cell style specified by the current table style should be applied. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
dataDxfId (Data Area Format Id)	A zero based integer index into the differential formatting records `<dxfs>` in the styleSheet indicating which format to apply to the data area of this table. The spreadsheet should fail to load if this index is out of bounds. The possible values for this attribute are defined by the ST_DxfId simple type (§3.18.26).
displayName (Table Name)	A string representing the name of the table. This is the name that shall be used in formula references, and displayed in the UI to the spreadsheet user. This name shall not have any spaces in it, and it must be unique amongst all other `displayNames` and `definedNames` in the workbook. The character lengths and restrictions are the same as for `definedNames`. See *SpreadsheetML Reference - Workbook definedNames* section for details The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
headerRowBorderDxfId (Header Row Border Format Id)	A zero based integer index into the differential formatting records `<dxfs>` in the styleSheet indicating what border formatting to apply to the header row of this table. The spreadsheet should fail to load if this index is out of bounds. The possible values for this attribute are defined by the ST_DxfId simple type (§3.18.26).
headerRowCellStyle (Header Row Style)	A string representing the name of the cell style that is applied to the header row cells of the table. If this string is missing or invalid, then the header row style specified by the current table

Attributes	Description
	style should be applied. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
headerRowCount (Header Row Count)	An integer representing the number of header rows showing at the top of the table. 0 means that the header row is not shown. It is up to the spreadsheet application to determine if numbers greater than 1 are allowed. Unless the spreadsheet application has a feature where there may ever be more than one header row, this number should not be higher than 1. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
headerRowDxfId (Header Row Format Id)	A zero based integer index into the differential formatting records <dxfs>in the styleSheet indicating which format to apply to the header row of this table. The spreadsheet should fail to load if this index is out of bounds. The possible values for this attribute are defined by the ST_DxfId simple type (§3.18.26).
id (Table Id)	A non zero integer representing the unique identifier for this table. Each table in the workbook shall have a unique id. Note: Ids can be used to refer to the specific table in the workbook. For instance a future records bucket could refer to the table using this id. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
insertRow (Insert Row Showing)	A Boolean value indicating whether the insert row is showing. True when the insert row is showing, false otherwise. The insert row should only be shown if the table has no data. Note: When a user clicks the insert row in the UI, it provides them an easy way to enter data into a table. The possible values for this attribute are defined by the XML Schema boolean datatype.
insertRowShift (Insert Row Shift)	A Boolean that indicates whether cells in the sheet had to be inserted when the insert row was shown for this table. True if the cells were shifted, false otherwise. Note: This happens when there are values in cells immediately below the table when the table is created and the insert row is shown. In this case blank cells for the insert row are inserted, and the existing values in the sheet are shifted down by one row to make room. The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description
name (Name)	A string representing the name of the table that is used to reference the table programmatically through the spreadsheet applications object model. This string shall be unique per table per sheet. It has the same length and character restrictions as for displayName. By default this should be the same as the table's `displayName`. This name should also be kept in synch with the `displayName` when the `displayName` is updated in the UI by the spreadsheet user. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
published (Published)	A Boolean representing whether this table is marked as published for viewing by a server based spreadsheet application. True if it should be viewed by the server spreadsheet application, false otherwise. Note: Such an application might only display objects from the workbook that are marked as published, thus being able to load and calculate the entire workbook but only show the specific items that are marked as published. This can allow the server spreadsheet rendering to provide a more restricted view of the workbook. The possible values for this attribute are defined by the XML Schema boolean datatype.
ref (Reference)	The range on the relevant sheet that the table occupies expressed using A1 style referencing. The reference shall include the totals row if it is shown. The possible values for this attribute are defined by the ST_Ref simple type (§3.18.64).
tableBorderDxfId (Table Border Format Id)	A zero based integer index into the differential formatting records <dxfs> in the styleSheet indicating what border formatting to apply to the borders of this table. The spreadsheet should fail to load if this index is out of bounds. The possible values for this attribute are defined by the ST_DxfId simple type (§3.18.26).
tableType (Table Type)	An optional enumeration specifying the type or source of the table. Indicates whether the table is based off of an external data query, data in a worksheet, or from an xml data mapped to a worksheet. The possible values for this attribute are defined by the ST_TableType simple type (§3.18.80).
totalsRowBorderDxfId (Totals Row Border Format Id)	A zero based integer index into the differential formatting records <dxfs> in the styleSheet indicating what border formatting to apply to the totals row of this table. The spreadsheet should fail to load if this index is out of bounds.

Attributes	Description
	The possible values for this attribute are defined by the ST_DxfId simple type (§3.18.26).
totalsRowCellStyle (Totals Row Style)	A string representing the name of the cell style that is applied to the totals row cells of the table. If this string is missing or invalid, then the totals row style specified by the current table style should be applied. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
totalsRowCount (Totals Row Count)	An integer representing the number of totals rows that shall be shown at the bottom of the table. 0 means that the totals row is not shown. It is up to the spreadsheet application to determine if numbers greater than 1 are allowed. Unless the spreadsheet application has a feature where they may ever be more than one totals row, this number should not be higher than 1. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
totalsRowDxfId (Totals Row Format Id)	A zero based integer index into the differential formatting records <dxfs> in the styleSheet indicating which format to apply to the totals row of this table. The spreadsheet should fail to load if this index is out of bounds. The possible values for this attribute are defined by the ST_DxfId simple type (§3.18.26).
totalsRowShown (Totals Row Shown)	A Boolean indicating whether the totals row has ever been shown in the past for this table. True if the totals row has been shown, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Table">
   <sequence>
      <element name="autoFilter" type="CT_AutoFilter" minOccurs="0" maxOccurs="1"/>
      <element name="sortState" type="CT_SortState" minOccurs="0" maxOccurs="1"/>
      <element name="tableColumns" type="CT_TableColumns" minOccurs="1" maxOccurs="1"/>
      <element name="tableStyleInfo" type="CT_TableStyleInfo" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="id" type="xsd:unsignedInt" use="required"/>
   <attribute name="name" type="ST_Xstring" use="optional"/>
   <attribute name="displayName" type="ST_Xstring" use="required"/>
   <attribute name="comment" type="ST_Xstring" use="optional"/>
   <attribute name="ref" type="ST_Ref" use="required"/>
   <attribute name="tableType" type="ST_TableType" use="optional" default="worksheet"/>
   <attribute name="headerRowCount" type="xsd:unsignedInt" use="optional" default="1"/>
   <attribute name="insertRow" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="insertRowShift" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="totalsRowCount" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="totalsRowShown" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="published" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="headerRowDxfId" type="ST_DxfId" use="optional"/>
   <attribute name="dataDxfId" type="ST_DxfId" use="optional"/>
   <attribute name="totalsRowDxfId" type="ST_DxfId" use="optional"/>
   <attribute name="headerRowBorderDxfId" type="ST_DxfId" use="optional"/>
   <attribute name="tableBorderDxfId" type="ST_DxfId" use="optional"/>
   <attribute name="totalsRowBorderDxfId" type="ST_DxfId" use="optional"/>
   <attribute name="headerRowCellStyle" type="ST_Xstring" use="optional"/>
   <attribute name="dataCellStyle" type="ST_Xstring" use="optional"/>
   <attribute name="totalsRowCellStyle" type="ST_Xstring" use="optional"/>
   <attribute name="connectionId" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.5.1.3 tableColumn (Table Column)

An element representing a single column for this table.

Parent Elements
tableColumns (§3.5.1.4)

Child Elements	Subclause
calculatedColumnFormula (Calculated Column Formula)	§3.5.1.1
extLst (Future Feature Data Storage Area)	§3.2.10
totalsRowFormula (Totals Row Formula)	§3.5.1.6
xmlColumnPr (XML Column Properties)	§3.5.1.7

Attributes	Description

Attributes	Description
dataCellStyle (Data Area Style Name)	A string representing the name of the cell style that is applied to the cells in the data area of this table column. If this string is missing or invalid, then the data cell style specified by the current table style should be applied. This cell style should get precedence over the `dataCellStyle` defined by the `table`. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
dataDxfId (Data & Insert Row Format Id)	A zero based integer index into the differential formatting records <dxfs> in the styleSheet indicating which format to apply to the data area of this column. This formatting shall also apply to cells on the insert row for this column. The spreadsheet should fail to load if this index is out of bounds. The possible values for this attribute are defined by the ST_DxfId simple type (§3.18.26).
headerRowCellStyle (Header Row Cell Style)	A string representing the name of the cell style that is applied to the header row cell of this column. If this string is missing or invalid, then header row style specified by the current table style should be applied. This cell style should get precedence over the `headerRowCellStyle` defined by the `table`. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
headerRowDxfId (Header Row Cell Format Id)	A zero based integer index into the differential formatting records <dxfs> in the styleSheet indicating which format to apply to the header cell of this column. The possible values for this attribute are defined by the ST_DxfId simple type (§3.18.26).
id (Table Field Id)	An integer representing the unique identifier of this column. This shall be unique per table. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
name (Column name)	A string representing the unique caption of the table column. This is what shall be displayed in the header row in the UI, and is referenced through functions. This name shall be unique per table. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
queryTableFieldId	An integer representing the query table field ID corresponding to this table column.

Attributes	Description
(Query Table Field Id)	The relationship between this table and the corresponding query table is expressed in _rels part for this table. Each queryTableField has a unique `id` attribute, and this `id` is what is referenced here. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
totalsRowCellStyle (Totals Row Style Name)	A string representing the name of the cell style that is applied to the Totals Row cell of this column. If this string is missing or invalid, then the totals row cell style specified by the current table style should be applied. This cell style should get precedence over the `totalsRowCellStyle` defined by the `table`. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
totalsRowDxfId (Totals Row Format Id)	A zero based integer index into the differential formatting records <dxfs> in the styleSheet indicating which format to apply to the totals row cell of this column. The spreadsheet shall not load if this index is out of bounds. The possible values for this attribute are defined by the ST_DxfId simple type (§3.18.26).
totalsRowFunction (Totals Row Function)	An enumeration indicating which type of aggregation to show in the totals row cell for this column. The possible values for this attribute are defined by the ST_TotalsRowFunction simple type (§3.18.83).
totalsRowLabel (Totals Row Label)	A String to show in the totals row cell for this column. This string shall be ignored unless the `totalsRowFunction`="none" for this column, in which case it is displayed in the totals row. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
uniqueName (Unique Name)	An optional string representing the unique name of the table column. This string is used to bind the column to a field in an data table, so it shall should only be used when this table's `tableType` is `queryTable` or `xml`. This name shall be unique per table when it is used. For tables created from xml mappings, by default this should be the same as the name of the column, and should be kept in synch with the name of the column if that name is

Attributes	Description
	altered by the spreadsheet application. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TableColumn">
   <sequence>
      <element name="calculatedColumnFormula" type="CT_TableFormula" minOccurs="0" maxOccurs="1"/>
      <element name="totalsRowFormula" type="CT_TableFormula" minOccurs="0" maxOccurs="1"/>
      <element name="xmlColumnPr" type="CT_XmlColumnPr" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="id" type="xsd:unsignedInt" use="required"/>
   <attribute name="uniqueName" type="ST_Xstring" use="optional"/>
   <attribute name="name" type="ST_Xstring" use="required"/>
   <attribute name="totalsRowFunction" type="ST_TotalsRowFunction" use="optional" default="none"/>
   <attribute name="totalsRowLabel" type="ST_Xstring" use="optional"/>
   <attribute name="queryTableFieldId" type="xsd:unsignedInt" use="optional"/>
   <attribute name="headerRowDxfId" type="ST_DxfId" use="optional"/>
   <attribute name="dataDxfId" type="ST_DxfId" use="optional"/>
   <attribute name="totalsRowDxfId" type="ST_DxfId" use="optional"/>
   <attribute name="headerRowCellStyle" type="ST_Xstring" use="optional"/>
   <attribute name="dataCellStyle" type="ST_Xstring" use="optional"/>
   <attribute name="totalsRowCellStyle" type="ST_Xstring" use="optional"/>
</complexType>
```

3.5.1.4 tableColumns (Table Columns)

An element representing the collection of all table columns for this table.

Parent Elements
table (§3.5.1.2)

Child Elements	Subclause
tableColumn (Table Column)	§3.5.1.3

Attributes	Description
count (Column Count)	An integer representing the total count of how many columns there are in this Table. This count shall include both query-defined and user-defined columns. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TableColumns">
   <sequence>
      <element name="tableColumn" type="CT_TableColumn" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.5.1.5 tableStyleInfo (Table Style)

This element describes which style is used to display this table, and specifies which portions of the table have the style applied.

Styles define set of formatting properties that may be easily referenced by cells or other objects in the spreadsheet application. A style can be applied to a table, but tables can define specific parts of the table that should not have the style applied independently of other table parts. For instance a table may not apply the row striping of the style, and may not show the style's formatting of the last column, but will apply the column striping and the formatting to the first column.

Parent Elements
table (§3.5.1.2)

Attributes	Description
name (Style Name)	A string representing the name of the table style to use with this table. If the style name isn't valid then the spreadsheet application should use default style. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
showColumnStripes (Show Column Stripes)	A Boolean indicating whether column stripe formatting is applied. True when style column stripe formatting is applied, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.
showFirstColumn (Show First Column)	A Boolean indicating whether the first column in the table should have the style applied. True if the first column has the style applied, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.
showLastColumn (Show Last Column)	A Boolean indicating whether the last column in the table should have the style applied. True if the last column has the style applied, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.
showRowStripes (Show Row Stripes)	A Boolean indicating whether row stripe formatting is applied. True when style row stripe formatting is applied, false otherwise.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TableStyleInfo">
    <attribute name="name" type="ST_Xstring" use="optional"/>
    <attribute name="showFirstColumn" type="xsd:boolean" use="optional"/>
    <attribute name="showLastColumn" type="xsd:boolean" use="optional"/>
    <attribute name="showRowStripes" type="xsd:boolean" use="optional"/>
    <attribute name="showColumnStripes" type="xsd:boolean" use="optional"/>
</complexType>
```

3.5.1.6 totalsRowFormula (Totals Row Formula)

This element contains a custom formula for aggregating values from the column.

Each tableColumn has a `totalsRowFunction` that can be used for simple aggregations such as average, standard deviation, min, max, count, and others. If a more custom calculation is desired, then this element should be used, and the `totalsRowFunction` shall be set to "`custom`".

Parent Elements
tableColumn (§3.5.1.3)

Attributes	Description
array (Array)	A Boolean value that indicates whether this formula is an array style formula. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TableFormula">
    <simpleContent>
        <extension base="ST_Formula">
            <attribute name="array" type="xsd:boolean" default="false"/>
        </extension>
    </simpleContent>
</complexType>
```

3.5.1.7 xmlColumnPr (XML Column Properties)

An element defining the XML column properties for a column. This is only used for tables created from XML mappings.

[*Example*: Here is a simple example showing a table column that has an xmlColumPr.

```
<tableColumn id="1" uniqueName="SomeElement" name="SomeElement">
  <xmlColumnPr mapId="1" xpath="/xml/foo/element" xmlDataType="string"/>
</tableColumn>
```

end example]

Parent Elements
tableColumn (§3.5.1.3)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10

Attributes	Description

denormalized (Denormalized)	A Boolean that indicates whether the contents of the column have been filled down due to flattening. `True` if it has been filled down (denormilized), `false` otherwise.
	
	This should be used when an XML mapping parent value has many children, and both the parent and child fields are mapped to their own column in the table.
	
	[*Example*:
	`<Order ID="3">`
	`<Item>Milk</Item>`
	`<Item>Bread</Item>`
	`<Item>Cheese</Item>`
	`</Order>`
	
	The resulting table in the spreadsheet application would have two columns, the first with the item ID, filled down for each item in the table as follows:
	
	Item ID Item
	3 Milk
	3 Bread
	3 Cheese
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

mapId (XML Map Id)	An integer representing the ID of the XML map this table field is associated with. The XML map will be defined in the xml maps part, and the Map element should have the corresponding `id`. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
xmlDataType (XML Data Type)	An enumeration indicating which XML data type is used by this column. The possible values for this attribute are defined by the ST_XmlDataType simple type

Attributes	Description
	(§3.18.95).
xpath (XPath)	A string representing the XML path to the element this column is associated with. The spreadsheet application should support XPath limited to the following • The XPath is an absolute path to a simple-content element or attribute [*Example*: "/ns1:root/ns1:row/ns1:column1" is supported if 'column1' is a child-most node, but not "/ns1:root/ns1:row" for the same document since 'row' is not a child. *end example*] • The XPath does not express axes, but uses the default child axes [*Example*: "/ns1:root/ns1:row" is supported but not "/ns1:root/child::ns1:row *end example*] • An optional filter can be expressed at the end of the xpath [*Example*: "/ns1:root/ns1:row/ns1:column1[@foo='abc']" is supported but not "/ns1:root/ns1:row[@foo='abc']/ns1:column1" *end example*] • The filter can only contain a single expression comparing a named attribute to a specific value • Filters are only supported on XPaths that resolve to a simple-content element (not attributes) • The named attribute must be defined as an attribute of the simple-content element • The attribute name must be preceded by the short-hand (@) symbol representing the axes 'attribute' [*Example*: "/ns1:root/ns1:row/ns1:column1[@foo='abc']" is supported not "/ns1:root/ns1:row/ns1:column1[attribute::foo='abc']" *end example*] • An arbitrary amount of white-space can be embedded between filter tokens [*Example*: "/ns1:root/ns1:row/ns1:column1[@ foo='abc']" is valid

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_XmlColumnPr">
   <sequence>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="mapId" type="xsd:unsignedInt" use="required"/>
   <attribute name="xpath" type="ST_Xstring" use="required"/>
   <attribute name="denormalized" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="xmlDataType" type="ST_XmlDataType" use="required"/>
</complexType>
```

3.5.2 Single Cell Tables

A single cell table is generated from an XML mapping. These really just look like regular cells to the spreadsheet user, but shall be implemented as Tables "under the covers."

These tables don't have the full set of properties that multi cell tables do. They only have the various XML properties, and core table properties (such as `id` and `name`) that are needed to create a table and XML mapping. For instance the formatting properties, totals row, and headers row don't exist for the single cell XML tables. The formatting for these cells is maintained in the style sheet.

3.5.2.1 singleXmlCell (Table Properties)

This element represents the table properties for a single cell XML table.

Parent Elements
singleXmlCells (§3.5.2.2)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
xmlCellPr (Cell Properties)	§3.5.2.3

Attributes	Description
connectionId (Connection ID)	An integer representing an ID to indicate which connection from the connections collection is used by this table. This is only used for tables that are based off of xml maps

Attributes	Description
	The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
id (Table Id)	An integer representing the unique identifier of the table. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
r (Reference)	An A1 cell style reference to the cell that the single cell xml table occupies. The possible values for this attribute are defined by the ST_CellRef simple type (§3.18.8).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SingleXmlCell">
   <sequence>
      <element name="xmlCellPr" type="CT_XmlCellPr" minOccurs="1" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="id" type="xsd:unsignedInt" use="required"/>
   <attribute name="r" type="ST_CellRef" use="required"/>
   <attribute name="connectionId" type="xsd:unsignedInt" use="required"/>
</complexType>
```

3.5.2.2 singleXmlCells (Single Cells)

This element is a container for a collection of singleXmlCell tables.

Parent Elements
Root element of SpreadsheetML Single Cell Table Definitions part

Child Elements	Subclause
singleXmlCell (Table Properties)	§3.5.2.1

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SingleXmlCells">
   <sequence>
      <element name="singleXmlCell" type="CT_SingleXmlCell" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.5.2.3 xmlCellPr (Cell Properties)

This element stores the XML properties for the cell of a single cell xml table.

Parent Elements
singleXmlCell (§3.5.2.1)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
xmlPr (Column XML Properties)	§3.5.2.4

Attributes	Description
id (Table Field Id)	The unique identifier of the XML properties for the cell. This should always be set to the value of 1 since this id is always meant to be for a single cell xml table. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
uniqueName (Unique Table Name)	An optional string representing the unique name of the table column. By default this is the same as the name of the column. This should hold the name of the element or attribute that this cell is referring to in the XML. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_XmlCellPr">
   <sequence>
      <element name="xmlPr" type="CT_XmlPr" minOccurs="1" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="id" type="xsd:unsignedInt" use="required"/>
   <attribute name="uniqueName" type="ST_Xstring" use="optional"/>
</complexType>
```

3.5.2.4 xmlPr (Column XML Properties)

This element represents the column properties for single cell XML tables.

Parent Elements
xmlCellPr (§3.5.2.3)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10

Attributes	Description

Attributes	Description
mapId (XML Map Id)	An integer representing the ID of the XML map this table field is associated with. The XML map will be defined in the xml maps part, and the Map element should have the corresponding id. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
xmlDataType (XML Data Type)	An enumeration indicating which XML data type is used by this column. The possible values for this attribute are defined by the ST_XmlDataType simple type (§3.18.95).
xpath (XPath)	A string representing the XML path to the element this column is associated with. The spreadsheet application should support XPath limited to the following: • The XPath is an absolute path to a simple-content element or attribute [*Example*: "/ns1:root/ns1:row/ns1:column1" is supported if 'column1' is a child-most node, but not "/ns1:root/ns1:row" for the same document since 'row' is not a child. *end example*] • The XPath does not express axes, but uses the default child axes [*Example*: "/ns1:root/ns1:row" is supported but not "/ns1:root/child::ns1:row *end example*] • An optional filter can be expressed at the end of the xpath [*Example*: "/ns1:root/ns1:row/ns1:column1[@foo='abc']" is supported but not "/ns1:root/ns1:row[@foo='abc']/ns1:column1" *end example*] • The filter can only contain a single expression comparing a named attribute to a specific value • Filters are only supported on XPaths that resolve to a simple-content element (not attributes) • The named attribute must be defined as an attribute of the simple-content element • The attribute name must be preceded by the short-hand (@) symbol representing the axes 'attribute' [*Example*:

Attributes	Description
	"/ns1:root/ns1:row/ns1:column1[@foo='abc']" is supported not "/ns1:root/ns1:row/ns1:column1[attribute::foo='abc']" *end example*] • An arbitrary amount of white-space can be embedded between filter tokens [*Example*: "/ns1:root/ns1:row/ns1:column1[@ foo='abc']" is valid *end example*] The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_XmlPr">
   <sequence>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="mapId" type="xsd:unsignedInt" use="required"/>
   <attribute name="xpath" type="ST_Xstring" use="required"/>
   <attribute name="xmlDataType" type="ST_XmlDataType" use="required"/>
</complexType>
```

3.6 Calculation Chain

The cells in a workbook can be calculated in different orders depending on various optimizations and dependencies. The calculation chain specifies the order in which the cells in a workbook were last calculated.

The calculation chain only deals with cells that require calculation - i.e., it only deals with cells that contain formulas. It does not track or express dependencies amongst the formulas, but rather only records the order in which the cells were last calculated.

The calculation chain order may change over time. One obvious way this can happen is that new formulas can be added, formulas can be removed or updated. The spreadsheet application may also optionally implement partial calculation as an optimization. Partial calculation is when the spreadsheet only recalculates cells that have had their dependencies or values changed. This way, when a number in a cell is changed, requiring an update to a dependent formula, only the cells that are affected by the update will be recalculated, as opposed to recalculating the entire workbook.

The calculation chain described in this section is not required by the spreadsheet application, but can be used if the spreadsheet application finds it useful. It can be loaded by a spreadsheet application, or the application may optionally construct it at run time in memory based on formula dependencies. Since the xml data described in this section is not strictly required, the spreadsheet application is free to ignore the order in which the calculation chain specifies calculations - i.e., even if the calculation chain is loaded, the spreadsheet application is free to perform calculations in a different order at run time.

[*Example*:

Consider the following workbook (the formulas shown instead of cell values):

	A	B	C	D	E
1	1	=A1	=B1+A1	=C1+B1+A1	=D1+C1+B1+A1
2					
3					
4					
5	1	=A5	=B5+A5	=C5+B5+A5	=D5+C5+B5+A5
6					
7					

There is a constant entered in A1 and A5, and next to each of those cells are a series of cells which contain formulas that depend on those cells.

After entering the cells on the first row, and then the cells on the 5th row, the calc chain xml looks like this:

```
<calcChain xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/5/main">
  <c r="E5" i="1"/>
  <c r="D5"/>
  <c r="C5"/>
  <c r="B5"/>
  <c r="E1"/>
  <c r="D1"/>
  <c r="C1"/>
  <c r="B1"/>
</calcChain>
```

It is in this order because B1 was calced first (it was the first formula enterd in the workbook), followed by C1, D1, and so on. Then B5 was entered in the 5th row, followed by the other cells in the 5th row, ending with E5.

But, after a full recalculation, the spreadsheet application has realized that cells B5:E5 are on the same child chain, and cells B1:E1 are likewise on their own child chain. The xml now looks like this:

```
<calcChain xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/5/main">
  <c r="B1" i="1"/>
  <c r="C1" s="1"/>
  <c r="D1" s="1"/>
  <c r="E1" s="1"/>
  <c r="B5"/>
  <c r="C5" s="1"/>
  <c r="D5" s="1"/>
  <c r="E5" s="1"/>
</calcChain>
```

end example]

3.6.1 c (Cell)

This element represents a single cell, which shall contain a formula, in the calc chain. Cell's are calculated in the same order as c attribute is listed in the xml, starting from the top.

Parent Elements
calcChain (§3.6.2)

Attributes	Description
a (Array)	A Boolean flag indicating whether the cell's formula is an array formula. True if this cell's formula is an array entered formula, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.
i (Sheet Id)	A sheet Id of a sheet the cell belongs to. If this is omitted, it is assumed to be the same as the i value of the previous cell. The possible values for this attribute are defined by the XML Schema int datatype.
l (New Dependency Level)	A Boolean flag indicating that the cell's formula starts a new dependency level. True if the formula starts a new dependency level, false otherwise. Starting a new dependency level means that all concurrent calculations, and child calculations, must be completed - and the cells have new values - before the calc chain can continue. In other words, this dependency level may depend on levels that came before it, and any later dependency levels may depend on this level; but not later dependency levels can have any calculations started until this dependency level completes. The possible values for this attribute are defined by the XML Schema boolean datatype.
r (Cell Reference)	An A-1 style reference to a cell. The possible values for this attribute are defined by the ST_CellRef simple type (§3.18.8).
s (Child Chain)	A Boolean flag indicating whether the cell's formula is on a child chain. True if this cell is part of a child chain, false otherwise. If this is omitted, it is assumed to be the same as the s value of the previous cell . A child chain is a list of calculations that occur which depend on the parent to the chain. There shall not be cross dependencies between child chains. Child chains are not the same as dependency levels - a child chain and its parent are all on the same dependency level. Child chains are series of calculations that can be independently farmed out to other threads or processors. The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description
t (New Thread)	A Boolean flag indicating whether the cell's formula starts a new thread. True if the cell's formula starts a new thread, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CalcCell">
   <attribute name="r" type="ST_CellRef" use="required"/>
   <attribute name="i" type="xsd:int" use="optional" default="0"/>
   <attribute name="s" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="l" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="t" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="a" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.6.2 calcChain (Calculation Chain Info)

This element represents the root of the calculation chain.

Parent Elements
Root element of SpreadsheetML Calculation Chain part

Child Elements	Subclause
c (Cell)	§3.6.1
extLst (Future Feature Data Storage Area)	§3.2.10

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CalcChain">
   <sequence>
      <element name="c" type="CT_CalcCell" minOccurs="1" maxOccurs="unbounded"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
</complexType>
```

3.7 Comments

A comment is a rich text note that is attached to & associated with a cell, separate from other cell content. Comment content is stored separate from the cell, and is displayed in a drawing object (like a text box) that is separate from, but associated with, a cell. Comments are used as reminders, such as noting how a complex formula works, or to provide feedback to other users. Comments can also be used to explain assumptions made in a formula or to call out something special about the cell.

[*Example*:

```
<comments>
  <authors>
    <author>Bob</author>
    <author>CBR</author>
  </authors>
  <commentList>
    <comment ref="D4" authorId="0">
      <text>
        <r>
          <rPr>
            <b/>
            <sz val="8"/>
            <color indexed="81"/>
            <rFont val="Calibri"/>
            <charset val="1"/>
            <scheme val="minor"/>
          </rPr>
          <t>Bob:</t>
        </r>
        <r>
          <rPr>
            <sz val="8"/>
           <color indexed="81"/>
           <rFont val="Calibri"/>
           <charset val="1"/>
           <scheme val="minor"/>
         </rPr>
         <t xml:space="preserve">Why such high expense?</t>
        </r>
     </text>
    </comment>
  </commentList>
</comments>
```

end example]

This xml sample displays a comment by "Bob" (bolded) that says, "Why such high expense?" (non bolded).

3.7.1 author (Author)

This element holds a string representing the name of a single author of comments. Every comment must have an author. The maximum length of the author string is an implementation detail, but a good guideline is 255 chars.

The possible values for this element are defined by the ST_Xstring simple type (§3.18.96).

Parent Elements
authors (§3.7.2)

3.7.2 authors (Authors)

This element is a container that holds a list of comment author names. There may be many comment authors per sheet, but each author name must be unique per sheet. The information for each author is stored only once for that sheet, and comments refer to the author by zero based index.

Note that there can be multiple lists of authors per workbook since each sheet contains its own comments part, and each comments part defines a list of authors for comments on that sheet.

Parent Elements
comments (§3.7.5)

Child Elements	Subclause
author (Author)	§3.7.1

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Authors">
   <sequence>
      <element name="author" type="ST_Xstring" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.7.3 comment (Comment)

This element represents a single user entered comment.. Each comment shall have an author and can optionally contain richly formatted text.

Parent Elements
commentList (§3.7.4)

Child Elements	Subclause
text (Comment Text)	§3.7.6

Attributes	Description
authorId (Author Id)	Required. An unsigned integer which is used as the zero based index into the list of authors for this set of comments. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

Attributes	Description
guid (Unique Identifier for Comment)	Unique identifier for this comment. The attribute is required and shall be unique across all comments in shared workbooks. The possible values for this attribute are defined by the ST_Guid simple type (§3.18.41).
ref (Cell Reference)	Required. A string that serves as the A1 style reference to the cell that the comment is associated with. May only reference a single cell, not a range of cells, since comments are on a per cell basis. The possible values for this attribute are defined by the ST_Ref simple type (§3.18.64).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Comment">
   <sequence>
      <element name="text" type="CT_Rst" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="ref" type="ST_Ref" use="required"/>
   <attribute name="authorId" type="xsd:unsignedInt" use="required"/>
   <attribute name="guid" type="ST_Guid" use="optional"/>
</complexType>
```

3.7.4 commentList (List of Comments)

This element is a container that holds a list of comments for the sheet.

Parent Elements
comments (§3.7.5)

Child Elements	Subclause
comment (Comment)	§3.7.3

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CommentList">
   <sequence>
      <element name="comment" type="CT_Comment" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.7.5 comments (Comments)

This element is the root container of a set of comments and comment authors for a particular sheet. Each set of comments for a sheet is stored in a separate xml part. The relationship part for a sheet defines a link to the correct comment part for that sheet.

Parent Elements

Parent Elements
Root element of SpreadsheetML Comments part

Child Elements	Subclause
authors (Authors)	§3.7.2
commentList (List of Comments)	§3.7.4
extLst (Future Feature Data Storage Area)	§3.2.10

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Comments">
    <sequence>
        <element name="authors" type="CT_Authors" minOccurs="1" maxOccurs="1"/>
        <element name="commentList" type="CT_CommentList" minOccurs="1" maxOccurs="1"/>
        <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
    </sequence>
</complexType>
```

3.7.6 text (Comment Text)

This element contains rich text which represents the text of a comment. The maximum length for this text is a spreadsheet application implementation detail. A recommended guideline is 32767 chars.

Parent Elements
comment (§3.7.3)

Child Elements	Subclause
phoneticPr (Phonetic Properties)	§3.4.3
r (Rich Text Run)	§3.4.4
rPh (Phonetic Run)	§3.4.6
t (Text)	§3.4.12

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Rst">
    <sequence>
        <element name="t" type="ST_Xstring" minOccurs="0" maxOccurs="1"/>
        <element name="r" type="CT_RElt" minOccurs="0" maxOccurs="unbounded"/>
        <element name="rPh" type="CT_PhoneticRun" minOccurs="0" maxOccurs="unbounded"/>
        <element name="phoneticPr" minOccurs="0" maxOccurs="1" type="CT_PhoneticPr"/>
    </sequence>
</complexType>
```

3.8 Styles

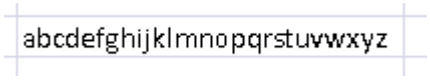
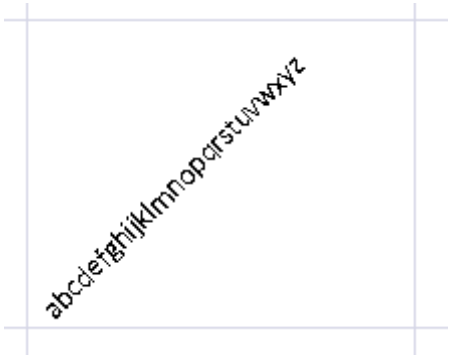
This subclause specifies the possible formatting information for the contents of the cells on a sheet in a SpreadsheetML document.

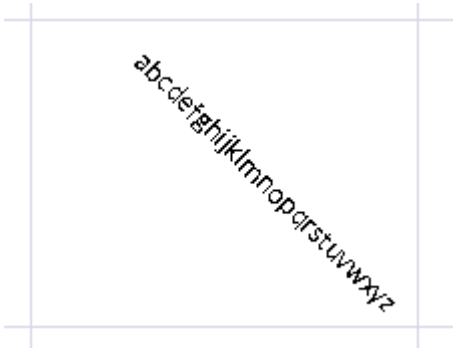
3.8.1 alignment (Alignment)

Formatting information pertaining to text alignment in cells. There are a variety of choices for how text is aligned both horizontally and vertically, as well as indentation settings, and so on.

Parent Elements
dxf (§3.8.14); ndxf (§3.11.1.4); odxf (§3.11.1.6); xf (§3.8.45)

Attributes	Description
horizontal (Horizontal Alignment)	Specifies the type of horizontal alignment in cells. The possible values for this attribute are defined by the ST_HorizontalAlignment simple type (§3.18.42).
indent (Indent)	An integer value, where an increment of 1 represents 3 spaces. Indicates the number of spaces (of the normal style font) of indentation for text in a cell. The number of spaces to indent is calculated as following: Number of spaces to indent = indent value * 3 [*Example*: For example, an indent value of '1' means that the text begins 3 space widths (of the normal style font) from the edge of the cell. *end example*] [*Note*: The width of one space character is defined by the font. *end note*] Only `left`, `right`, and `distributed` horizontal alignments are supported. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
justifyLastLine (Justify Last Line)	A boolean value indicating if the cells justified or distributed alignment should be used on the last line of text. (This is typical for East Asian alignments but not typical in other contexts.) The possible values for this attribute are defined by the XML Schema boolean datatype.
readingOrder	An integer value indicating whether the reading order (bidirectionality) of the cell is left-

Attributes	Description
(Reading Order)	to-right, right-to-left, or context dependent. 0 - Context Dependent 1 - Left-to-Right 2 - Right-to-Left The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
relativeIndent (Relative Indent)	An integer value (used only in a dxf element) to indicate the additional number of spaces of indentation to adjust for text in a cell. The possible values for this attribute are defined by the XML Schema int datatype.
shrinkToFit (Shrink To Fit)	A boolean value indicating if the displayed text in the cell should be shrunk to fit the cell width. Not applicable when a cell contains multiple lines of text. The possible values for this attribute are defined by the XML Schema boolean datatype.
textRotation (Text Rotation)	Text rotation in cells. Expressed in degrees. Values range from 0 to 180. The first letter of the text is considered the center-point of the arc. For 0 - 90, the value represents degrees above horizon. For 91-180 the degrees below the horizon is calculated as: [degrees below horizon] = 90 - textRotation. 0  45  90

Attributes	Description
	 135  180  The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
vertical (Vertical Alignment)	Vertical alignment in cells. The possible values for this attribute are defined by the ST_VerticalAlignment simple

Attributes	Description
	type (§3.18.89).
wrapText (Wrap Text)	A boolean value indicating if the text in a cell should be line-wrapped within the cell. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CellAlignment">
   <attribute name="horizontal" type="ST_HorizontalAlignment" use="optional"/>
   <attribute name="vertical" type="ST_VerticalAlignment" use="optional"/>
   <attribute name="textRotation" type="xsd:unsignedInt" use="optional"/>
   <attribute name="wrapText" type="xsd:boolean" use="optional"/>
   <attribute name="indent" type="xsd:unsignedInt" use="optional"/>
   <attribute name="relativeIndent" type="xsd:int" use="optional"/>
   <attribute name="justifyLastLine" type="xsd:boolean" use="optional"/>
   <attribute name="shrinkToFit" type="xsd:boolean" use="optional"/>
   <attribute name="readingOrder" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.8.2 b (Bold)

Displays characters in bold face font style.

Parent Elements
font (§3.8.21); rPr (§3.4.7)

Attributes	Description
val (Value)	A boolean value for the property specified by the parent XML element. If omitted, the default value is `true`. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BooleanProperty">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

3.8.3 bgColor (Background Color)

Background color of the cell fill pattern. Cell fill patterns operate with two colors: a background color and a foreground color. These combine together to make a patterned cell fill.

Parent Elements
patternFill (§3.8.32)

Attributes	Description
auto (Automatic)	A boolean value indicating the color is automatic and system color dependent. The possible values for this attribute are defined by the XML Schema boolean datatype.
indexed (Index)	Indexed color value. Only used for backwards compatibility. References a color in indexedColors. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
rgb (Alpha Red Green Blue Color Value)	Standard Alpha Red Green Blue color value (ARGB). The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§3.18.86).
theme (Theme Color)	Index into the <clrScheme> collection, referencing a particular <sysClr> or <srgbClr> value expressed in the Theme part. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
tint (Tint)	Specifies the tint value applied to the color. If tint is supplied, then it is applied to the RGB value of the color to determine the final color applied. The tint value is stored as a double from -1.0 .. 1.0, where -1.0 means 100% darken and 1.0 means 100% lighten. Also, 0.0 means no change. In loading the RGB value, it is converted to HLS where HLS values are (0..HLSMAX), where HLSMAX is currently 255. [*Example*: Here are some examples of how to apply tint to color: **If (tint < 0)** Lum' = Lum * (1.0 + tint) For example: Lum = 200; tint = -0.5; Darken 50% Lum‘ = 200 * (0.5) => 100 For example: Lum = 200; tint = -1.0; Darken 100% (make black) Lum‘ = 200 * (1.0-1.0) => 0 **If (tint > 0)** Lum‘ = Lum * (1.0-tint) + (HLSMAX – HLSMAX * (1.0-tint)) For example: Lum = 100; tint = 0.75; Lighten 75%

Attributes	Description
	Lum‘ = 100 * (1-.75) + (HLSMAX – HLSMAX*(1-.75)) = 100 * .25 + (255 – 255 * .25) = 25 + (255 – 63) = 25 + 192 = 217 For example: Lum = 100; tint = 1.0; Lighten 100% (make white) Lum‘ = 100 * (1-1) + (HLSMAX – HLSMAX*(1-1)) = 100 * 0 + (255 – 255 * 0) = 0 + (255 – 0) = 255 *end example*] The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <attribute name="auto" type="xsd:boolean" use="optional"/>
   <attribute name="indexed" type="xsd:unsignedInt" use="optional"/>
   <attribute name="rgb" type="ST_UnsignedIntHex" use="optional"/>
   <attribute name="theme" type="xsd:unsignedInt" use="optional"/>
   <attribute name="tint" type="xsd:double" use="optional" default="0.0"/>
</complexType>
```

3.8.4 border (Border)

Expresses a single set of cell border formats (left, right, top, bottom, diagonal). Color is optional. When missing, 'automatic' is implied.

Parent Elements
borders (§3.8.5); dxf (§3.8.14); ndxf (§3.11.1.4); odxf (§3.11.1.6)

Child Elements	Subclause
bottom (Bottom Border)	§3.8.6
diagonal (Diagonal)	§3.8.13
horizontal (Horizontal Inner Borders)	§3.8.24
left (Left Border)	§3.8.27
right (Right Border)	§3.8.35
top (Top Border)	§3.8.43
vertical (Vertical Inner Border)	§3.8.44

Attributes	Description
diagonalDown	A boolean value indicating if the cell's diagonal border includes a diagonal line, starting at

Attributes	Description
(Diagonal Down)	the top left corner of the cell and moving down to the bottom right corner of the cell. [*Example*: This example shows a thin diagonal down line:  *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
diagonalUp (Diagonal Up)	A boolean value indicating if the cell's diagonal border includes a diagonal line, starting at the bottom left corner of the cell and moving up to the top right corner of the cell. [*Example*: This example shows a thin diagonal up line:  *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
outline (Outline)	A boolean value indicating if left, right, top, and bottom borders should be applied only to outside borders of a cell range. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <sequence>
      <element name="left" type="CT_BorderPr" minOccurs="0" maxOccurs="1"/>
      <element name="right" type="CT_BorderPr" minOccurs="0" maxOccurs="1"/>
      <element name="top" type="CT_BorderPr" minOccurs="0" maxOccurs="1"/>
      <element name="bottom" type="CT_BorderPr" minOccurs="0" maxOccurs="1"/>
      <element name="diagonal" type="CT_BorderPr" minOccurs="0" maxOccurs="1"/>
      <element name="vertical" type="CT_BorderPr" minOccurs="0" maxOccurs="1"/>
      <element name="horizontal" type="CT_BorderPr" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="diagonalUp" type="xsd:boolean" use="optional"/>
   <attribute name="diagonalDown" type="xsd:boolean" use="optional"/>
   <attribute name="outline" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

3.8.5 borders (Borders)

This element contains borders formatting information, specifying all border definitions for all cells in the workbook.

[*Example*: In this example the first border definition specifies that there are no borders, the second definition specifies that a there is a thin bottom border and medium right border, and the third definition specifies that there is a double top border.

```
<borders count="3">
  <border>
    <left/>
    <right/>
    <top/>
    <bottom/>
    <diagonal/>
  </border>
  <border>
    <left/>
    <right style="medium">
      <color indexed="64"/>
    </right>
    <top/>
    <bottom style="thin">
      <color indexed="64"/>
    </bottom>
    <diagonal/>
  </border>
```

```
    <border>
      <left/>
      <right/>
      <top style="double">
        <color auto="1"/>
      </top>
      <bottom/>
      <diagonal/>
    </border>
  </borders>
```

end example]

Parent Elements
styleSheet (§3.8.39)

Child Elements	Subclause
border (Border)	§3.8.4

Attributes	Description
count (Border Count)	Count of border elements. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Borders">
   <sequence>
      <element name="border" type="CT_Border" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.8.6 bottom (Bottom Border)

This element specifies the color and line style for the bottom border of a cell.

Parent Elements
border (§3.8.4)

Child Elements	Subclause
color (Data Bar Color)	§3.3.1.14

Attributes	Description
style (Line Style)	The line style for this border. The possible values for this attribute are defined by the ST_BorderStyle simple type (§3.18.3).

The following XML Schema fragment defines the contents of this element:

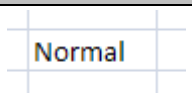
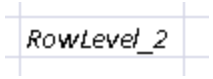
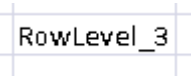
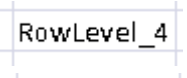
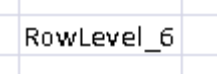
```
<complexType name="CT_BorderPr">
   <sequence>
      <element name="color" type="CT_Color" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="style" type="ST_BorderStyle" use="optional" default="none"/>
</complexType>
```

3.8.7 cellStyle (Cell Style)

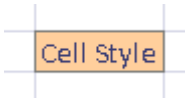
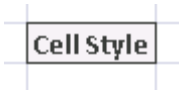
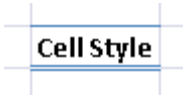
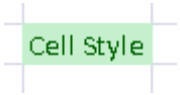
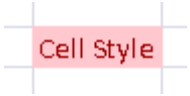
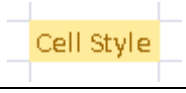
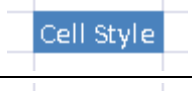
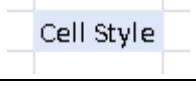
This element expresses the name and related formatting records for a named cell style in this workbook.

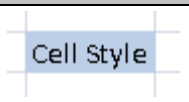
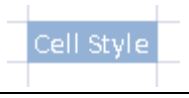
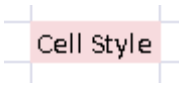
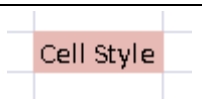
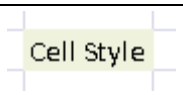
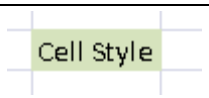
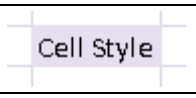
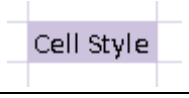
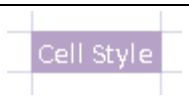
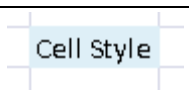
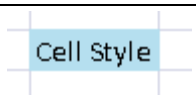
The built-in cell styles are written here by name, but the corresponding formatting records are assumed rather than explicitly written. Following is a listing of each of the built-in cell style definitions, whose normative definition is in Annex D. Also following is a table mapping the builtinId value to the cell style name.

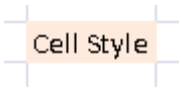
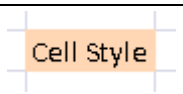
For all built-in cell styles, the builtinId determines the style, not the name. For all cell styles, Normal is applied by default.

builtinId	Cell Style Name	[*Example*: (informative)
0	Normal	
1	RowLevel_ + level #	Depends on level: 1 =  2 =  3 =  4 =  5 =  6 = 

builtinId	Cell Style Name	[*Example*: (informative)
		7 = RowLevel_7
2	ColLevel_ + level #	Depends on level: 1 = **ColLevel_1** 2 = *ColLevel_2* 3 = ColLevel_3 4 = ColLevel_4 5 = ColLevel_5 6 = ColLevel_6 7 = ColLevel_7
3	Comma	1,234.00
4	Currency	$1,234.00
5	Percent	123400%
6	Comma [0]	1,234
7	Currency [0]	$ 1,234
8	Hyperlink	hyperlink
9	Followed Hyperlink	followed hyperlink
10	Note	Cell Style
11	Warning Text	Cell Style

builtinId	Cell Style Name	[*Example*: (informative)
15	Title	
16	Heading 1	
17	Heading 2	
18	Heading 3	
19	Heading 4	
20	Input	
21	Output	
22	Calculation	
23	Check Cell	
24	Linked Cell	
25	Total	
26	Good	
27	Bad	
28	Neutral	
29	Accent1	
30	20% - Accent1	

builtinId	Cell Style Name	[*Example*: (informative)
31	40% - Accent1	
32	60% - Accent1	
33	Accent2	
34	20% - Accent2	
35	40% - Accent2	
36	60% - Accent2	
37	Accent3	
38	20% - Accent3	
39	40% - Accent3	
40	60% - Accent3	
41	Accent4	
42	20% - Accent4	
43	40% - Accent4	
44	60% - Accent4	
45	Accent5	
46	20% - Accent5	
47	40% - Accent5	

builtinId	Cell Style Name	[*Example*: (informative)
48	60% - Accent5	
49	Accent6	
50	20% - Accent6	
51	40% - Accent6	
52	60% - Accent6	
53	Explanatory Text	

Parent Elements
cellStyles (§3.8.8)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10

Attributes	Description
builtinId (Built-In Style Id)	The index of a built-in cell style: The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
customBuiltin (Custom Built In)	True indicates that this built-in cell style has been customized. By default built-in styles are not persisted when not in use. This flag indicates that a built-in style has been modified, and therefore should be saved with the workbook, even if not currently in use. The possible values for this attribute are defined by the XML Schema boolean datatype.
hidden (Hidden Style)	If `'true'` do not show this style in the application UI. The possible values for this attribute are defined by the XML Schema boolean datatype.
iLevel (Outline Style)	Indicates that this formatting is for an outline style . When styles are applied to outline levels (using the outline feature), this value is set and the formatting specified on this cell

Attributes	Description
	style is applied to the corresponding level of the outline. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
name (User Defined Cell Style)	The name of the cell style. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
xfId (Format Id)	Zero-based index referencing an xf record in the cellStyleXfs collection. This is used to determine the formatting defined for this named cell style. The possible values for this attribute are defined by the ST_CellStyleXfId simple type (§3.18.11).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CellStyle">
   <sequence>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="name" type="ST_Xstring" use="optional"/>
   <attribute name="xfId" type="ST_CellStyleXfId" use="required"/>
   <attribute name="builtinId" type="xsd:unsignedInt" use="optional"/>
   <attribute name="iLevel" type="xsd:unsignedInt" use="optional"/>
   <attribute name="hidden" type="xsd:boolean" use="optional"/>
   <attribute name="customBuiltin" type="xsd:boolean" use="optional"/>
</complexType>
```

3.8.8 cellStyles (Cell Styles)

This element contains the named cell styles, consisting of a sequence of named style records. A named cell style is a collection of direct or themed formatting (e.g., cell border, cell fill, and font type/size/style) grouped together into a single named style, and can be applied to a cell.

[*Example*: For example, "Normal", "Heading 1", "Title", and "20% Accent1" are named cell styles expressed below. They have builtInId's associated with them, and use xfId to reference the specific formatting elements pertaining to the particular style. The xfId is a zero-based index, referencing an xf record in the cellStyleXfs collection.

```
<cellStyles count="4">
  <cellStyle name="20% - Accent1" xfId="3" builtinId="30"/>
  <cellStyle name="Heading 1" xfId="2" builtinId="16"/>
  <cellStyle name="Normal" xfId="0" builtinId="0"/>
  <cellStyle name="Title" xfId="1" builtinId="15"/>
</cellStyles>
```

end example]

Parent Elements
styleSheet (§3.8.39)

Child Elements	Subclause
cellStyle (Cell Style)	§3.8.7

Attributes	Description
count (Style Count)	Count of style elements. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CellStyles">
   <sequence>
      <element name="cellStyle" type="CT_CellStyle" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.8.9 cellStyleXfs (Formatting Records)

This element contains the master formatting records (xf's) which define the formatting for all named cell styles in this workbook. Master formatting records reference individual elements of formatting (e.g., number format, font definitions, cell fills, etc) by specifying a zero-based index into those collections. Master formatting records also specify whether to apply or ignore particular aspects of formatting, for example whether to apply a border or not.

A cell can have both direct formatting (e.g., bold) and a cell style (e.g., Explanatory) applied to it. Therefore, both the cell style xf records and cell xf records must be read to understand the full set of formatting applied to a cell.

[*Example*: This example shows 4 master formatting records, each defining formatting for a named cell style (expressed in the cellStyles collection). Note that 0th record does not express any "apply" attributes, while the other records do express "apply" attribute values. For example, the last record specifies that number format, alignment, and protection formatting will not be applied to the cell, even when that information is specified in related formatting records.

```
<cellStyleXfs count="4">
  <xf numFmtId="0" fontId="0" fillId="0" borderId="0"/>
  <xf numFmtId="0" fontId="2" fillId="0" borderId="0" applyNumberFormat="0"
    applyFill="0" applyBorder="0" applyAlignment="0" applyProtection="0"/>
  <xf numFmtId="0" fontId="3" fillId="0" borderId="1" applyNumberFormat="0"
    applyFill="0" applyAlignment="0" applyProtection="0"/>
```

```
    <xf numFmtId="0" fontId="4" fillId="2" borderId="2" applyNumberFormat="0"
      applyAlignment="0" applyProtection="0"/>
  </cellStyleXfs>
```

end example]

Parent Elements
styleSheet (§3.8.39)

Child Elements	Subclause
xf (Format)	§3.8.45

Attributes	Description
count (Style Count)	Count of cell style (xf) elements. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CellStyleXfs">
   <sequence>
      <element name="xf" type="CT_Xf" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.8.10 cellXfs (Cell Formats)

This element contains the master formatting records (xf) which define the formatting applied to cells in this workbook. These records are the starting point for determining the formatting for a cell. Cells in the Sheet Part reference the xf records by zero-based index.

A cell can have both direct formatting (e.g., bold) and a cell style (e.g., Explanatory) applied to it. Therefore, both the cell style xf records and cell xf records must be read to understand the full set of formatting applied to a cell.

Parent Elements
styleSheet (§3.8.39)

Child Elements	Subclause
xf (Format)	§3.8.45

Attributes	Description
count (Format Count)	Count of xf elements. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CellXfs">
   <sequence>
      <element name="xf" type="CT_Xf" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.8.11 colors (Colors)

Color information associated with this stylesheet. This collection is written whenever the legacy color palette has been modified (backwards compatibility settings) or a custom color has been selected while using this workbook.

When the color palette is modified, the indexedColors collection is written. When a custom color has been selected, the mruColors collection is written.

Parent Elements
styleSheet (§3.8.39)

Child Elements	Subclause
indexedColors (Color Indexes)	§3.8.26
mruColors (MRU Colors)	§3.8.28

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Colors">
   <sequence>
      <element name="indexedColors" type="CT_IndexedColors" minOccurs="0" maxOccurs="1"/>
      <element name="mruColors" type="CT_MRUColors" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

3.8.12 condense (Condense)

Macintosh compatibility setting. Represents special word/character rendering on Macintosh, when this flag is set. The effect is to condense the text (squeeze it together). SpreadsheetML applications are not required to render according to this flag.

Parent Elements

Parent Elements
font (§3.8.21); rPr (§3.4.7)

Attributes	Description
val (Value)	A boolean value for the property specified by the parent XML element. If omitted, the default value is `true`. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BooleanProperty">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

3.8.13 diagonal (Diagonal)

This element specifies the color and line style for the diagonal border(s) of a cell, possibly including diagonally up and diagonally down. The line style for diagonal up and diagonal down lines must be the same.

Parent Elements
border (§3.8.4)

Child Elements	Subclause
color (Data Bar Color)	§3.3.1.14

Attributes	Description
style (Line Style)	The line style for this border. The possible values for this attribute are defined by the ST_BorderStyle simple type (§3.18.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BorderPr">
   <sequence>
      <element name="color" type="CT_Color" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="style" type="ST_BorderStyle" use="optional" default="none"/>
</complexType>
```

3.8.14 dxf (Formatting)

A single dxf record, expressing incremental formatting to be applied.

Parent Elements
dxfs (§3.8.15); rfmt (§3.11.1.17)

Child Elements	Subclause
alignment (Alignment)	§3.8.1
border (Border)	§3.8.4
extLst (Future Feature Data Storage Area)	§3.2.10
fill (Fill)	§3.8.19
font (Font)	§3.8.21
numFmt (Number Format)	§3.8.30
protection (Protection Properties)	§3.8.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Dxf">
   <sequence>
      <element name="font" type="CT_Font" minOccurs="0" maxOccurs="1"/>
      <element name="numFmt" type="CT_NumFmt" minOccurs="0" maxOccurs="1"/>
      <element name="fill" type="CT_Fill" minOccurs="0" maxOccurs="1"/>
      <element name="alignment" type="CT_CellAlignment" minOccurs="0" maxOccurs="1"/>
      <element name="border" type="CT_Border" minOccurs="0" maxOccurs="1"/>
      <element name="protection" type="CT_CellProtection" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

3.8.15 dxfs (Formats)

This element contains the master differential formatting records (dxf's) which define formatting for all non-cell formatting in this workbook. Whereas xf records fully specify a particular aspect of formatting (e.g., cell borders) by referencing those formatting definitions elsewhere in the Styles part, dxf records specify incremental (or differential) aspects of formatting directly inline within the dxf element. The dxf formatting is to be applied on top of or in addition to any formatting already present on the object using the dxf record.

Parent Elements
styleSheet (§3.8.39)

Child Elements	Subclause
dxf (Formatting)	§3.8.14

Attributes	Description

Attributes	Description
count (Format Count)	Count of dxf elements. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Dxfs">
   <sequence>
      <element name="dxf" type="CT_Dxf" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.8.16 extend (Extend)

This element specifies a compatibility setting used for previous spreadsheet applications, resulting in special word/character rendering on those legacy applications, when this flag is set. The effect extends or stretches out the text. SpreadsheetML applications are not required to render according to this flag.

Parent Elements
font (§3.8.21); rPr (§3.4.7)

Attributes	Description
val (Value)	A boolean value for the property specified by the parent XML element. If omitted, the default value is `true`. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BooleanProperty">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

3.8.17 family (Font Family)

The font family this font belongs to. A font family is a set of fonts having common stroke width and serif characteristics. This is system level font information. The font name overrides when there are conflicting values.

Value	Font Family
0	Not applicable.
1	Roman
2	Swiss

Value	Font Family
3	Modern
4	Script
5	Decorative

Parent Elements
font (§3.8.21); rPr (§3.4.7)

Attributes	Description
val (Value)	The value of an integer, where each value corresponds to a different character set. Valid values are 0 to 255. The possible values for this attribute are defined by the XML Schema int datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_IntProperty">
   <attribute name="val" type="xsd:int" use="required"/>
</complexType>
```

3.8.18 fgColor (Foreground Color)

Foreground color of the cell fill pattern. Cell fill patterns operate with two colors: a background color and a foreground color. These combine together to make a patterned cell fill.

Parent Elements
patternFill (§3.8.32)

Attributes	Description
auto (Automatic)	A boolean value indicating the color is automatic and system color dependent. The possible values for this attribute are defined by the XML Schema boolean datatype.
indexed (Index)	Indexed color value. Only used for backwards compatibility. References a color in indexedColors. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
rgb (Alpha Red Green Blue Color Value)	Standard Alpha Red Green Blue color value (ARGB). The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§3.18.86).

Attributes	Description
theme (Theme Color)	Index into the <clrScheme> collection, referencing a particular <sysClr> or <srgbClr> value expressed in the Theme part. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
tint (Tint)	Specifies the tint value applied to the color. If tint is supplied, then it is applied to the RGB value of the color to determine the final color applied. The tint value is stored as a double from -1.0 .. 1.0, where -1.0 means 100% darken and 1.0 means 100% lighten. Also, 0.0 means no change. In loading the RGB value, it is converted to HLS where HLS values are (0..HLSMAX), where HLSMAX is currently 255. [*Example*: Here are some examples of how to apply tint to color: **If (tint < 0)** Lum' = Lum * (1.0 + tint) For example: Lum = 200; tint = -0.5; Darken 50% Lum‘ = 200 * (0.5) => 100 For example: Lum = 200; tint = -1.0; Darken 100% (make black) Lum‘ = 200 * (1.0-1.0) => 0 **If (tint > 0)** Lum‘ = Lum * (1.0-tint) + (HLSMAX – HLSMAX * (1.0-tint)) For example: Lum = 100; tint = 0.75; Lighten 75% Lum‘ = 100 * (1-.75) + (HLSMAX – HLSMAX*(1-.75)) = 100 * .25 + (255 – 255 * .25) = 25 + (255 – 63) = 25 + 192 = 217 For example: Lum = 100; tint = 1.0; Lighten 100% (make white) Lum‘ = 100 * (1-1) + (HLSMAX – HLSMAX*(1-1)) = 100 * 0 + (255 – 255 * 0) = 0 + (255 – 0) = 255 *end example*] The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
    <attribute name="auto" type="xsd:boolean" use="optional"/>
    <attribute name="indexed" type="xsd:unsignedInt" use="optional"/>
    <attribute name="rgb" type="ST_UnsignedIntHex" use="optional"/>
    <attribute name="theme" type="xsd:unsignedInt" use="optional"/>
    <attribute name="tint" type="xsd:double" use="optional" default="0.0"/>
</complexType>
```

3.8.19 fill (Fill)

This element specifies fill formatting.

Parent Elements
dxf (§3.8.14); fills (§3.8.20); ndxf (§3.11.1.4); odxf (§3.11.1.6)

Child Elements	Subclause
gradientFill (Gradient)	§3.8.23
patternFill (Pattern)	§3.8.32

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Fill">
    <choice minOccurs="1" maxOccurs="1">
        <element name="patternFill" type="CT_PatternFill" minOccurs="0" maxOccurs="1"/>
        <element name="gradientFill" type="CT_GradientFill" minOccurs="0" maxOccurs="1"/>
    </choice>
</complexType>
```

3.8.20 fills (Fills)

This element defines the cell fills portion of the Styles part, consisting of a sequence of fill records. A cell fill consists of a background color, foreground color, and pattern to be applied across the cell.

[*Example*: This cell has a yellow fill:

This is the corresponding XML:

```
<fill>
  <patternFill patternType="solid">
    <fgColor rgb="FFFFFF00"/>
    <bgColor indexed="64"/>
  </patternFill>
</fill>
```

This cell has a yellow fill with a thin horizontal crosshatch pattern applied (patternType = `lightGrid`):

This is the corresponding XML:

```
<fill>
  <patternFill patternType="lightGrid">
  <bgColor rgb="FFFFFF00"/>
  </patternFill>
</fill>
```

end example]

Parent Elements
styleSheet (§3.8.39)

Child Elements	Subclause
fill (Fill)	§3.8.19

Attributes	Description
count (Fill Count)	Count of fill elements. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Fills">
   <sequence>
      <element name="fill" type="CT_Fill" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.8.21 font (Font)

This element defines the properties for one of the fonts used in this workbook.

Parent Elements
dxf (§3.8.14); fonts (§3.8.22); ndxf (§3.11.1.4); odxf (§3.11.1.6)

Child Elements	Subclause

Child Elements	Subclause
b (Bold)	§3.8.2
charset (Character Set)	§3.4.1
color (Data Bar Color)	§3.3.1.14
condense (Condense)	§3.8.12
extend (Extend)	§3.8.16
family (Font Family)	§3.8.17
i (Italic)	§3.8.25
name (Font Name)	§3.8.29
outline (Outline)	§3.4.2
scheme (Scheme)	§3.8.36
shadow (Shadow)	§3.8.37
strike (Strike Through)	§3.4.10
sz (Font Size)	§3.4.11
u (Underline)	§3.4.13
vertAlign (Vertical Alignment)	§3.4.14

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Font">
   <choice maxOccurs="unbounded">
      <element name="name" type="CT_FontName" minOccurs="0" maxOccurs="1"/>
      <element name="charset" type="CT_IntProperty" minOccurs="0" maxOccurs="1"/>
      <element name="family" type="CT_IntProperty" minOccurs="0" maxOccurs="1"/>
      <element name="b" type="CT_BooleanProperty" minOccurs="0" maxOccurs="1"/>
      <element name="i" type="CT_BooleanProperty" minOccurs="0" maxOccurs="1"/>
      <element name="strike" type="CT_BooleanProperty" minOccurs="0" maxOccurs="1"/>
      <element name="outline" type="CT_BooleanProperty" minOccurs="0" maxOccurs="1"/>
      <element name="shadow" type="CT_BooleanProperty" minOccurs="0" maxOccurs="1"/>
      <element name="condense" type="CT_BooleanProperty" minOccurs="0" maxOccurs="1"/>
      <element name="extend" type="CT_BooleanProperty" minOccurs="0" maxOccurs="1"/>
      <element name="color" type="CT_Color" minOccurs="0" maxOccurs="1"/>
      <element name="sz" type="CT_FontSize" minOccurs="0" maxOccurs="1"/>
      <element name="u" type="CT_UnderlineProperty" minOccurs="0" maxOccurs="1"/>
      <element name="vertAlign" type="CT_VerticalAlignFontProperty" minOccurs="0" maxOccurs="1"/>
      <element name="scheme" type="CT_FontScheme" minOccurs="0" maxOccurs="1"/>
   </choice>
</complexType>
```

3.8.22 fonts (Fonts)

This element contains all font definitions for this workbook.

[*Example*: This example expresses two fonts in the workbook. A Calibri family font, with font size of 11, and an Arial family font, with font size 12. The second font has strikethrough applied.

```
<fonts count="2">
  <font>
    <sz val="11"/>
    <color theme="1"/>
    <name val="Calibri"/>
    <family val="2"/>
    <scheme val="minor"/>
  </font>
  <font>
    <strike/>
    <sz val="12"/>
    <color theme="1"/>
    <name val="Arial"/>
    <family val="2"/>
  </font>
</fonts>
```

end example]

Parent Elements
styleSheet (§3.8.39)

Child Elements	Subclause
font (Font)	§3.8.21

Attributes	Description
count (Font Count)	Count of font elements. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Fonts">
   <sequence>
      <element name="font" type="CT_Font" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.8.23 gradientFill (Gradient)

This element defines a gradient-style cell fill. Gradient cell fills can use one or two colors as the end points of color interpolation.

[*Example*:

This example shows a gradient cell fill, with color green at the top transitioning into blue at the bottom.

This is the XML:

```
<fill>
  <gradientFill degree="90">
    <stop position="0">
      <color rgb="FF92D050"/>
    </stop>
    <stop position="1">
      <color rgb="FF0070C0"/>
    </stop>
  </gradientFill>
</fill>
```

This example shows a gradient cell fill, from the center. Note the left, right, top, and bottom values (and see explanation in the attribute section):

```
<fill>
  <gradientFill type="path" left="0.2" right="0.8" top="0.2" bottom="0.8">
    <stop position="0">
      <color theme="0"/>
    </stop>
    <stop position="1">
      <color theme="4"/>
    </stop>
  </gradientFill>
</fill>
```

end example]

Parent Elements

Parent Elements
fill (§3.8.19)

Child Elements	Subclause
stop (Gradient Stop)	§3.8.38

Attributes	Description
bottom (Bottom Convergence)	Valid values are 0 to 1. Specifies in percentage format (from the top to the bottom) the position of the bottom edge of the inner rectangle (color 1). For bottom, 0 means the bottom edge of the inner rectangle is on the top edge of the cell, and 1 means it is on the bottom edge of the cell. The possible values for this attribute are defined by the XML Schema double datatype.
degree (Linear Gradient Degree)	Angle of the linear gradient - vertical, horizontal, diagonal. [*Example*: Note: in these examples, color 1 is white and color 2 is blue. 90 = Horizontal & color 1 to color 2  270 = Horizontal & color 1 to color 2  0 = Vertical & color 1 to color 2  180 = Vertical & color 1 to color 2  45 = Diagonal Up & top to bottom (color 1 to color 2) 225 = Diagonal Up & bottom to top (color 1 to color 2) 135 = Diagonal Down & top to bottom (color 1 to color 2) 315 = Diagonal Down & bottom to top (color 1 to color 2) *end example*]

Attributes	Description
	The possible values for this attribute are defined by the XML Schema double datatype.
left (Left Convergence)	Valid values are 0 to 1. Specifies in percentage format (from the left to the right) the position of the left edge of the inner rectangle (color 1). For left, 0 means the left edge of the inner rectangle is on the left edge of the cell, and 1 means it is on the right edge of the cell. (applies to From Corner and From Center gradients). The possible values for this attribute are defined by the XML Schema double datatype.
right (Right Convergence)	Valid values are 0 to 1. Specifies in percentage format (from the left to the right) the position of the right edge of the inner rectangle (color 1). For right, 0 means the right edge of the inner rectangle is on the left edge of the cell, and 1 means it is on the right edge of the cell. (applies to From Corner and From Center gradients). The possible values for this attribute are defined by the XML Schema double datatype.
top (Top Gradient Convergence)	Valid values are 0 to 1. Specifies in percentage format (from the top to the bottom) the position of the top edge of the inner rectangle (color 1). For top, 0 means the top edge of the inner rectangle is on the top edge of the cell, and 1 means it is on the bottom edge of the cell. (applies to From Corner and From Center gradients). The possible values for this attribute are defined by the XML Schema double datatype.
type (Gradient Fill Type)	Type of this gradient fill. The possible values for this attribute are defined by the ST_GradientType simple type (§3.18.38).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GradientFill">
   <sequence>
      <element name="stop" type="CT_GradientStop" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="type" type="ST_GradientType" use="optional" default="linear"/>
   <attribute name="degree" type="xsd:double" use="optional" default="0"/>
   <attribute name="left" type="xsd:double" use="optional" default="0"/>
   <attribute name="right" type="xsd:double" use="optional" default="0"/>
   <attribute name="top" type="xsd:double" use="optional" default="0"/>
   <attribute name="bottom" type="xsd:double" use="optional" default="0"/>
</complexType>
```

3.8.24 horizontal (Horizontal Inner Borders)

This element specifies the color and line style for the horizontal inner border(s) of a range of cells. Used in the context of dxf elements only. For example, see the borders definitions for **TableStyleMedium28**.

Parent Elements
border (§3.8.4)

Child Elements	Subclause
color (Data Bar Color)	§3.3.1.14

Attributes	Description
style (Line Style)	The line style for this border. The possible values for this attribute are defined by the ST_BorderStyle simple type (§3.18.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BorderPr">
   <sequence>
      <element name="color" type="CT_Color" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="style" type="ST_BorderStyle" use="optional" default="none"/>
</complexType>
```

3.8.25 i (Italic)

Displays characters in italic font style. The italic style is defined by the font at a system level and is not specified by this specification.

Parent Elements
font (§3.8.21); rPr (§3.4.7)

Attributes	Description
val (Value)	A boolean value for the property specified by the parent XML element. If omitted, the default value is `true`. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

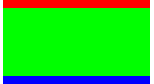
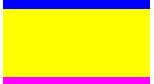
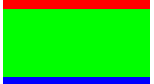
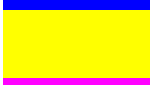
```
<complexType name="CT_BooleanProperty">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

3.8.26 indexedColors (Color Indexes)

A deprecated indexing scheme for colours that is still required for some records, and for backwards compatibility with legacy formats.

This element contains a sequence of RGB color values that correspond to color indexes (zero-based). When using the default indexed color palette, the values are not written out, but instead are implied. When the color palette has been modified from default, then the entire color palette is written out.

Here is the table of default mappings from indexed color value to ARGB value. Note that 0-7 are redundant of 8-15 to preserve backwards compatibility.

Color Index	**ARGB Value**	[*Example*:
indexed="0"	00000000	
indexed="1"	00FFFFFF	
indexed="2"	00FF0000	
indexed="3"	0000FF00	
indexed="4"	000000FF	
indexed="5"	00FFFF00	
indexed="6"	00FF00FF	
indexed="7"	0000FFFF	
(none)	(none)	
indexed="8"	00000000	
indexed="9"	00FFFFFF	
indexed="10"	00FF0000	
indexed="11"	0000FF00	
indexed="12"	000000FF	
indexed="13"	00FFFF00	
indexed="14"	00FF00FF	
indexed="15"	0000FFFF	
indexed="16"	00800000	
indexed="17"	00008000	
indexed="18"	00000080	

indexed="19"	00808000	
indexed="20"	00800080	
indexed="21"	00008080	
indexed="22"	00C0C0C0	
indexed="23"	00808080	
indexed="24"	009999FF	
indexed="25"	00993366	
indexed="26"	00FFFFCC	
indexed="27"	00CCFFFF	
indexed="28"	00660066	
indexed="29"	00FF8080	
indexed="30"	000066CC	
indexed="31"	00CCCCFF	
indexed="32"	00000080	
indexed="33"	00FF00FF	
indexed="34"	00FFFF00	
indexed="35"	0000FFFF	
indexed="36"	00800080	
indexed="37"	00800000	
indexed="38"	00008080	
indexed="39"	000000FF	
indexed="40"	0000CCFF	
indexed="41"	00CCFFFF	
indexed="42"	00CCFFCC	
indexed="43"	00FFFF99	

indexed="44"	0099CCFF	
indexed="45"	00FF99CC	
indexed="46"	00CC99FF	
indexed="47"	00FFCC99	
indexed="48"	003366FF	
indexed="49"	0033CCCC	
indexed="50"	0099CC00	
indexed="51"	00FFCC00	
indexed="52"	00FF9900	
indexed="53"	00FF6600	
indexed="54"	00666699	
indexed="55"	00969696	
indexed="56"	00003366	
indexed="57"	00339966	
indexed="58"	00003300	
indexed="59"	00333300	
indexed="60"	00993300	
indexed="61"	00993366	
indexed="62"	00333399	
indexed="63"	00333333	
indexed="64"	System Foreground	n/a
indexed="65"	System Background	n/a

Parent Elements
colors (§3.8.11)

Child Elements	Subclause
rgbColor (RGB Color)	§3.8.34

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_IndexedColors">
   <sequence>
      <element name="rgbColor" type="CT_RgbColor" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.8.27 left (Left Border)

This element specifies the color and line style for the left border of a cell.

Parent Elements
border (§3.8.4)

Child Elements	Subclause
color (Data Bar Color)	§3.3.1.14

Attributes	Description
style (Line Style)	The line style for this border. The possible values for this attribute are defined by the ST_BorderStyle simple type (§3.18.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BorderPr">
   <sequence>
      <element name="color" type="CT_Color" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="style" type="ST_BorderStyle" use="optional" default="none"/>
</complexType>
```

3.8.28 mruColors (MRU Colors)

This element contains sequence of RGB values that correspond to custom colors selected by the user for this workbook.

Parent Elements
colors (§3.8.11)

Child Elements	Subclause

Child Elements	Subclause
color (Data Bar Color)	§3.3.1.14

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MRUColors">
   <sequence>
      <element name="color" type="CT_Color" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.8.29 name (Font Name)

This element specifies the face name of this font.

Parent Elements
font (§3.8.21)

Attributes	Description
val (String Value)	A string representing the name of the font. If the font doesn't exist (because it isn't installed on the system), or the charset is invalid for that font, then another font should be substituted. The string length for this attribute shall be 0 to 31 characters. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FontName">
   <attribute name="val" type="ST_Xstring" use="required"/>
</complexType>
```

3.8.30 numFmt (Number Format)

This element specifies number format properties which indicate how to format and render the numeric value of a cell.

Following is a listing of number formats whose formatCode value is implied rather than explicitly saved in the file. In this case a numFmtId value is written on the xf record, but no corresponding numFmt element is written. Some of these Ids are interpreted differently, depending on the UI language of the implementing application.

All Languages

ID	formatCode

ID	formatCode
0	General
1	0
2	0.00
3	#,##0
4	#,##0.00
9	0%
10	0.00%
11	0.00E+00
12	# ?/?
13	# ??/??
14	mm-dd-yy
15	d-mmm-yy
16	d-mmm
17	mmm-yy
18	h:mm AM/PM
19	h:mm:ss AM/PM
20	h:mm
21	h:mm:ss
22	m/d/yy h:mm
37	#,##0 ;(#,##0)
38	#,##0 ;[Red](#,##0)
39	#,##0.00;(#,##0.00)
40	#,##0.00;[Red](#,##0.00)
45	mm:ss
46	[h]:mm:ss
47	mmss.0
48	##0.0E+0
49	@

"General" Format

Some additional comments about the "General" number format are appropriate.

The primary goal when a cell is using "General" formatting is to render the cell content without user-specified guidance to the best ability of the application.

Alignment

(Specified for Left-to-Right mode)

- Strings: left aligned
- Boolean/error values: centered
- Numbers: right aligned
- Dates: do not follow the "General" format, instead automatically convert to date formatting.

Numbers

The application shall attempt to display the full number up to 11 digits (inc. decimal point). If the number is too large, the application shall attempt to show exponential format. If the number has too many significant digits, the display shall be truncated. The optimal method of display is based on the available cell width. If the number cannot be displayed using any of these formats in the available width, the application shall show "#" across the width of the cell.

Conditions for switching to exponential format:

14. The cell value must have at least five digits for xE-xx
15. If the exponent is bigger than the size allowed, a floating point number cannot fit, so try exponential notation.
16. Similarly, for negative exponents, check if there is space for even one (non-zero) digit in floating point format.
17. Finally, if there isn't room for all of the significant digits in floating point format (for a negative exponent), exponential format shall display more digits if the exponent is less than -3. (The 3 is because E-xx takes 4 characters, and the leading 0 in floating point takes only 1 character. Thus, for an exponent less than -3, there is more than 3 additional leading 0's, more than enough to compensate for the size of the E-xx.)

Floating point rule:

For general formatting in cells, max overall length for cell display is 11, not including negative sign, but includes leading zeros and decimal separator.

CHT and CHS

ID	CHT formatCode	CHS formatCode
27	[$-404]e/m/d	yyyy"年"m"月"
28	[$-404]e"年"m"月"d"日"	m"月"d"日"
29	[$-404]e"年"m"月"d"日"	m"月"d"日"

ID	CHT formatCode	CHS formatCode
30	m/d/yy	m-d-yy
31	yyyy"年"m"月"d"日"	yyyy"年"m"月"d"日"
32	hh"時"mm"分"	h"时"mm"分"
33	hh"時"mm"分"ss"秒"	h"时"mm"分"ss"秒"
34	上午/下午hh"時"mm"分"	上午/下午h"时"mm"分"
35	上午/下午hh"時"mm"分"ss"秒"	上午/下午h"时"mm"分"ss"秒"
36	[$-404]e/m/d	yyyy"年"m"月"
50	[$-404]e/m/d	yyyy"年"m"月"
51	[$-404]e"年"m"月"d"日"	m"月"d"日"
52	上午/下午hh"時"mm"分"	yyyy"年"m"月"
53	上午/下午hh"時"mm"分"ss"秒"	m"月"d"日"
54	[$-404]e"年"m"月"d"日"	m"月"d"日"
55	上午/下午hh"時"mm"分"	上午/下午h"时"mm"分"
56	上午/下午hh"時"mm"分"ss"秒"	上午/下午h"时"mm"分"ss"秒"
57	[$-404]e/m/d	yyyy"年"m"月"
58	[$-404]e"年"m"月"d"日"	m"月"d"日"

CHT and CHS (with unicode values provided for language glyphs where they occur)

ID	CHT formatCode	CHS formatCode
27	[$-404]e/m/d	yyyy"5E74"m"6708"
28	[$-404]e"5E74"m"6708"d"65E5"	m"6708"d"65E5"
29	[$-404]e"5E74"m"6708"d"65E5"	m"6708"d"65E5"
30	m/d/yy	m-d-yy
31	yyyy"5E74"m"6708"d"65E5"	yyyy"5E74"m"6708"d"65E5"
32	hh"6642"mm"5206"	h"65F6"mm"5206"
33	hh"6642"mm"5206"ss"79D2"	h"65F6"mm"5206"ss"79D2"
34	4E0A5348/4E0B5348hh"6642"mm"5206"	4E0A5348/4E0B5348h"65F6"mm"5206"
35	4E0A5348/4E0B5348hh"6642"mm"5206"ss"79D2"	4E0A5348/4E0B5348h"65F6"mm"5206"ss"79D2"
36	[$-404]e/m/d	yyyy"5E74"m"6708"
50	[$-404]e/m/d	yyyy"5E74"m"6708"

ID	CHT formatCode	CHS formatCode
51	[$-404]e"5E74"m"6708"d"65E5"	m"6708"d"65E5"
52	4E0A5348/4E0B5348hh"6642"mm"5206"	yyyy"5E74"m"6708"
53	4E0A5348/4E0B5348hh"6642"mm"5206"ss"79D2"	m"6708"d"65E5"
54	[$-404]e"5E74"m"6708"d"65E5"	m"6708"d"65E5"
55	4E0A5348/4E0B5348hh"6642"mm"5206"	4E0A5348/4E0B5348h"65F6"mm"5206"
56	4E0A5348/4E0B5348hh"6642"mm"5206"ss"79D2"	4E0A5348/4E0B5348h"65F6"mm"5206"ss"79D2"
57	[$-404]e/m/d	yyyy"5E74"m"6708"
58	[$-404]e"5E74"m"6708"d"65E5"	m"6708"d"65E5"

JPN and KOR

ID	JPN formatCode	KOR formatCode
27	[$-411]ge.m.d	yyyy"年" mm"月" dd"日"
28	[$-411]ggge"年"m"月"d"日"	mm-dd
29	[$-411]ggge"年"m"月"d"日"	mm-dd
30	m/d/yy	mm-dd-yy
31	yyyy"年"m"月"d"日"	yyyy"년" mm"월" dd"일"
32	h"時"mm"分"	h"시" mm"분"
33	h"時"mm"分"ss"秒"	h"시" mm"분" ss"초"
34	yyyy"年"m"月"	yyyy-mm-dd
35	m"月"d"日"	yyyy-mm-dd
36	[$-411]ge.m.d	yyyy"年" mm"月" dd"日"
50	[$-411]ge.m.d	yyyy"年" mm"月" dd"日"
51	[$-411]ggge"年"m"月"d"日"	mm-dd
52	yyyy"年"m"月"	yyyy-mm-dd
53	m"月"d"日"	yyyy-mm-dd
54	[$-411]ggge"年"m"月"d"日"	mm-dd
55	yyyy"年"m"月"	yyyy-mm-dd
56	m"月"d"日"	yyyy-mm-dd
57	[$-411]ge.m.d	yyyy"年" mm"月" dd"日"

ID	JPN formatCode	KOR formatCode
58	[$-411]ggge"年"m"月"d"日"	mm-dd

JPN and KOR (with unicode values provided for language glyphs where they occur)

ID	JPN formatCode	KOR formatCode
27	[$-411]ge.m.d	yyyy"5E74" mm"6708" dd"65E5"
28	[$-411]ggge"5E74"m"6708"d"65E5"	mm-dd
29	[$-411]ggge"5E74"m"6708"d"65E5"	mm-dd
30	m/d/yy	mm-dd-yy
31	yyyy"5E74"m"6708"d"65E5"	yyyy"B144" mm"C6D4" dd"C77C"
32	h"6642"mm"5206"	h"C2DC" mm"BD84"
33	h"6642"mm"5206"ss"79D2"	h"C2DC" mm"BD84" ss"CD08"
34	yyyy"5E74"m"6708"	yyyy-mm-dd
35	m"6708"d"65E5"	yyyy-mm-dd
36	[$-411]ge.m.d	yyyy"5E74" mm"6708" dd"65E5"
50	[$-411]ge.m.d	yyyy"5E74" mm"6708" dd"65E5"
51	[$-411]ggge"5E74"m"6708"d"65E5"	mm-dd
52	yyyy"5E74"m"6708"	yyyy-mm-dd
53	m"6708"d"65E5"	yyyy-mm-dd
54	[$-411]ggge"5E74"m"6708"d"65E5"	mm-dd
55	yyyy"5E74"m"6708"	yyyy-mm-dd
56	m"6708"d"65E5"	yyyy-mm-dd
57	[$-411]ge.m.d	yyyy"5E74" mm"6708" dd"65E5"
58	[$-411]ggge"5E74"m"6708"d"65E5"	mm-dd

THA

ID	THA formatCode
59	t0
60	t0.00
61	t#,##0
62	t#,##0.00

ID	THA formatCode
67	t0%
68	t0.00%
69	t# ?/?
70	t# ??/??
71	ว/ด/ปปปป
72	ว-ดดด-ปป
73	ว-ดดด
74	ดดด-ปป
75	ช:นน
76	ช:นน:ทท
77	ว/ด/ปปปป ช:นน
78	นน:ทท
79	[ช]:นน:ทท
80	นน:ทท.0
81	d/m/bb

THA (with unicode values provided for language glyphs where they occur)

ID	THA formatCode
59	t0
60	t0.00
61	t#,##0
62	t#,##0.00
67	t0%
68	t0.00%
69	t# ?/?
70	t# ??/??
71	0E27/0E14/0E1B0E1B0E1B0E1B
72	0E27-0E140E140E14-0E1B0E1B
73	0E27-0E140E140E14
74	0E140E140E14-0E1B0E1B

ID	THA formatCode
75	0E0A:0E190E19
76	0E0A:0E190E19:0E170E17
77	0E27/0E14/0E1B0E1B0E1B0E1B 0E0A:0E190E19
78	0E190E19:0E170E17
79	[0E0A]:0E190E19:0E170E17
80	0E190E19:0E170E17.0
81	d/m/bb

Parent Elements
dxf (§3.8.14); ndxf (§3.11.1.4); numFmts (§3.8.31); odxf (§3.11.1.6)

Attributes	Description
formatCode (Number Format Code)	The number format code for this number format. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
numFmtId (Number Format Id)	Id used by the master style records (xf's) to reference this number format. The possible values for this attribute are defined by the ST_NumFmtId simple type (§3.18.49).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NumFmt">
   <attribute name="numFmtId" type="ST_NumFmtId" use="required"/>
   <attribute name="formatCode" type="ST_Xstring" use="required"/>
</complexType>
```

3.8.31 numFmts (Number Formats)

This element defines the number formats in this workbook, consisting of a sequence of numFmt records, where each numFmt record defines a particular number format, indicating how to format and render the numeric value of a cell.

[*Example*:

This cell is formatting as US currency:

The XML expressing this format shows that the formatId is "166" and the decoded formatCode is `$#,##0.00`

```
<numFmts count="1">
  <numFmt numFmtId="166" formatCode="&quot;$&quot;#,##0.00"/>
</numFmts>
```

end example]

Number Format Codes

Up to four sections of format codes can be specified. The format codes, separated by semicolons, define the formats for positive numbers, negative numbers, zero values, and text, in that order. If only two sections are specified, the first is used for positive numbers and zeros, and the second is used for negative numbers. If only one section is specified, it is used for all numbers. To skip a section, the ending semicolon for that section must be written.

The first section, "Format for positive numbers", is the format code that applies to the cell when the cell value contains a positive number.

The second section, "Format for negative numbers", is the format code that applies to the cell when the cell value contains a negative number.

The third section, "Format for zeros", is the format code that applies to the cell when the cell value is zero.

The fourth, and last, section, "Format for text", is the format code that applies to the cell when the cell value is text.

The & (ampersand) text operator is used to join, or concatenate, two values.

The following table describes the different symbols that are available for use in custom number formats.

Format symbol	Description and result
0	Digit placeholder. For example, if the value 8.9 is to be displayed as 8.90, use the format #.00
#	Digit placeholder. This symbol follows the same rules as the 0 symbol. However, the application shall not display extra zeros when the number typed has fewer digits on either side of the decimal than there are # symbols in the format. For example, if the custom format is #.##, and 8.9 is in the cell, the number 8.9 is displayed.
?	Digit placeholder. This symbol follows the same rules as the 0 symbol. However, the application shall put a space for insignificant zeros on either side of the decimal point so that decimal points are aligned in the column. For example, the custom format 0.0? aligns

Format symbol	Description and result
	the decimal points for the numbers 8.9 and 88.99 in a column.
. (period)	Decimal point.
%	Percentage. If the cell contains a number between 0 and 1, and the custom format 0% is used, the application shall multiply the number by 100 and adds the percentage symbol in the cell.
, (comma)	Thousands separator. The application shall separate thousands by commas if the format contains a comma that is enclosed by number signs (#) or by zeros. A comma that follows a placeholder scales the number by one thousand. For example, if the format is #.0,, and the cell value is 12,200,000 then the number 12.2 is displayed.
E- E+ e- e+	Scientific format. The application shall display a number to the right of the "E" symbol that corresponds to the number of places that the decimal point was moved. For example, if the format is 0.00E+00, and the value 12,200,000 is in the cell, the number 1.22E+07 is displayed. If the number format is #0.0E+0, then the number 12.2E+6 is displayed.
$-+/():space	Displays the symbol. If it is desired to display a character that differs from one of these symbols, precede the character with a backslash (\). Alternatively, enclose the character in quotation marks. For example, if the number format is (000), and the value 12 is in the cell, the number (012) is displayed.
\	Display the next character in the format. The application shall not display the backslash. For example, if the number format is 0\!, and the value 3 is in the cell, the value 3! is displayed.
*	Repeat the next character in the format enough times to fill the column to its current width. There shall not be more than one asterisk in one section of the format. If more than one asterisk appears in one section of the format, all but the last asterisk shall be ignored. For example, if the number format is 0*x, and the value 3 is in the cell, the value 3xxxxxx is displayed. The number of x characters that are displayed in the cell varies based on the width of the column.
_ (underline)	Skip the width of the next character. This is useful for lining up negative and positive values in different cells of the same column. For example, the number format _(0.0_);(0.0) aligns the numbers 2.3 and -4.5 in the column even though the negative number is enclosed by parentheses.
"text"	Display whatever text is inside the quotation marks. For example, the format 0.00 "dollars" displays 1.23 dollars when the value 1.23 is in the cell.
@	Text placeholder. If text is typed in the cell, the text from the cell is placed in the format where the at symbol (@) appears. For example, if the number format is "Bob "@" Smith" (including quotation marks), and the value "John" is in the cell, the value Bob John Smith is displayed.

Text and spacing

Display both text and numbers

To display both text and numbers in a cell, enclose the text characters in double quotation marks (" ") or precede a single character with a backslash (\). Single quotation marks shall not be used to denote text. Characters inside double quotes, or immediately following backslash shall never be interpreted as part of the format code lexicon; instead they shall always be treated as literal strings. Remember to include the characters in the appropriate section of the format codes. For example, type the format $0.00" Surplus";$-0.00" Shortage" to display a positive amount as "$125.74 Surplus" and a negative amount as "$-125.74 Shortage."

The following characters are displayed without the use of quotation marks.

$	Dollar sign				-	Minus sign
+	Plus sign				/	Slash mark
(	Left parenthesis				)	Right parenthesis
:	Colon				!	Exclamation point
^	Circumflex accent (caret)				&	Ampersand
'	Apostrophe				~	Tilde
{	Left curly bracket				}	Right curly bracket
<	Less-than sign				>	Greater-than sign
=	Equal sign					Space character

Include a section for text entry

If included, a text section must be the last section in the number format. Include an "at" sign (@) in the section, precisely where the cell's text value should be displayed. If the @ character is omitted from the text section, text typed in the cell will not be displayed. To always display specific text characters with the typed text, enclose the additional text in double quotation marks (" "). For example, if "June" is typed into the cell, and the text format is "gross receipts for "@ , then the cell will display "gross receipts for June".

If the format does not include a text section, text entered in a cell is not affected by the format code.

Add spaces

To create a space that is the width of a character in a number format, include an underscore, followed by the character. For example, when an underscore is followed with a right parenthesis, such as _), positive numbers line up correctly with negative numbers that are enclosed in parentheses because positive numbers are displayed with a blank space after them exactly the width of the right parenthesis character.

Repeat characters

To repeat the next character in the format to fill the column width, include an asterisk (*) in the number format. For example, type 0*- to include enough dashes after a number to fill the cell, or type *0 before any format to include leading zeros.

Decimal places, spaces, colors, and conditions

Include decimal places and significant digits

To format fractions or numbers with decimal points, include the following digit placeholders in a section. If a number has more digits to the right of the decimal point than there are placeholders in the format, the number rounds to as many decimal places as there are placeholders. If there are more digits to the left of the decimal point than there are placeholders, the extra digits are displayed. If the format contains only number signs (#) to the left of the decimal point, numbers less than 1 begin with a decimal point.

(number sign) displays only significant digits and does not display insignificant zeros.

0 (zero) displays insignificant zeros if a number has fewer digits than there are zeros in the format.

? (question mark) adds spaces for insignificant zeros on either side of the decimal point so that decimal points align when they are formatted with a fixed-width font, such as Courier New. ? can also be used for fractions that have varying numbers of digits.

To display	As	Use this code
1234.59	1234.6	####.#
8.9	8.900	#.000
.631	0.6	0.#
12 1234.568	12.0 1234.57	#.0#
44.398 102.65 2.8	44.398 102.65 2.8 (with aligned decimals)	???.???
5.25 5.3	5 1/4 5 3/10 (with aligned fractions)	# ???/???

Display a thousands separator

To display a comma as a thousands separator or to scale a number by a multiple of 1,000, include a comma in the number format.

To display	As	Use this code
12000	12,000	#,###
12000	12	#,
12200000	12.2	0.0,,

Specify colors

To set the text color for a section of the format, type the name of one of the following eight colors in square brackets in the section. The color code must be the first item in the section.

[Black]			[Blue]			[Cyan]
[Green]			[Magenta]			[Red]
[White]			[Yellow]			

Instead of using the name of the color, the color index can be used, like this [Color3] for Red. Valid numeric indexes for color range from 1 to 56, which reference by index to the legacy color palette.

[*Note*: the default legacy color palette values are listed in §3.8.26. In the format codes, [Color1] refers to the color associated with indexed="8", or black (by default), [Color2] refers to the color associated with indexed="9", or white (by default), and so on up to [Color56] referring to the color associated with indexed="63". If the color palette has been customized from default values, then the colors associated with these indexes will reflect those customizations.

Specify conditions

To set number formats that will be applied only if a number meets a specified condition, enclose the condition in square brackets. The condition consists of a comparison operator and a value. Comparison operators include: = Equal to; > Greater than; < Less than; >= Greater than or equal to, <= Less than or equal to, and <> Not equal to. For example, the following format displays numbers that are less than or equal to 100 in a red font and numbers that are greater than 100 in a blue font.

[Red][<=100];[Blue][>100]

If the cell value does not meet any of the criteria, then pound signs ("#") are displayed across the width of the cell.

Currency, percentages, and scientific notation

Include currency symbols

To include currency symbols, place the currency symbol in the location it should when displayed.

Display percentages

To display numbers as a percentage of 100 — for example, to display .08 as 8% or 2.8 as 280% — include the percent sign (%) in the number format.

Display scientific notations

To display numbers in scientific format, use exponent codes in a section — for example, E-, E+, e-, or e+.

If a format contains a zero (0) or number sign (#) to the right of an exponent code, the application displays the number in scientific format and inserts an "E" or "e". The number of zeros or number signs to the right of a code determines the number of digits in the exponent. "E-" or "e-" places a minus sign by negative exponents. "E+" or "e+" places a minus sign by negative exponents and a plus sign by positive exponents.

Dates and times

Display days, months, and years

To display	As	Use this code
Months	1–12	m
Months	01–12	mm
Months	Jan–Dec	mmm
Months	January–December	mmmm
Months	J–D	mmmmm
Days	1–31	d
Days	01–31	dd
Days	Sun–Sat	ddd
Days	Sunday–Saturday	dddd
Years	00–99	yy
Years	1900–9999	yyyy

See §3.17.4.1 for special handling of certain days in the year 1900.

Month versus minutes

If "m" or "mm" code is used immediately after the "h" or "hh" code (for hours) or immediately before the "ss" code (for seconds), the application shall display minutes instead of the month.

Display hours, minutes, and seconds

To display	As	Use this code

Hours	0–23	h
Hours	00–23	hh
Minutes	0–59	m
Minutes	00–59	mm
Seconds	0–59	s
Seconds	00–59	ss
Time	4 AM	h AM/PM
Time	4:36 PM	h:mm AM/PM
Time	4:36:03 P	h:mm:ss A/P
Time	4:36:03.75	h:mm:ss.00
Elapsed time (hours and minutes)	1:02	[h]:mm
Elapsed time (minutes and seconds)	62:16	[mm]:ss
Elapsed time (seconds and hundredths)	3735.80	[ss].00

Minutes versus month

The "m" or "mm" code must appear immediately after the "h" or "hh" code or immediately before the "ss" code; otherwise, these will display as the month instead of minutes.

AM and PM

If the format contains AM or PM, the hour is based on the 12-hour clock, where "AM" or "A" indicates times from midnight until noon and "PM" or "P" indicates times from noon until midnight. Otherwise, the hour is based on the 24-hour clock.

Invalid date and time values

Cells formatted with a date or time format and which contain invalid date or time values shall show the pound sign ("#") across the width of the cell.

International Considerations

Format Code	Description
r	JPN/CHT Only. When loading in JPN locale, code becomes "ee". When loading in CHT locale, code becomes "e".
rr	JPN/CHT Only. When loading in JPN locale, code becomes "gggee". When loading in CHT locale, code becomes "e".
g	When loading in JPN locale: Single Roman character emperor reign When loading in CHT (Taiwan only) locale: treat same as "gg".
gg	When loading in JPN locale: Single Kanji character emperor reign When loading in CHT locale: Last era short name (since 1911)
ggg	When loading in JPN locale: Tow Kanji character emperor reign When loading in CHT locale: Last era long name (since 1911)
e	When loading in JPN locale: Era year When laoding in CHT (Tawian only) locale: Era year since 1912. If preceeded by “g”, “gg”, or “ggg” then year of 1912, and year before 1912 are special, otherwise years less than 1912 are gregorian. OTHER locales: becomes "yy"
ee	When loading in JPN locale: Era year w/ leading zero When loading in CHT (Tawian only) locale: Era year since 1911 OTHER locales: becomes "yy"
b2	Hijri calander
b1	Gregorian calendar
[$USD-409]	Specifies currency and locale/date system/number system information. Syntax is [$<Currency String>-<language info>]. Currency string is a string to use as a currency symbol. Language info is a 32-bit value entered in hexidecimal format. Language info format (byte 3 is most significant byte): Bytes 0,1: 16-bit Language ID (LID). Byte 2: Calendar type. High bit indicates that input is parsed using specified calendar. Byte 3: Number system type. High bit indicates that input is parsed using specified number system.

Format Code	Description
	Special language info values: 0xf800: System long date format 0xf400: System time format

Parent Elements
styleSheet (§3.8.39)

Child Elements	Subclause
numFmt (Number Format)	§3.8.30

Attributes	Description
count (Number Format Count)	Count of number format elements. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NumFmts">
   <sequence>
      <element name="numFmt" type="CT_NumFmt" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.8.32 patternFill (Pattern)

This element is used to specify cell fill information for pattern and solid color cell fills. For solid cell fills (no pattern), fgColor is used. For cell fills with patterns specified, then the cell fill color is specified by the bgColor element.

Parent Elements
fill (§3.8.19)

Child Elements	Subclause
bgColor (Background Color)	§3.8.3
fgColor (Foreground Color)	§3.8.18

Attributes	Description
patternType (Pattern Type)	Specifies the fill pattern type (including solid and none) Default is none, when missing. The possible values for this attribute are defined by the ST_PatternType simple type (§3.18.57).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PatternFill">
   <sequence>
      <element name="fgColor" type="CT_Color" minOccurs="0" maxOccurs="1"/>
      <element name="bgColor" type="CT_Color" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="patternType" type="ST_PatternType" use="optional"/>
</complexType>
```

3.8.33 protection (Protection Properties)

Contains protection properties associated with the cell. Each cell has protection properties that can be set. The cell protection properties do not take effect unless the sheet has been protected.

Parent Elements
dxf (§3.8.14); ndxf (§3.11.1.4); odxf (§3.11.1.6); xf (§3.8.45)

Attributes	Description
hidden (Hidden Cell)	A boolean value indicating if the cell is hidden. When the cell is hidden and the sheet on which the cell resides is protected, then the cell value will be displayed in the cell grid location, but the contents of the cell will not be displayed in the formula bar. This is true for all types of cell content, including formula, text, or numbers. Therefore the cell A4 may contain a formula "=SUM(A1:A3)", but if the cell protection property of A4 is marked as hidden, and the sheet is protected, then the cell should display the calculated result (for example, "6"), but will not display the formula used to calculate the result. The possible values for this attribute are defined by the XML Schema boolean datatype.
locked (Cell Locked)	A boolean value indicating if the cell is locked. When cells are marked as "locked" and the sheet is protected, then the options specified in the Sheet Part's sheetProtection element (§3.3.1.81) are prohibited for these cells. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CellProtection">
   <attribute name="locked" type="xsd:boolean" use="optional"/>
   <attribute name="hidden" type="xsd:boolean" use="optional"/>
</complexType>
```

3.8.34 rgbColor (RGB Color)

A single ARGB entry for the corresponding color index.

Parent Elements
indexedColors (§3.8.26)

Attributes	Description
rgb (Alpha Red Green Blue)	Color value expressed in Alpha Red Green Blue format (ARGB). The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§3.18.86).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RgbColor">
   <attribute name="rgb" type="ST_UnsignedIntHex" use="optional"/>
</complexType>
```

3.8.35 right (Right Border)

This element specifies the color and line style for the right border of a cell.

Parent Elements
border (§3.8.4)

Child Elements	Subclause
color (Data Bar Color)	§3.3.1.14

Attributes	Description
style (Line Style)	The line style for this border. The possible values for this attribute are defined by the ST_BorderStyle simple type (§3.18.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BorderPr">
   <sequence>
      <element name="color" type="CT_Color" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="style" type="ST_BorderStyle" use="optional" default="none"/>
</complexType>
```

3.8.36 scheme (Scheme)

Defines the font scheme, if any, to which this font belongs. When a font definition is part of a theme definition, then the font is categorized as either a major or minor font scheme component. When a new theme is chosen, every font that is part of a theme definition is updated to use the new major or minor font definition for that theme. Usually major fonts are used for styles like headings, and minor fonts are used for body & paragraph text.

Parent Elements
font (§3.8.21); rPr (§3.4.7)

Attributes	Description
val (Font Scheme)	Sets font scheme property. The possible values for this attribute are defined by the ST_FontScheme simple type (§3.18.34).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FontScheme">
   <attribute name="val" type="ST_FontScheme" use="required"/>
</complexType>
```

3.8.37 shadow (Shadow)

Macintosh compatibility setting. Represents special word/character rendering on Macintosh, when this flag is set. The effect is to render a shadow behind, beneath and to the right of the text. SpreadsheetML applications are not required to render according to this flag.

Parent Elements
font (§3.8.21); rPr (§3.4.7)

Attributes	Description
val (Value)	A boolean value for the property specified by the parent XML element. If omitted, the default value is `true`. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BooleanProperty">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

3.8.38 stop (Gradient Stop)

One of a sequence of two or more gradient stops, constituting this gradient fill.

Parent Elements
gradientFill (§3.8.23)

Child Elements	Subclause
color (Data Bar Color)	§3.3.1.14

Attributes	Description
position (Gradient Stop Position)	Position information for this gradient stop. Interpreted exactly like gradientFill left, right, bottom, top. The position indicated here indicates the point where the color is pure. Before and and after this position the color can be in transition (or pure, depending on if this is the last stop or not). The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GradientStop">
   <sequence>
      <element name="color" type="CT_Color" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="position" type="xsd:double" use="required"/>
</complexType>
```

3.8.39 styleSheet (Style Sheet)

This is the root element of the Styles part.

Parent Elements
Root element of SpreadsheetML Styles part

Child Elements	Subclause
borders (Borders)	§3.8.5
cellStyles (Cell Styles)	§3.8.8
cellStyleXfs (Formatting Records)	§3.8.9
cellXfs (Cell Formats)	§3.8.10
colors (Colors)	§3.8.11
dxfs (Formats)	§3.8.15
extLst (Future Feature Data Storage Area)	§3.2.10

Child Elements	Subclause
fills (Fills)	§3.8.20
fonts (Fonts)	§3.8.22
numFmts (Number Formats)	§3.8.31
tableStyles (Table Styles)	§3.8.42

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Stylesheet">
   <sequence>
      <element name="numFmts" type="CT_NumFmts" minOccurs="0" maxOccurs="1"/>
      <element name="fonts" type="CT_Fonts" minOccurs="0" maxOccurs="1"/>
      <element name="fills" type="CT_Fills" minOccurs="0" maxOccurs="1"/>
      <element name="borders" type="CT_Borders" minOccurs="0" maxOccurs="1"/>
      <element name="cellStyleXfs" type="CT_CellStyleXfs" minOccurs="0" maxOccurs="1"/>
      <element name="cellXfs" type="CT_CellXfs" minOccurs="0" maxOccurs="1"/>
      <element name="cellStyles" type="CT_CellStyles" minOccurs="0" maxOccurs="1"/>
      <element name="dxfs" type="CT_Dxfs" minOccurs="0" maxOccurs="1"/>
      <element name="tableStyles" type="CT_TableStyles" minOccurs="0" maxOccurs="1"/>
      <element name="colors" type="CT_Colors" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

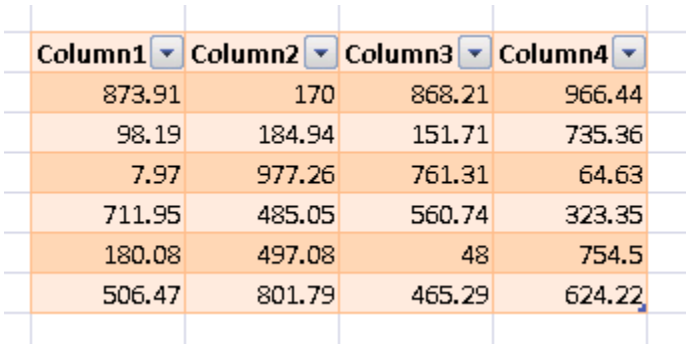
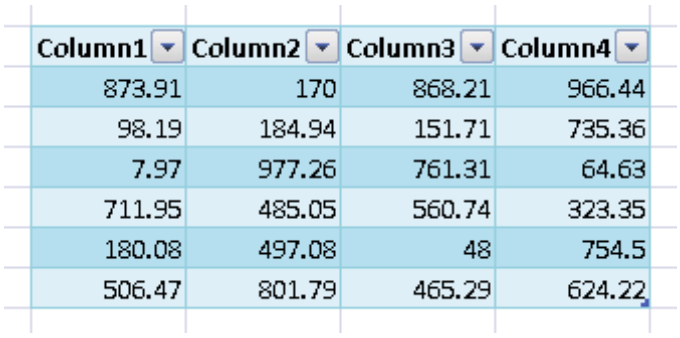
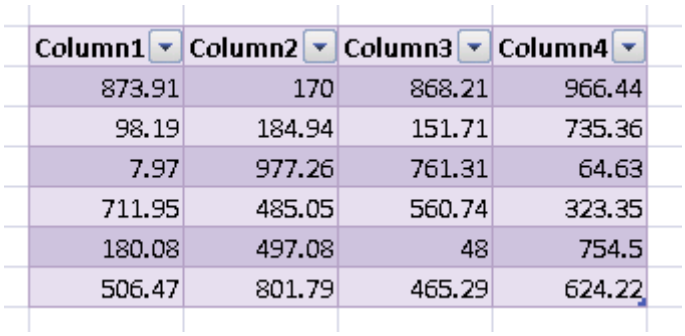
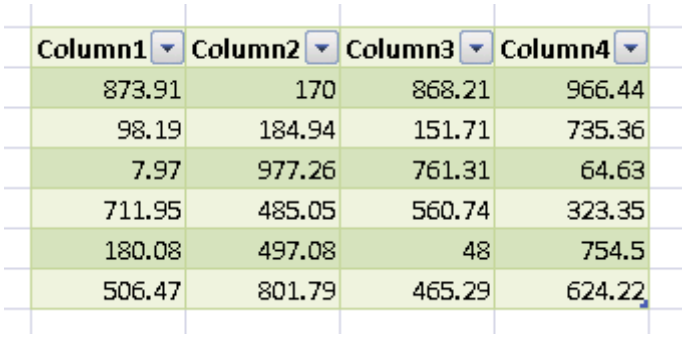
3.8.40 tableStyle (Table Style)

This element represents a single table style definition. The built-in table styles are written in the tableStyle element by name, but the corresponding tableStyleElement elements are assumed rather than explicitly written. Following is a listing of each of the built-in table style definitions, whose normative definition is in Annex D.

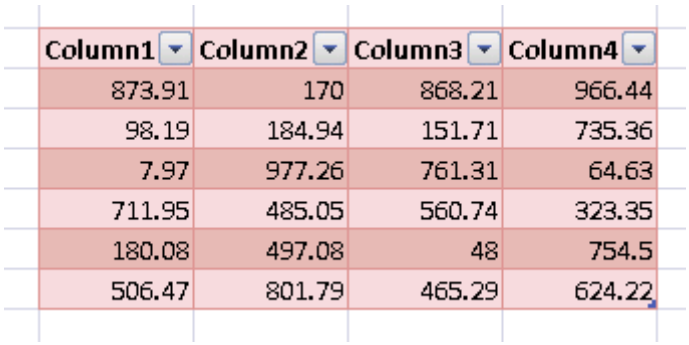
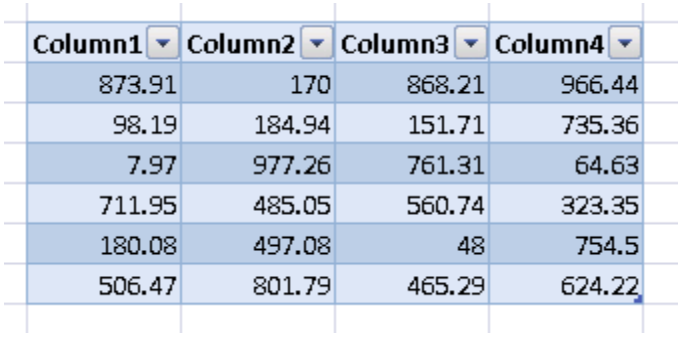
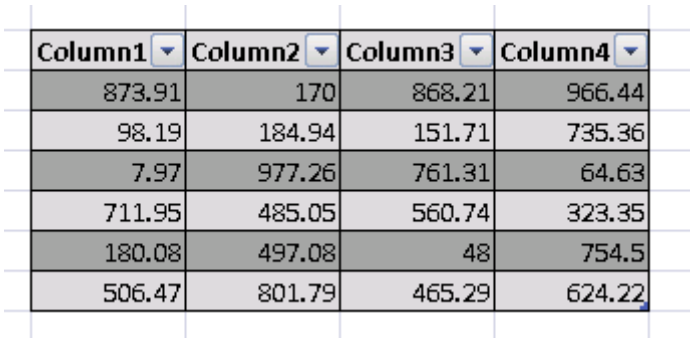
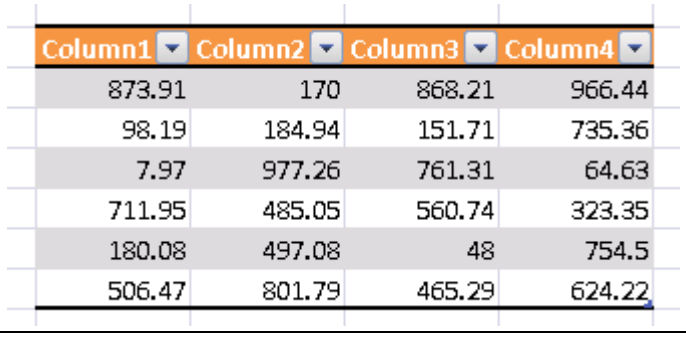
Any of the built-in tables styles, along with any additional table styles explicitly defined shall be supported by applications implementing table styles.

[*Note*: Each of the table styles is made up of a collection of formatting definitions, each of which corresponds to a particular structured region of the table. An application can decide to support these built-in types, and can also decide to define more styles, each with their own definitions. An application can also decide whether the user will be allowed to customize or further define additional table styles. *end note*]

Table Style	[*Example*: (informative)

Table Style	[*Example*: (informative)
TableStyleMedium28	
TableStyleMedium27	
TableStyleMedium26	
TableStyleMedium25	

Column1	Column2	Column3	Column4
873.91	170	868.21	966.44
98.19	184.94	151.71	735.36
7.97	977.26	761.31	64.63
711.95	485.05	560.74	323.35
180.08	497.08	48	754.5
506.47	801.79	465.29	624.22

Table Style	[*Example*: (informative)
TableStyleMedium24	
TableStyleMedium23	
TableStyleMedium22	
TableStyleMedium21	

Column1	Column2	Column3	Column4
873.91	170	868.21	966.44
98.19	184.94	151.71	735.36
7.97	977.26	761.31	64.63
711.95	485.05	560.74	323.35
180.08	497.08	48	754.5
506.47	801.79	465.29	624.22

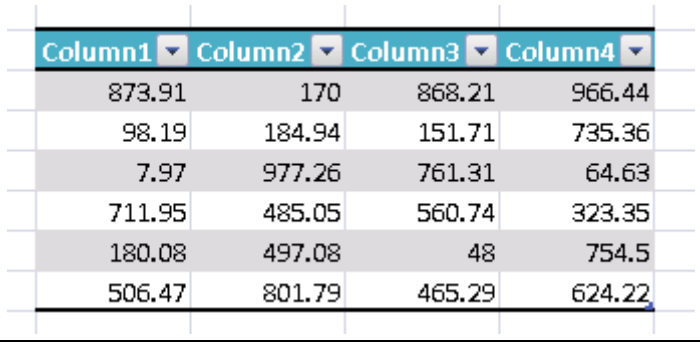
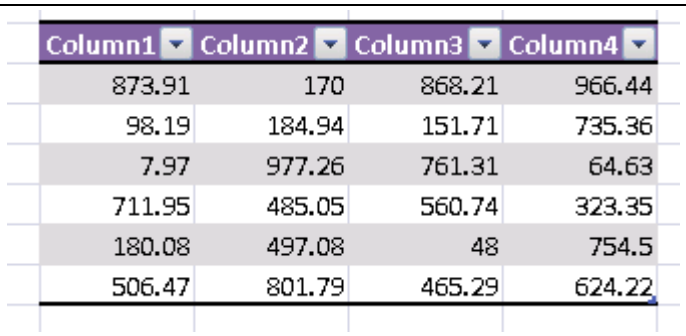
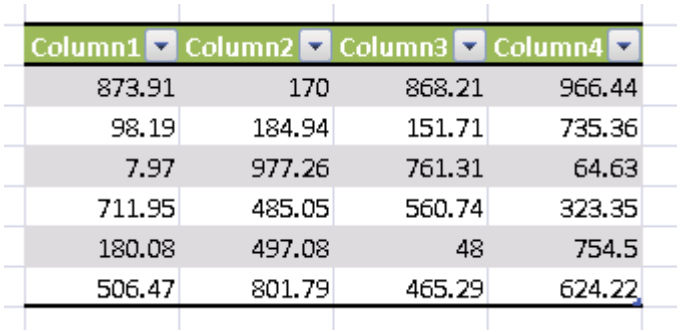
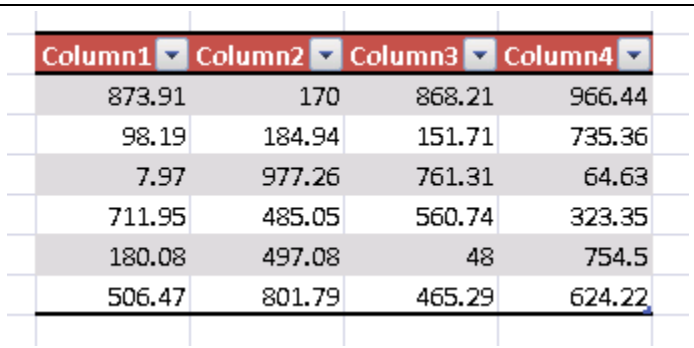
Table Style	[*Example*: (informative)
TableStyleMedium20	
TableStyleMedium19	
TableStyleMedium18	
TableStyleMedium17	

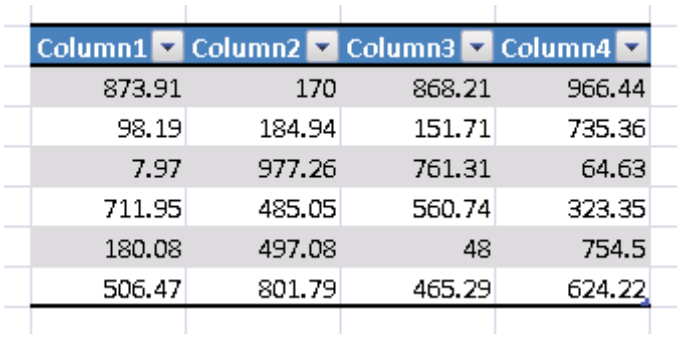
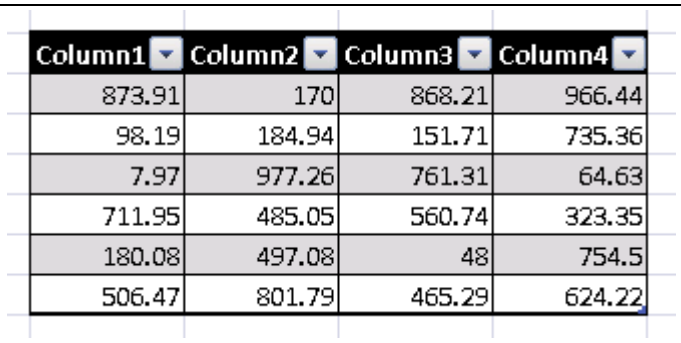
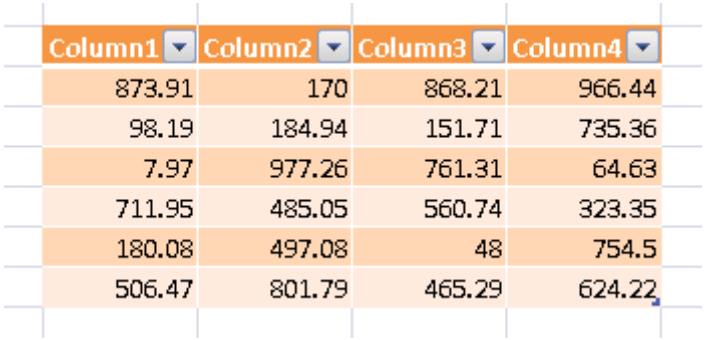
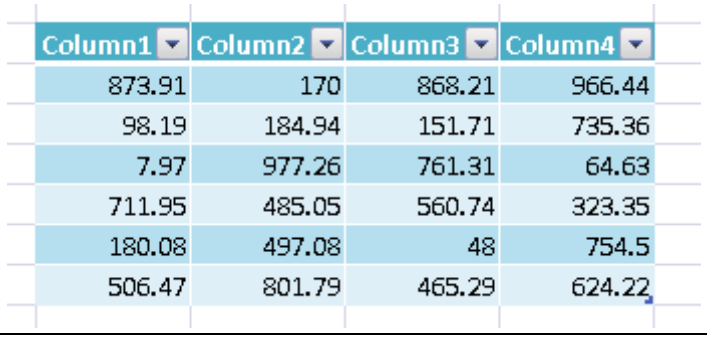
Table Style	[*Example*: (informative)
TableStyleMedium16	
TableStyleMedium15	
TableStyleMedium14	
TableStyleMedium13	

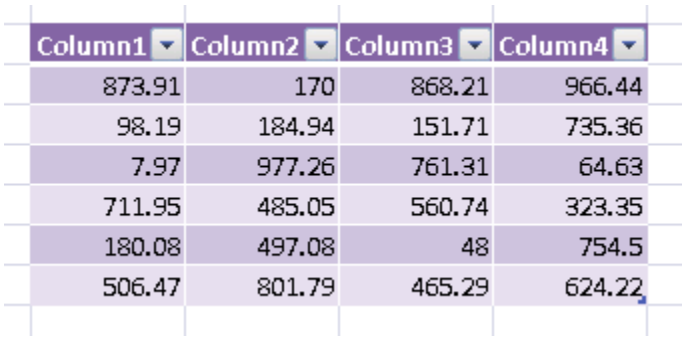
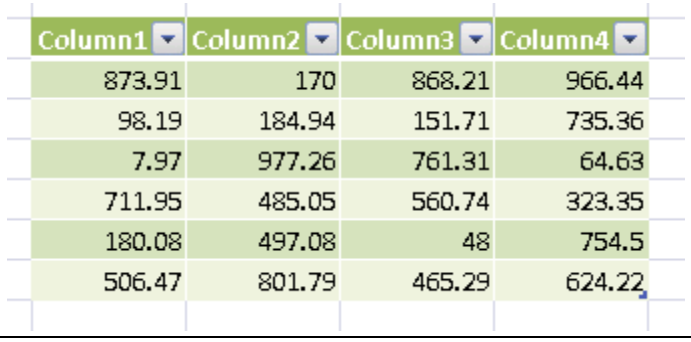
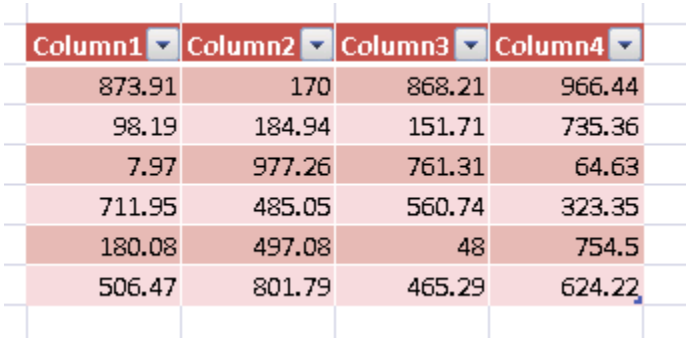
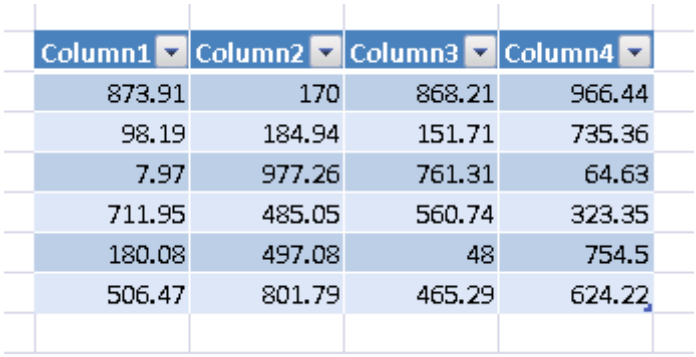
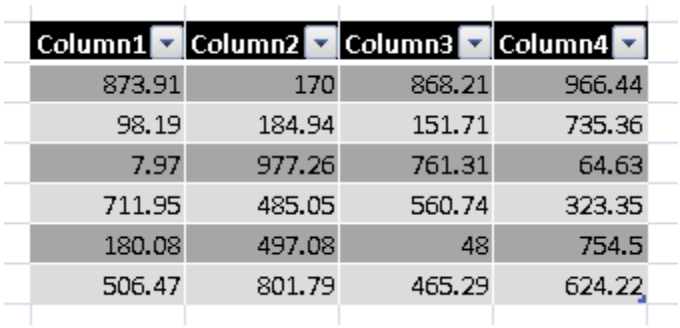
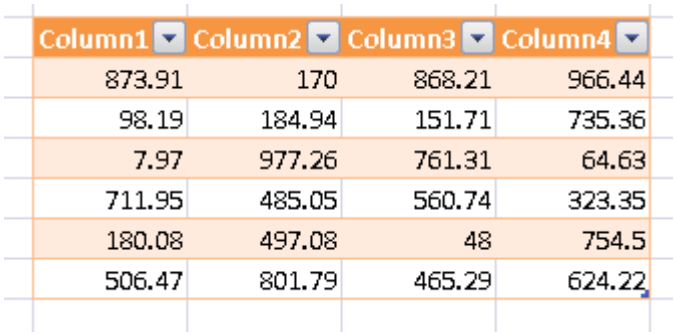
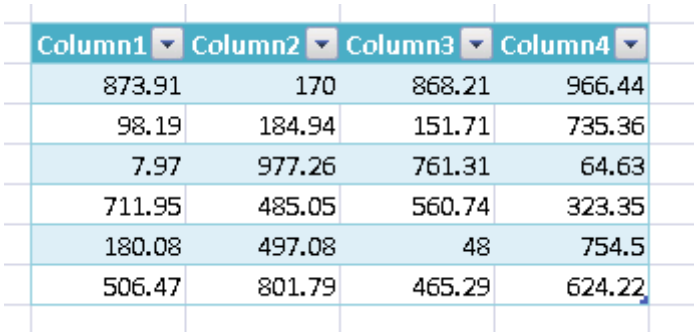
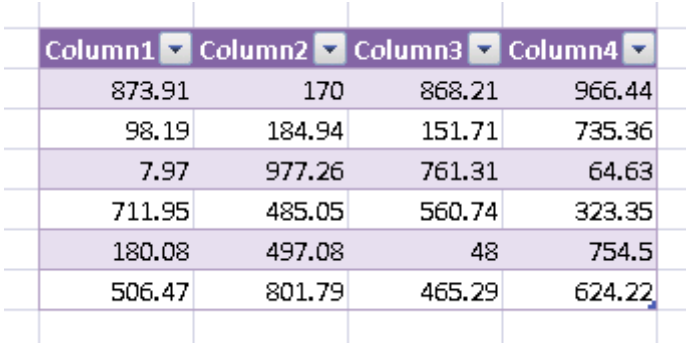
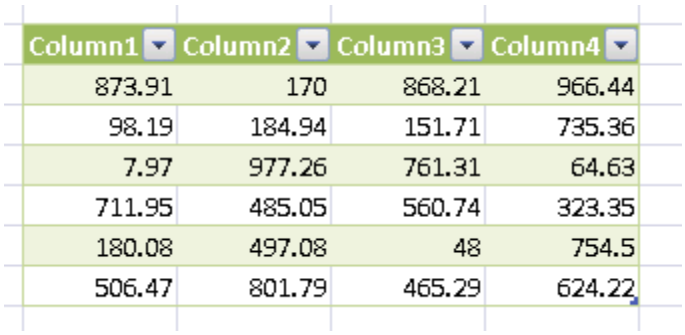
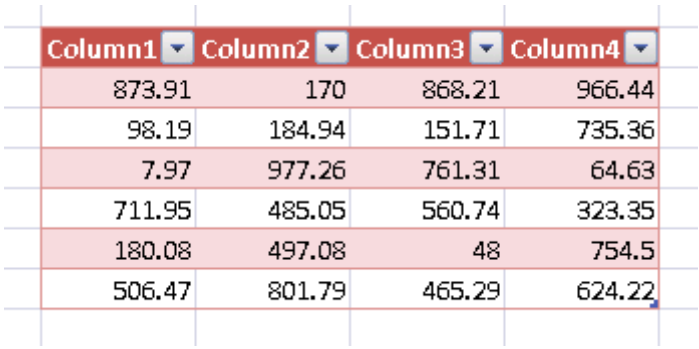
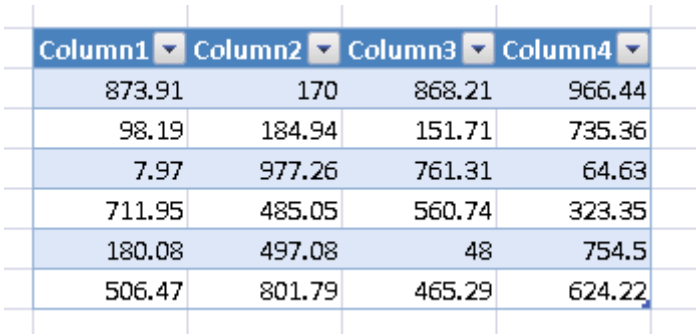
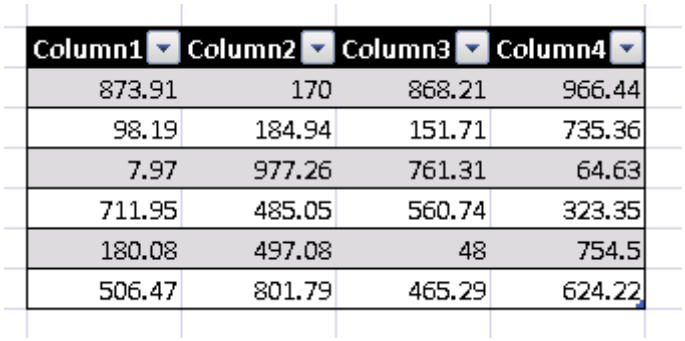
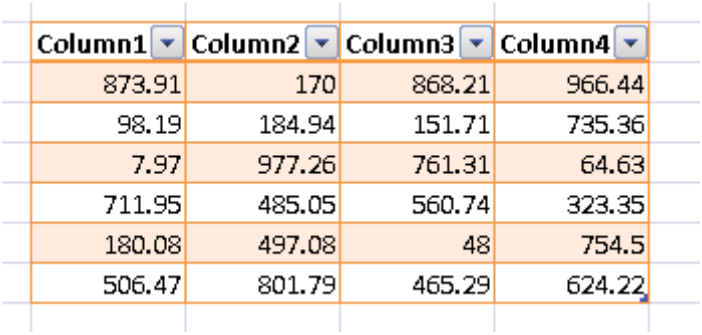
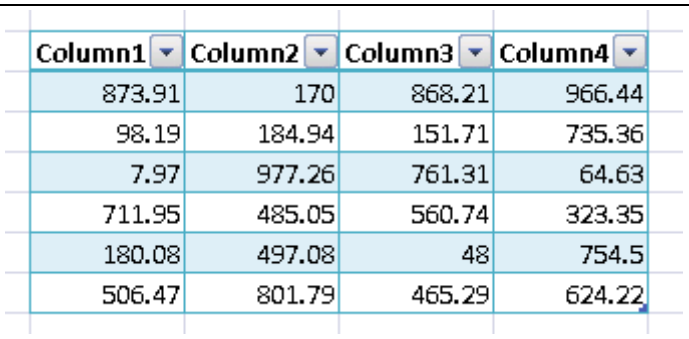
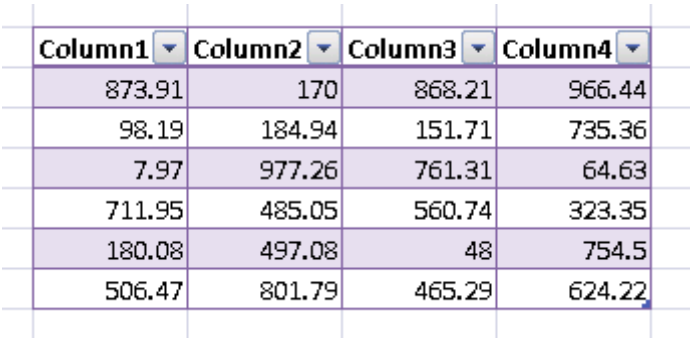
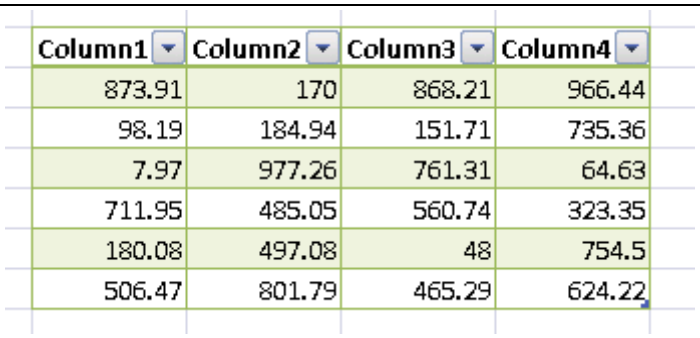
Table Style	[*Example*: (informative)
TableStyleMedium12	
TableStyleMedium11	
TableStyleMedium10	
TableStyleMedium9	

Table Style	[*Example*: (informative)
TableStyleMedium8	
TableStyleMedium7	
TableStyleMedium6	
TableStyleMedium5	

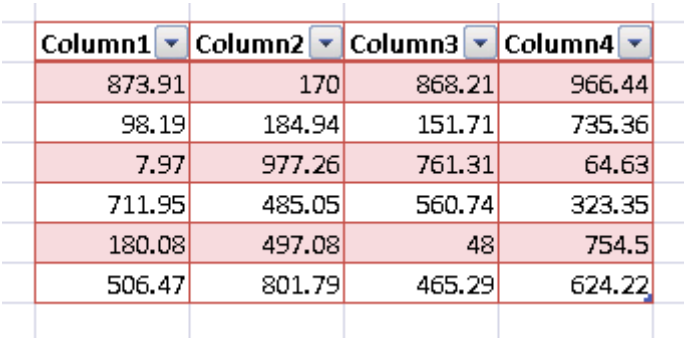
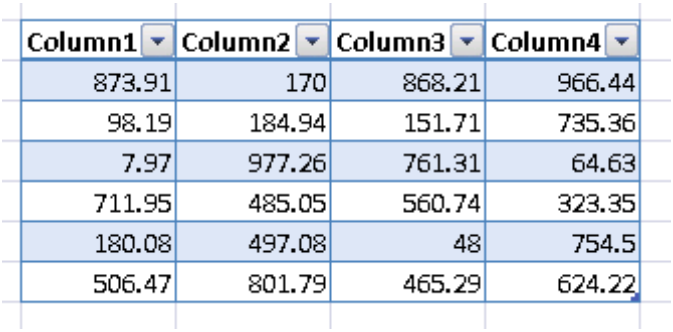
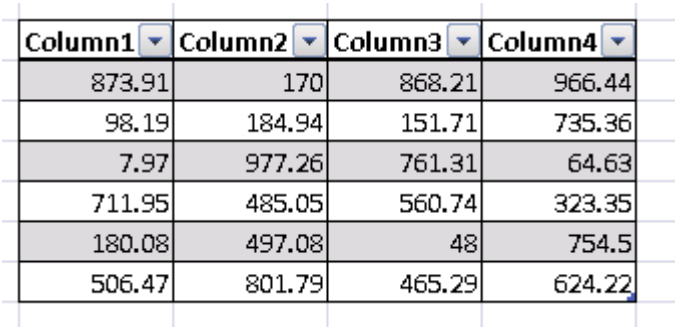
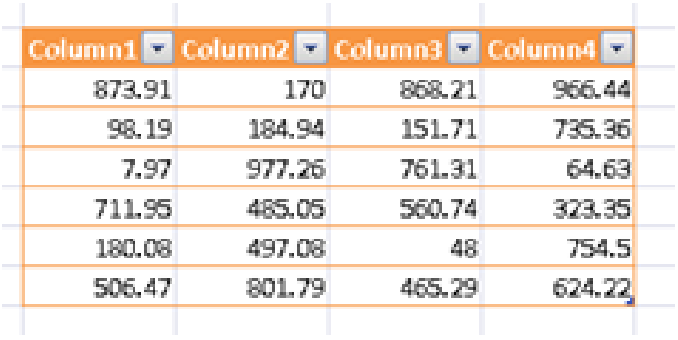
Column1	Column2	Column3	Column4
873.91	170	868.21	966.44
98.19	184.94	151.71	735.36
7.97	977.26	761.31	64.63
711.95	485.05	560.74	323.35
180.08	497.08	48	754.5
506.47	801.79	465.29	624.22

Table Style	[*Example*: (informative)
TableStyleMedium4	
TableStyleMedium3	
TableStyleMedium2	
TableStyleMedium1	

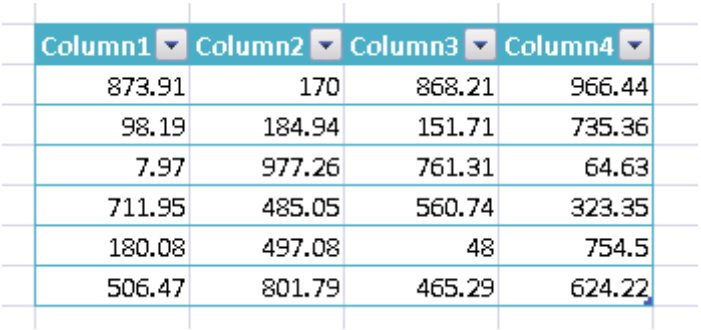
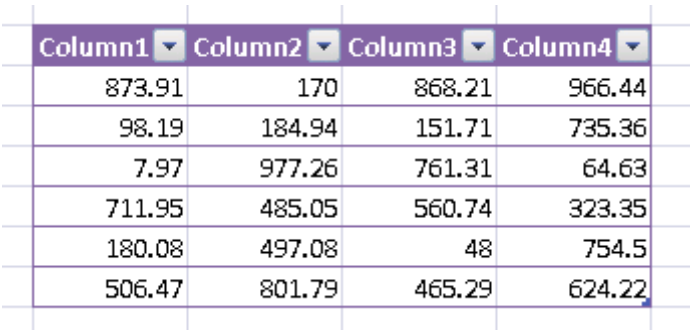
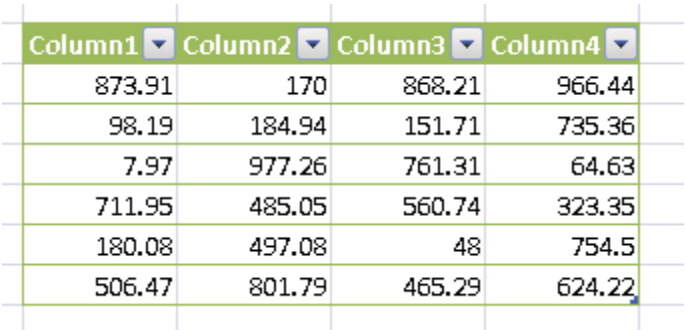
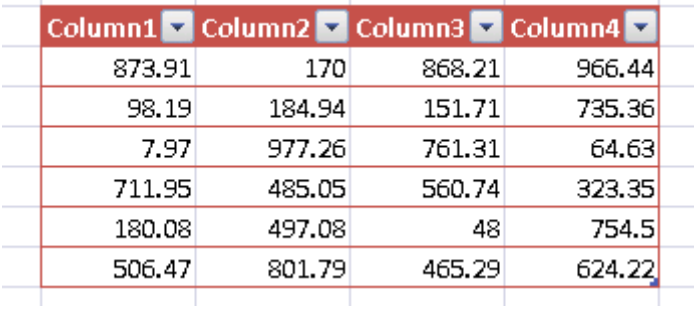
Column1	Column2	Column3	Column4
873.91	170	868.21	966.44
98.19	184.94	151.71	735.36
7.97	977.26	761.31	64.63
711.95	485.05	560.74	323.35
180.08	497.08	48	754.5
506.47	801.79	465.29	624.22

Table Style	[*Example*: (informative)
TableStyleLight21	
TableStyleLight20	
TableStyleLight19	
TableStyleLight18	

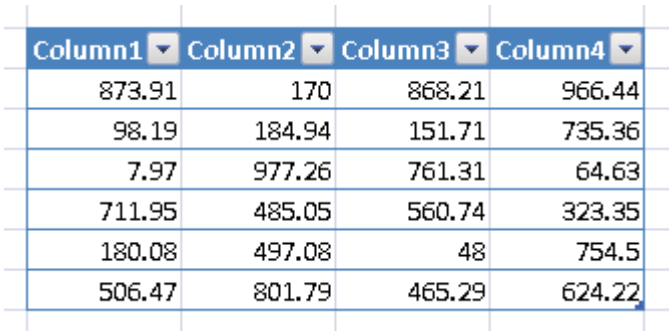
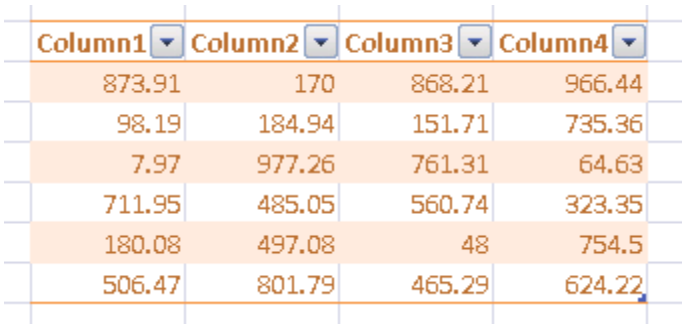
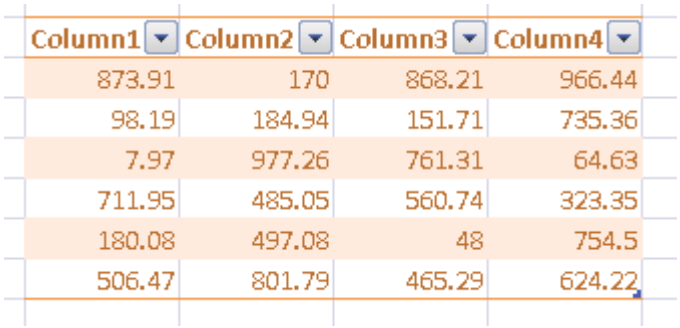
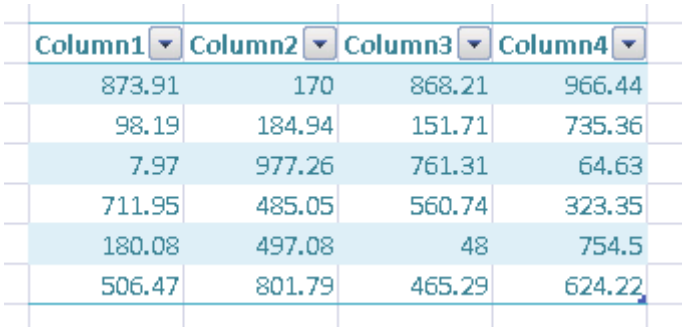
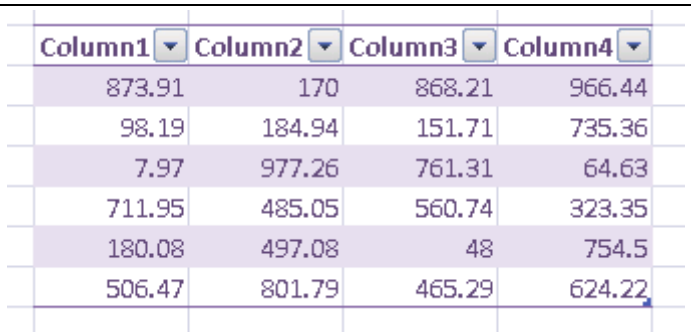
Column1	Column2	Column3	Column4
873.91	170	868.21	966.44
98.19	184.94	151.71	735.36
7.97	977.26	761.31	64.63
711.95	485.05	560.74	323.35
180.08	497.08	48	754.5
506.47	801.79	465.29	624.22

Table Style	[*Example*: (informative)
TableStyleLight17	
TableStyleLight16	
TableStyleLight15	
TableStyleLight14	

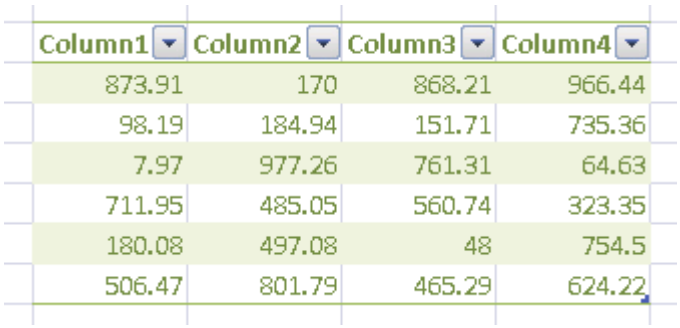
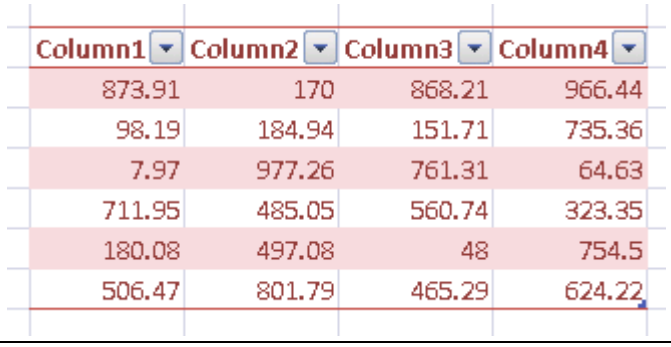
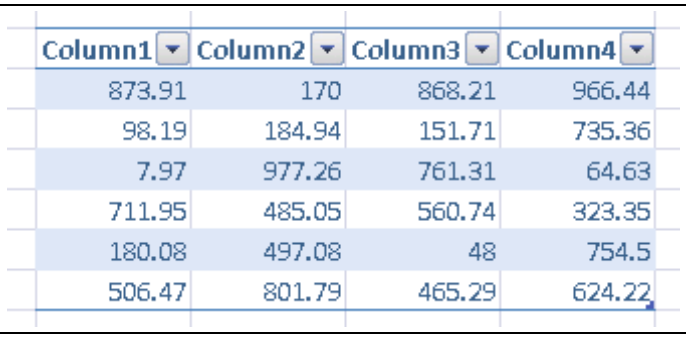
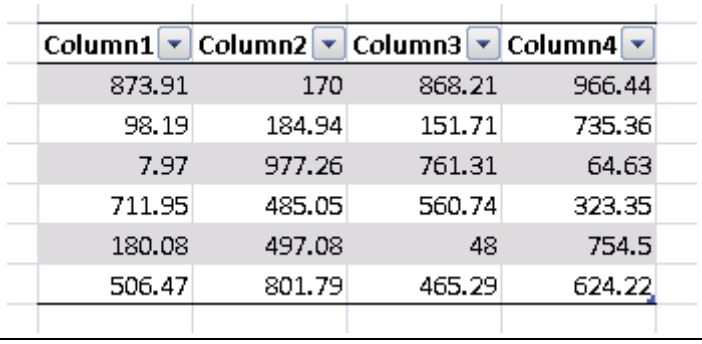
Column1	Column2	Column3	Column4
873.91	170	868.21	966.44
98.19	184.94	151.71	735.36
7.97	977.26	761.31	64.63
711.95	485.05	560.74	323.35
180.08	497.08	48	754.5
506.47	801.79	465.29	624.22

Table Style	[*Example*: (informative)
TableStyleLight13	
TableStyleLight12	
TableStyleLight11	
TableStyleLight10	

Column1	Column2	Column3	Column4
873.91	170	868.21	966.44
98.19	184.94	151.71	735.36
7.97	977.26	761.31	64.63
711.95	485.05	560.74	323.35
180.08	497.08	48	754.5
506.47	801.79	465.29	624.22

Table Style	[*Example*: (informative)
TableStyleLight9	
TableStyleLight8	
TableStyleLight7	
TableStyleLight6	
TableStyleLight5	

Column1	Column2	Column3	Column4
873.91	170	868.21	966.44
98.19	184.94	151.71	735.36
7.97	977.26	761.31	64.63
711.95	485.05	560.74	323.35
180.08	497.08	48	754.5
506.47	801.79	465.29	624.22

Table Style	[*Example*: (informative)
TableStyleLight4	
TableStyleLight3	
TableStyleLight2	
TableStyleLight1	

Column1	Column2	Column3	Column4
873.91	170	868.21	966.44
98.19	184.94	151.71	735.36
7.97	977.26	761.31	64.63
711.95	485.05	560.74	323.35
180.08	497.08	48	754.5
506.47	801.79	465.29	624.22

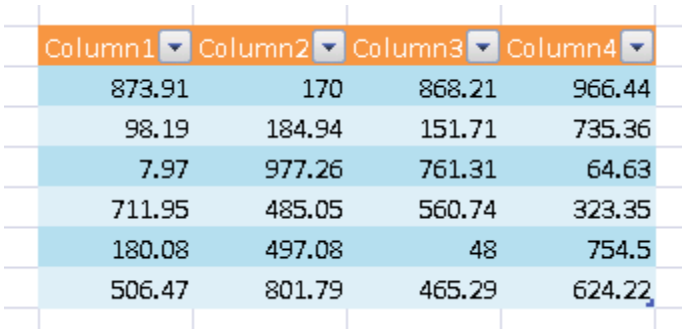
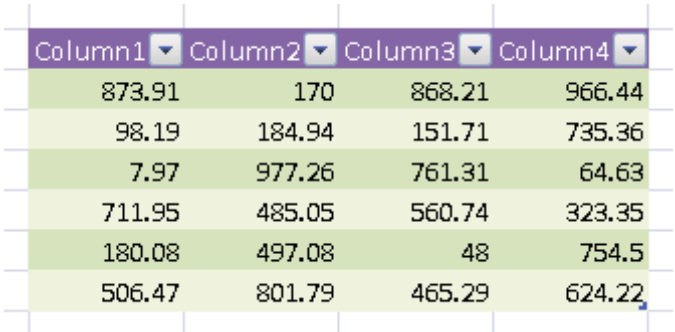
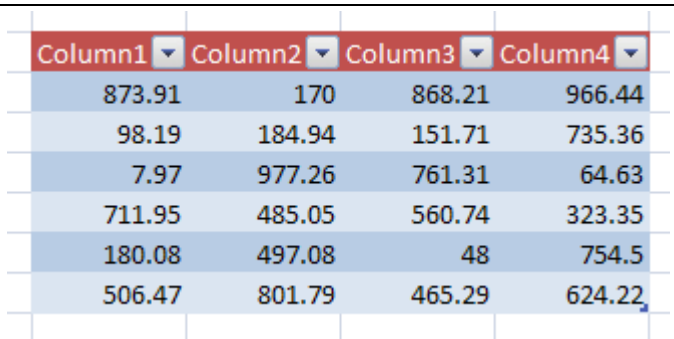
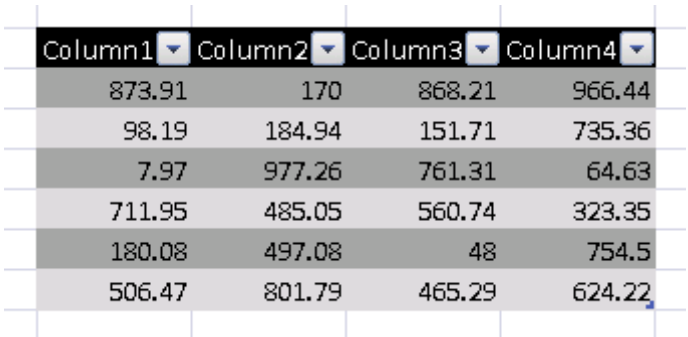
Table Style	[*Example*: (informative)
TableStyleDark11	
TableStyleDark10	
TableStyleDark9	
ableStyleDark8	

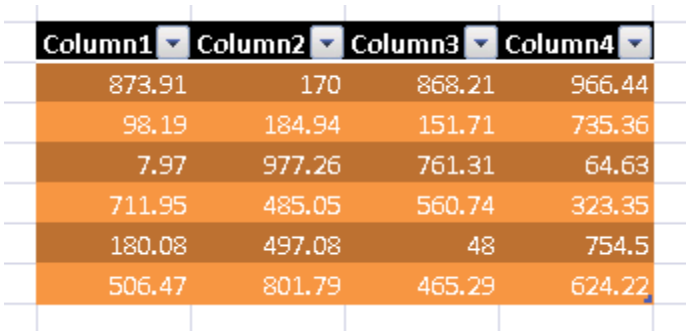
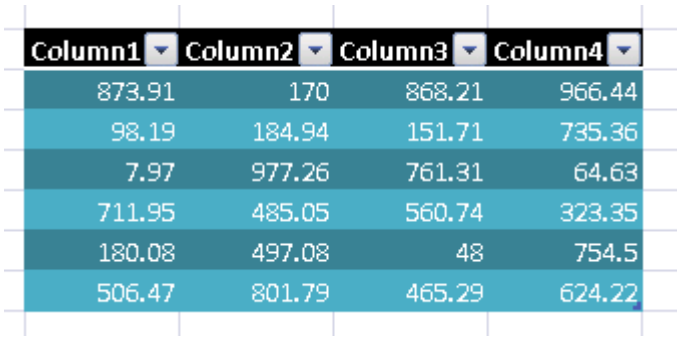
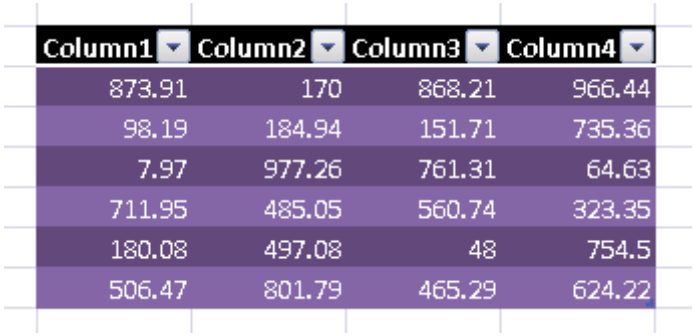
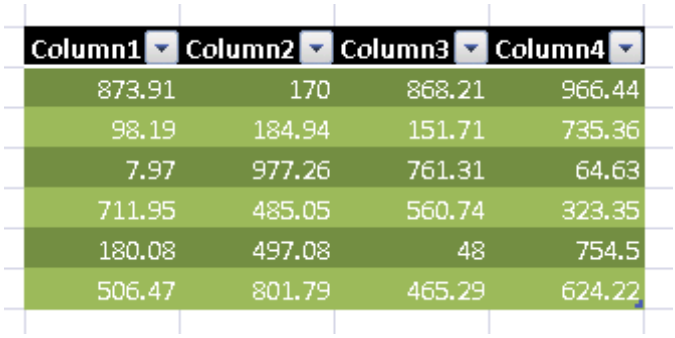
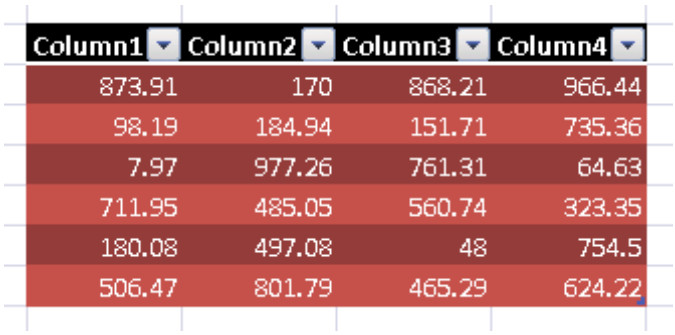
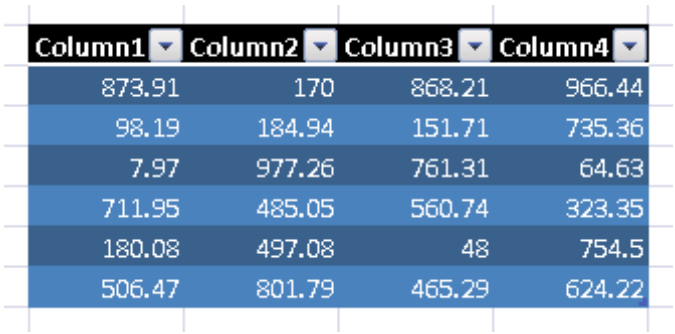
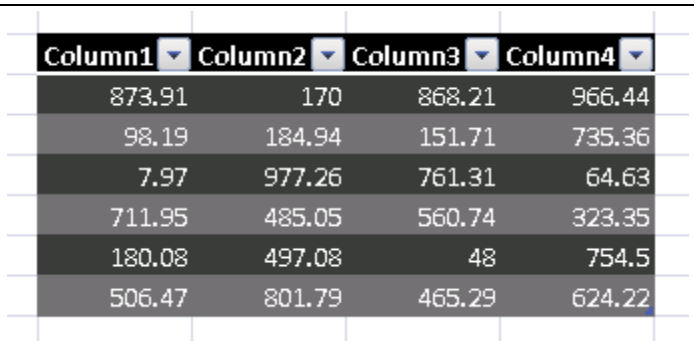
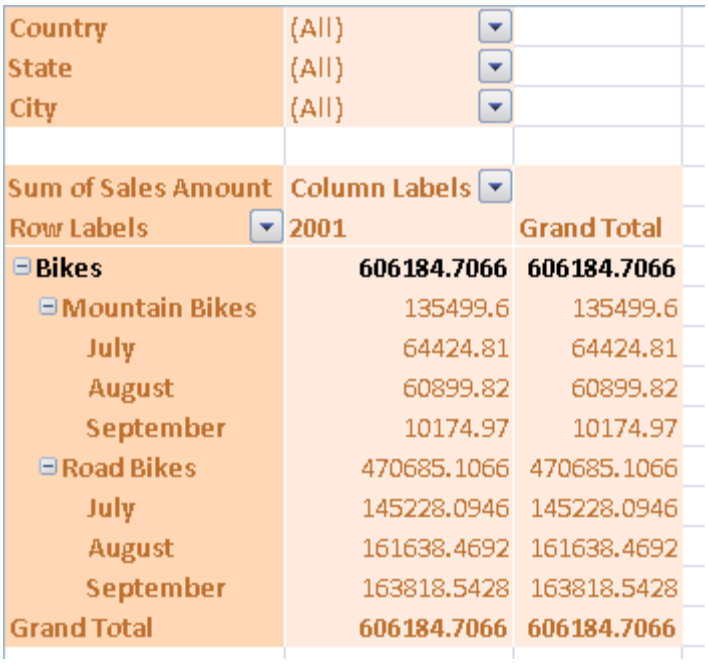
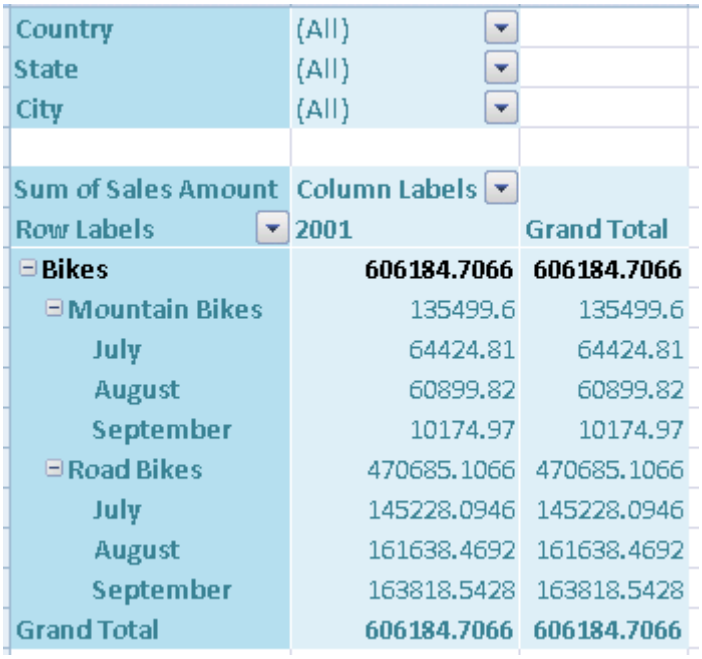
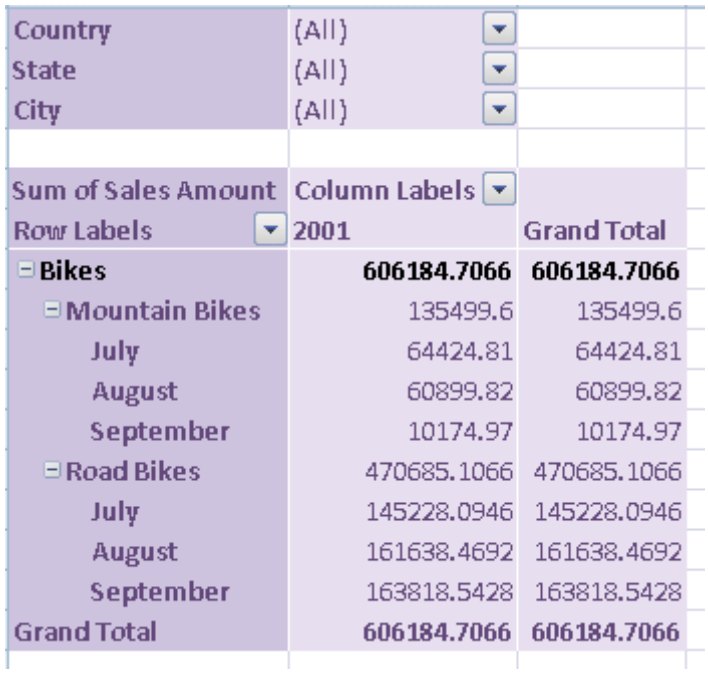
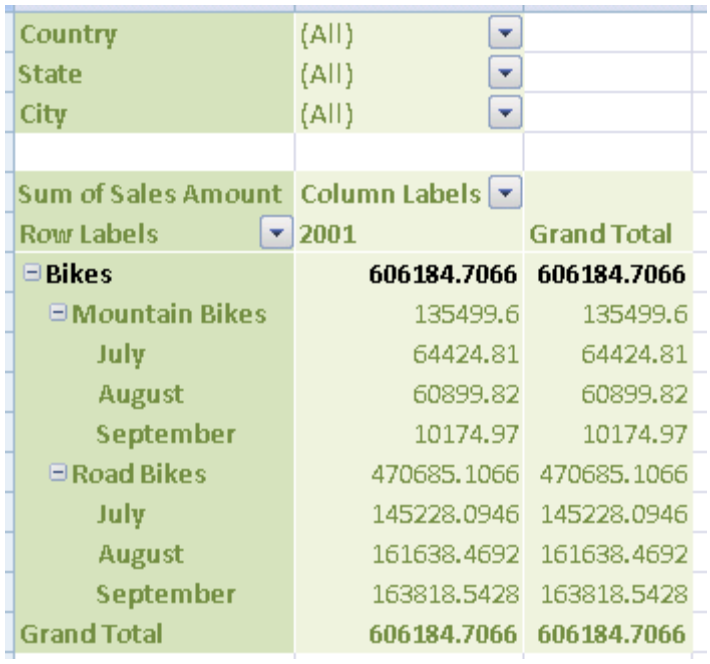
Table Style	[*Example*: (informative)
TableStyleDark7	
TableStyleDark6	
TableStyleDark5	
TableStyleDark4	

Table Style	[*Example*: (informative)
TableStyleDark3	
TableStyleDark2	
TableStyleDark1	

PivotTable Style	[*Example*: (informative)

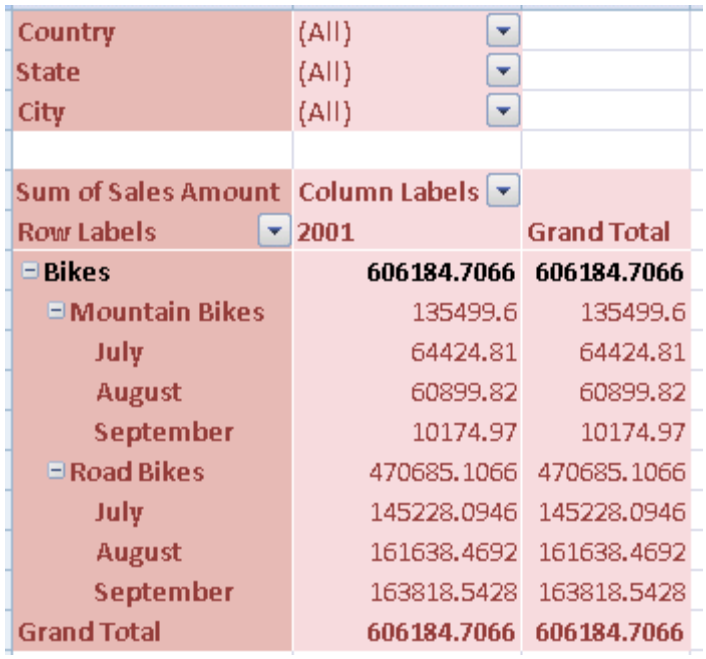
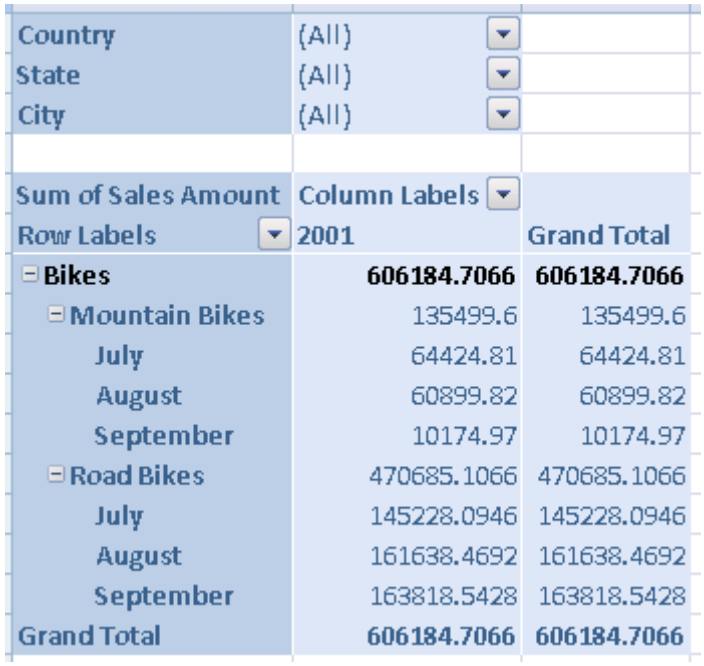
PivotTable Style	[*Example*: (informative)
PivotStyleMedium28	
PivotStyleMedium27	

Country	(All)	
State	(All)	
City	(All)	
Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	**606184.7066**	**606184.7066**
Mountain Bikes	135499.6	135499.6
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	470685.1066	470685.1066
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	**606184.7066**	**606184.7066**

PivotTable Style	[*Example*: (informative)
PivotStyleMedium26	
PivotStyleMedium25	

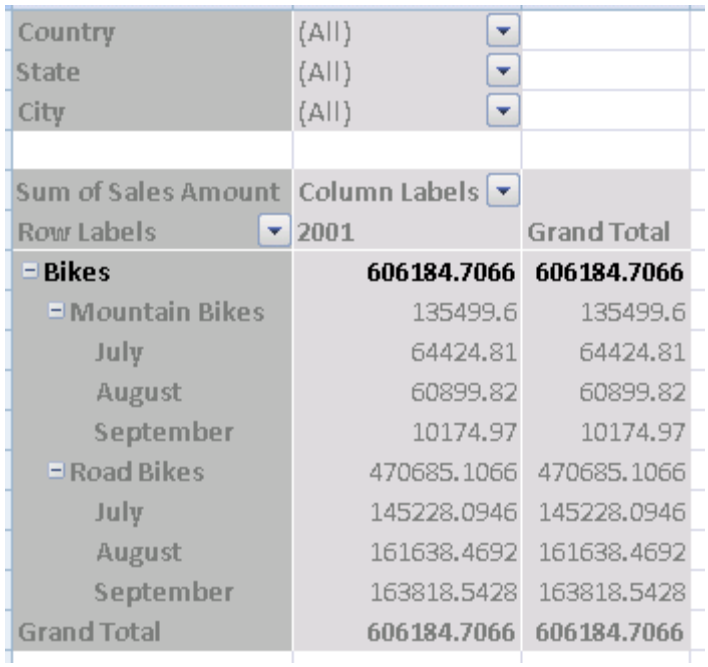
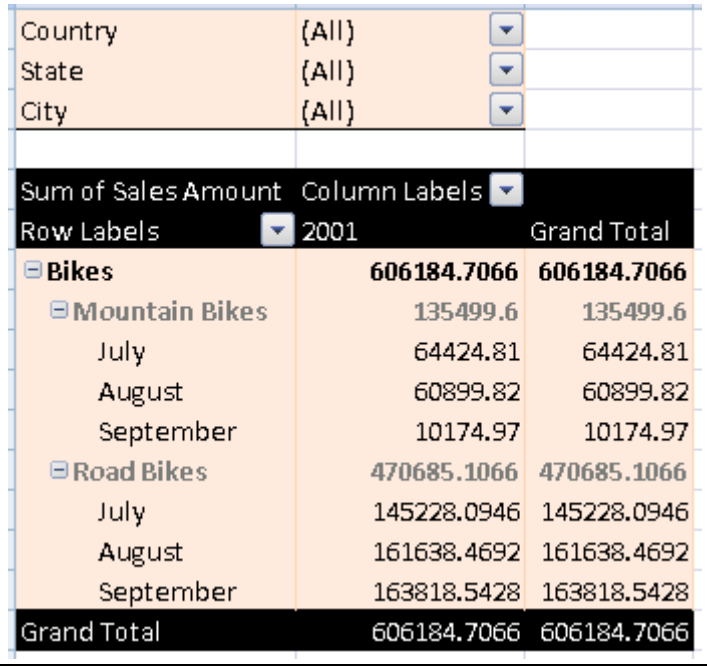
Country	(All)	
State	(All)	
City	(All)	
Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	**606184.7066**	**606184.7066**
Mountain Bikes	135499.6	135499.6
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	470685.1066	470685.1066
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	**606184.7066**	**606184.7066**

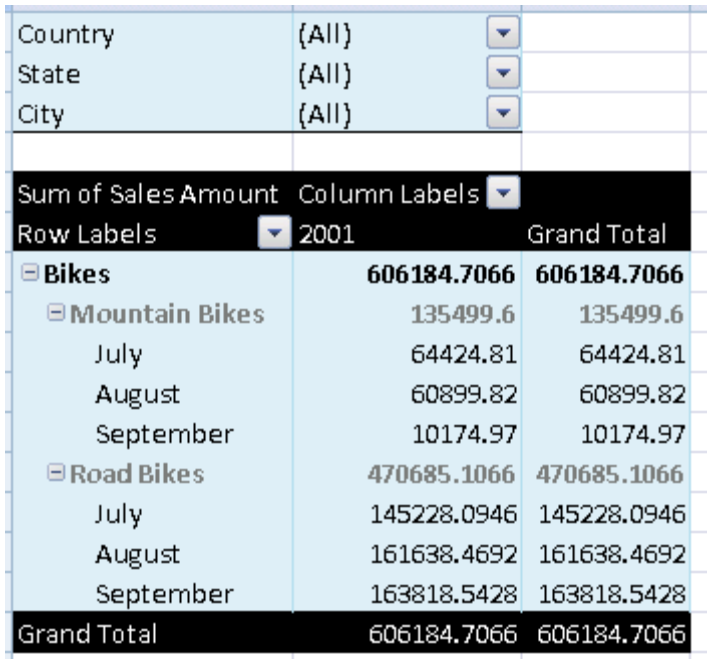
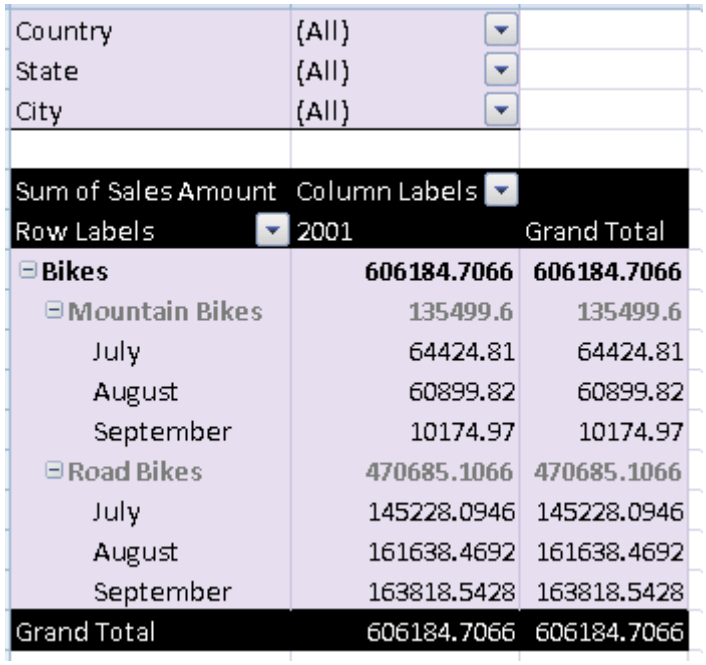
Country	(All)	
State	(All)	
City	(All)	
Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	**606184.7066**	**606184.7066**
Mountain Bikes	135499.6	135499.6
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	470685.1066	470685.1066
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	**606184.7066**	**606184.7066**

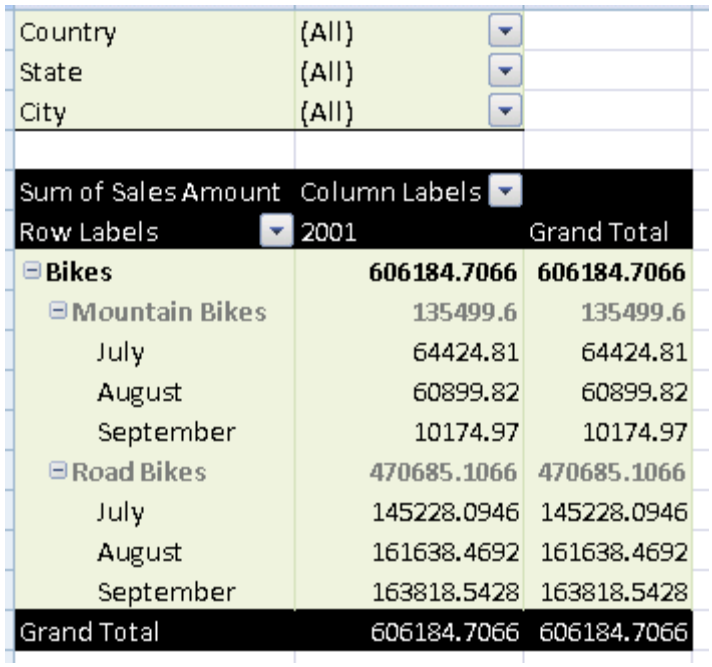
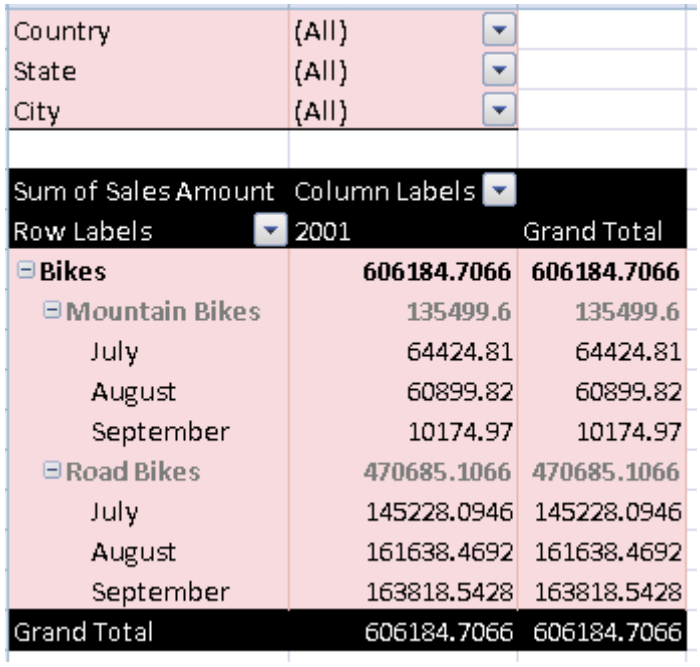
PivotTable Style	[*Example*: (informative)
PivotStyleMedium24	
PivotStyleMedium23	

Country	(All)	
State	(All)	
City	(All)	
Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	**606184.7066**	**606184.7066**
Mountain Bikes	135499.6	135499.6
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	470685.1066	470685.1066
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	**606184.7066**	**606184.7066**

Country	(All)	
State	(All)	
City	(All)	
Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	**606184.7066**	**606184.7066**
Mountain Bikes	135499.6	135499.6
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	470685.1066	470685.1066
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	**606184.7066**	**606184.7066**

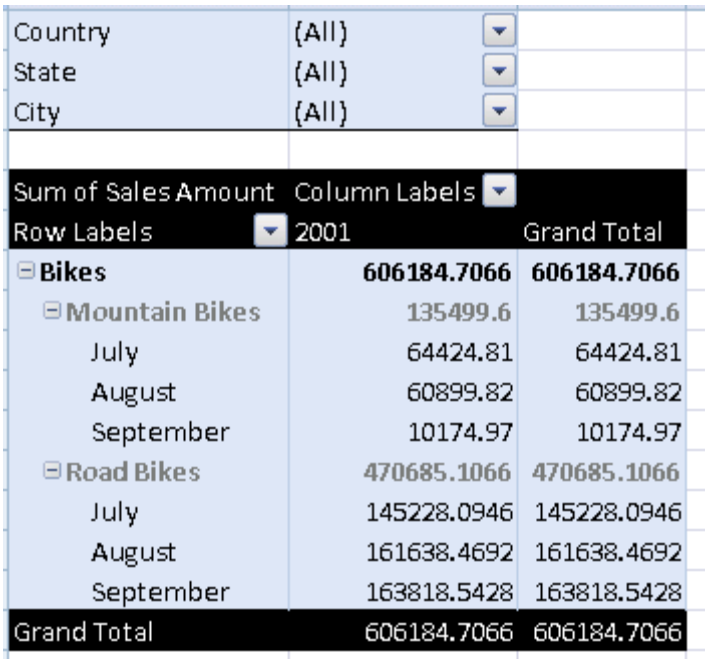
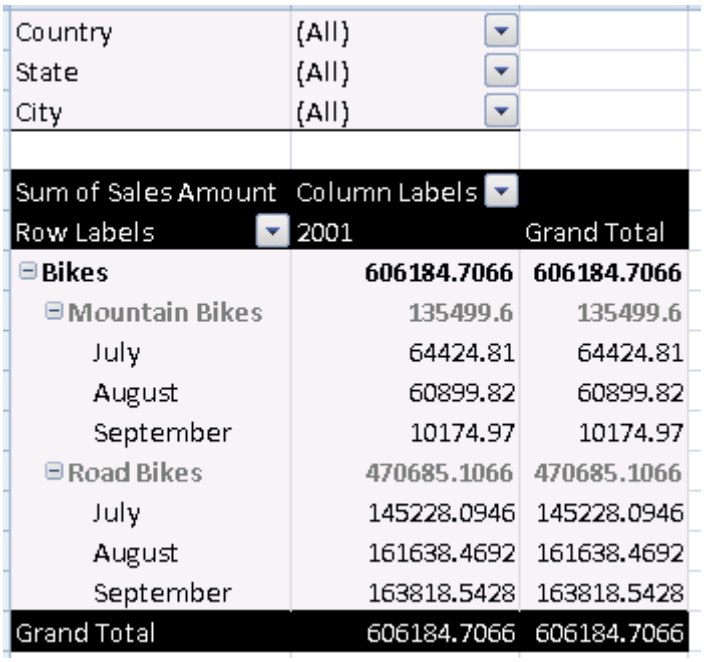
PivotTable Style	[*Example*: (informative)
PivotStyleMedium22	
PivotStyleMedium21	

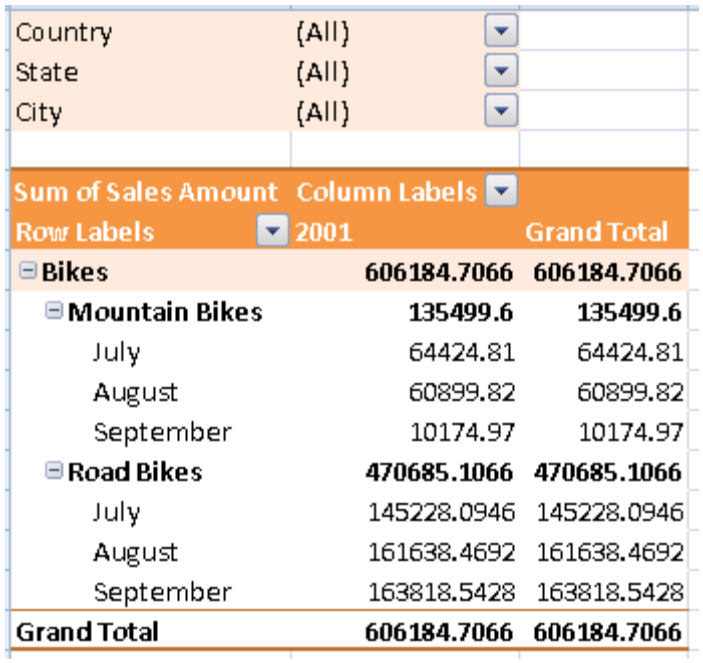
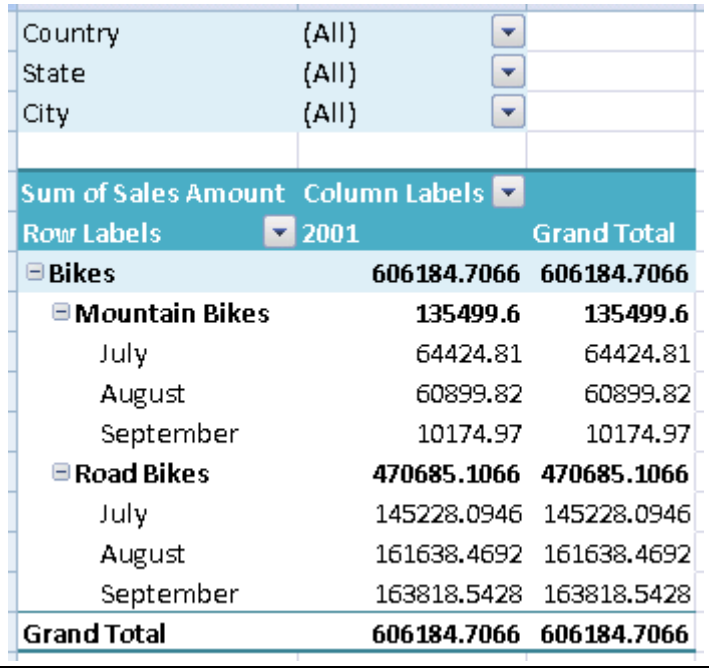
PivotTable Style	[*Example*: (informative)
PivotStyleMedium20	
PivotStyleMedium19	

PivotTable Style	[*Example*: (informative)
PivotStyleMedium18	
PivotStyleMedium17	

Country (All)
State (All)
City (All)

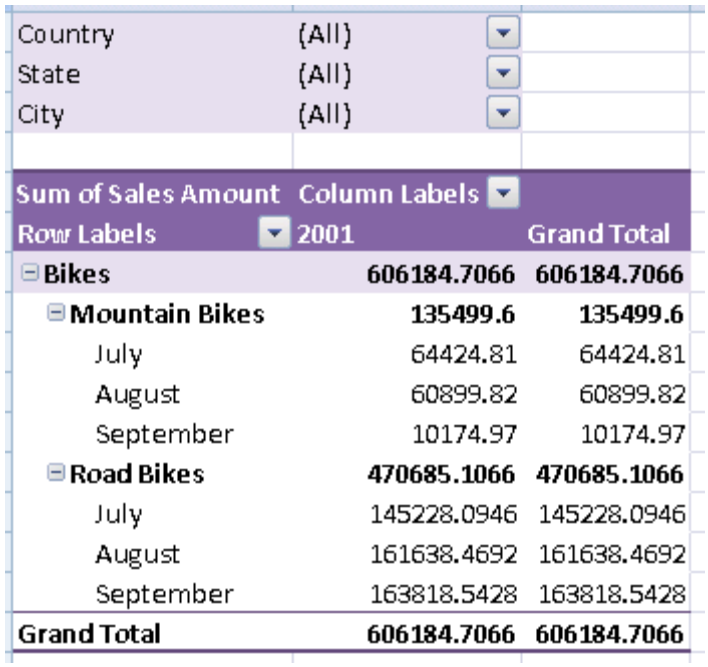
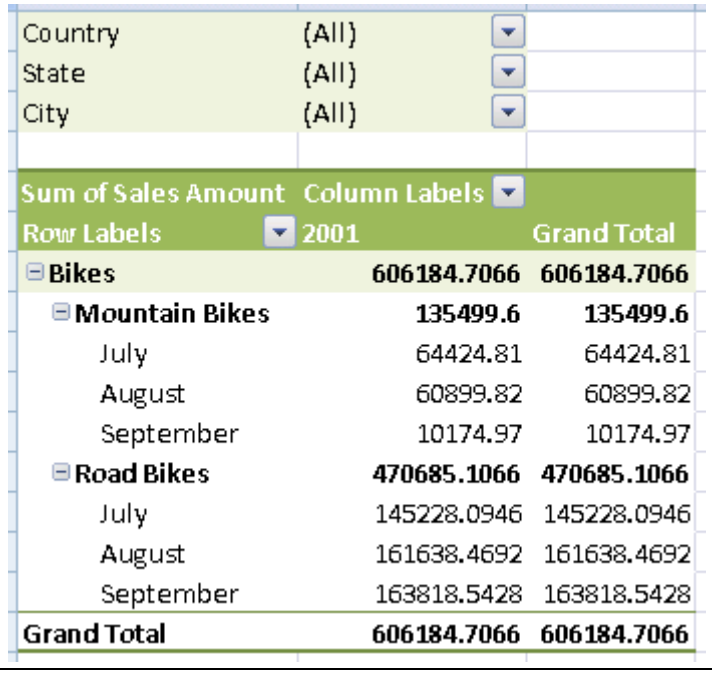
Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	**606184.7066**	**606184.7066**
Mountain Bikes	135499.6	135499.6
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	470685.1066	470685.1066
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	606184.7066	606184.7066

PivotTable Style	[*Example*: (informative)
PivotStyleMedium16	
PivotStyleMedium15	

PivotTable Style	[*Example*: (informative)
PivotStyleMedium14	
PivotStyleMedium13	

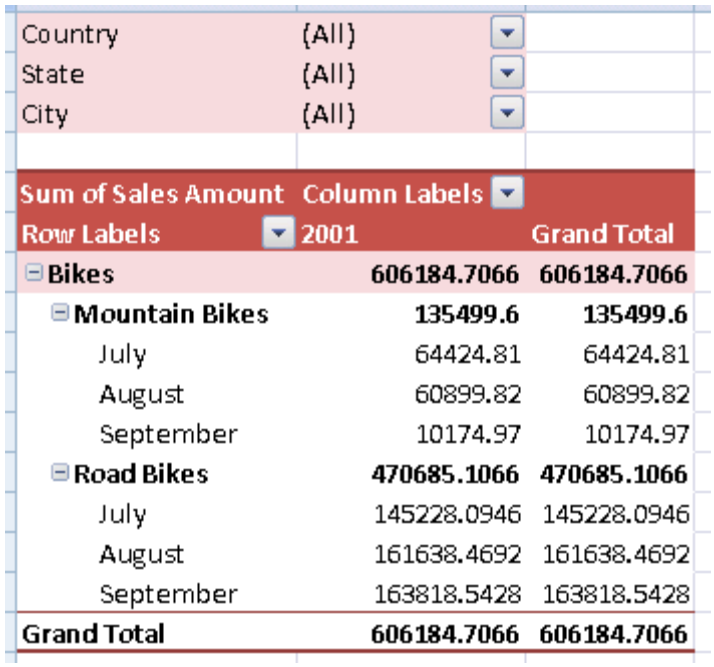
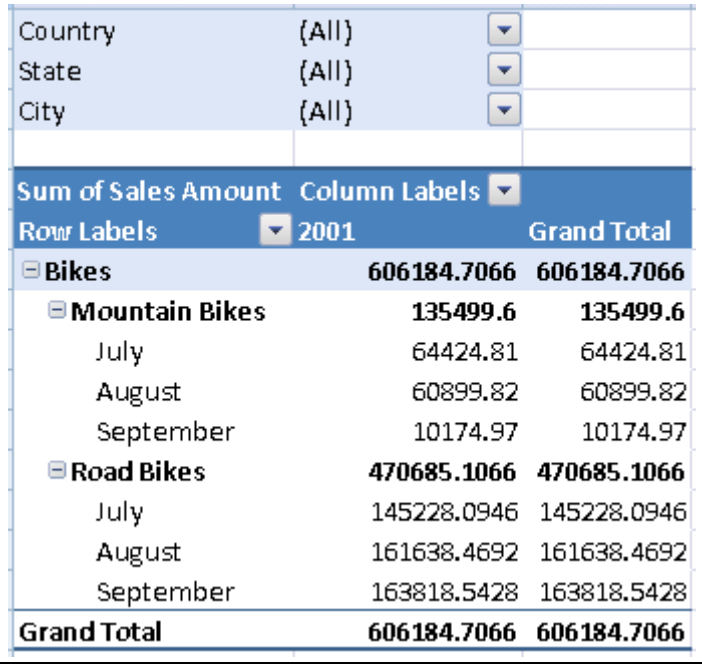
Country	(All)
State	(All)
City	(All)

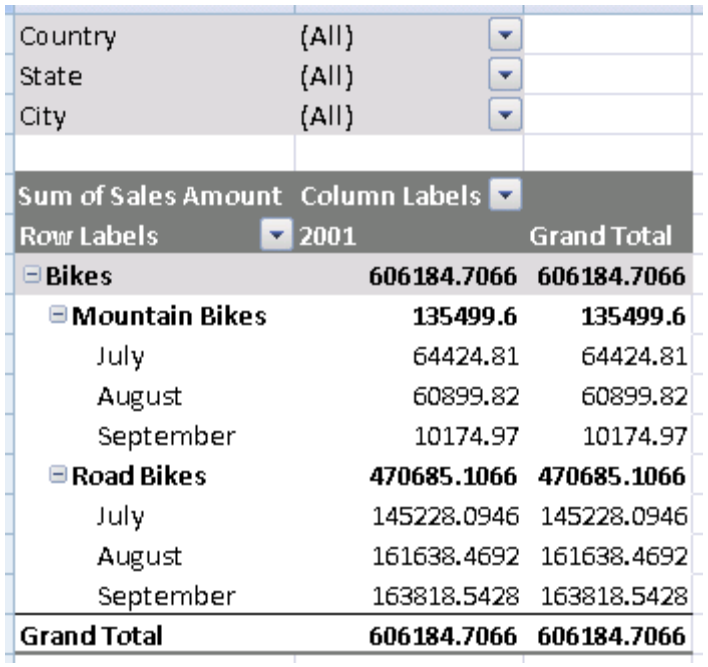
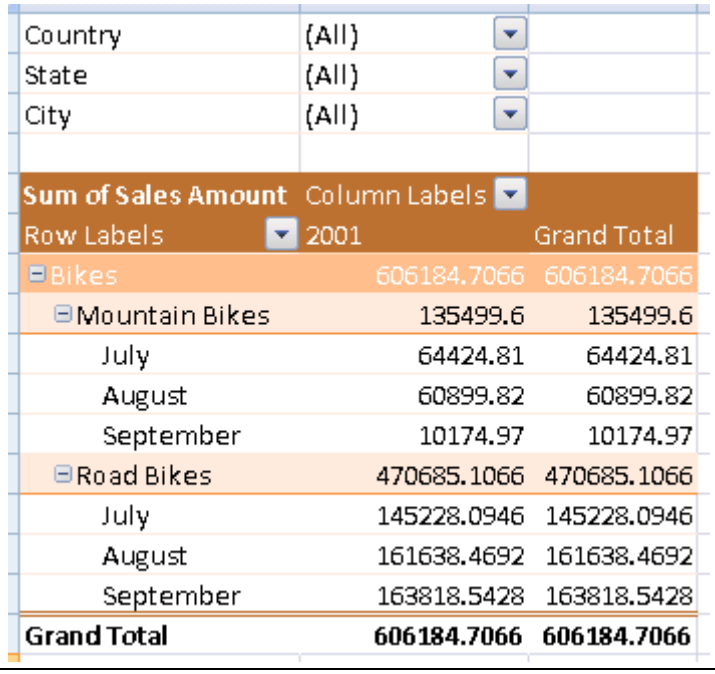
Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	**606184.7066**	**606184.7066**
Mountain Bikes	**135499.6**	**135499.6**
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	**470685.1066**	**470685.1066**
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	**606184.7066**	**606184.7066**

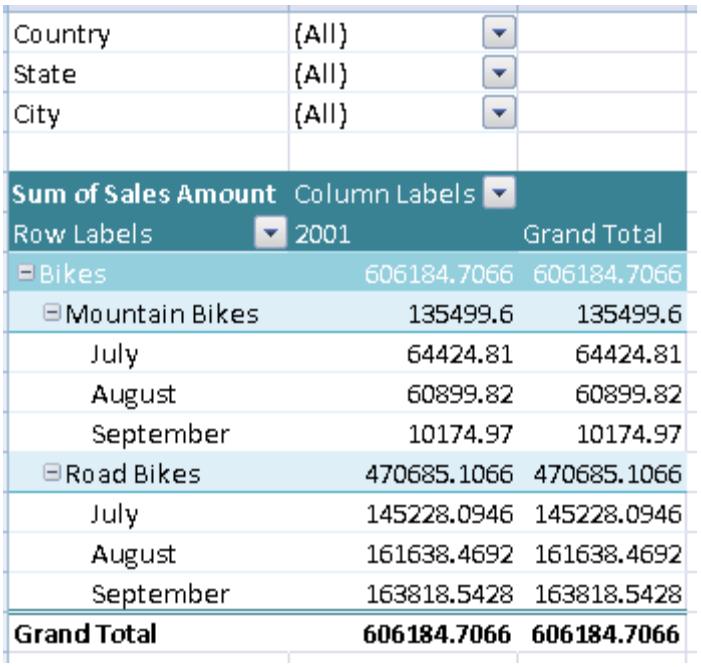
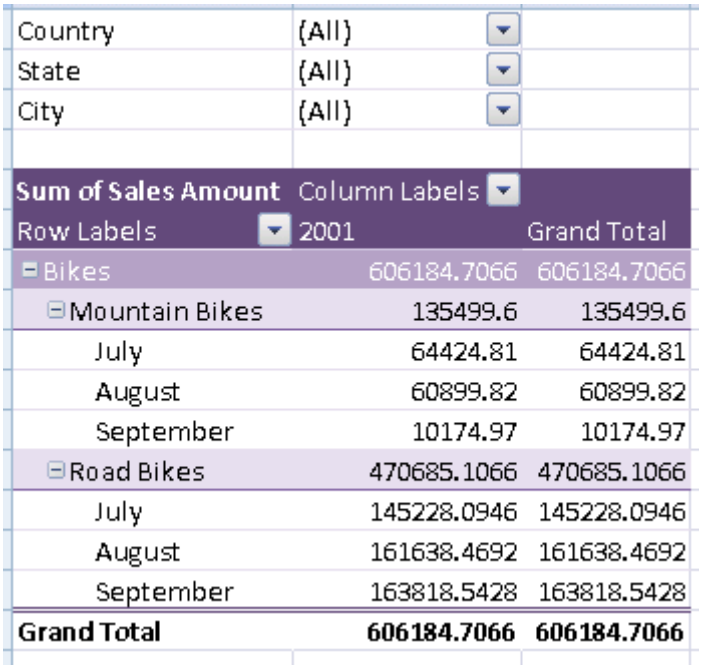
PivotTable Style	[*Example*: (informative)
PivotStyleMedium12	
PivotStyleMedium11	

Country (All)
State (All)
City (All)

Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	**606184.7066**	**606184.7066**
Mountain Bikes	**135499.6**	**135499.6**
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	**470685.1066**	**470685.1066**
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	**606184.7066**	**606184.7066**

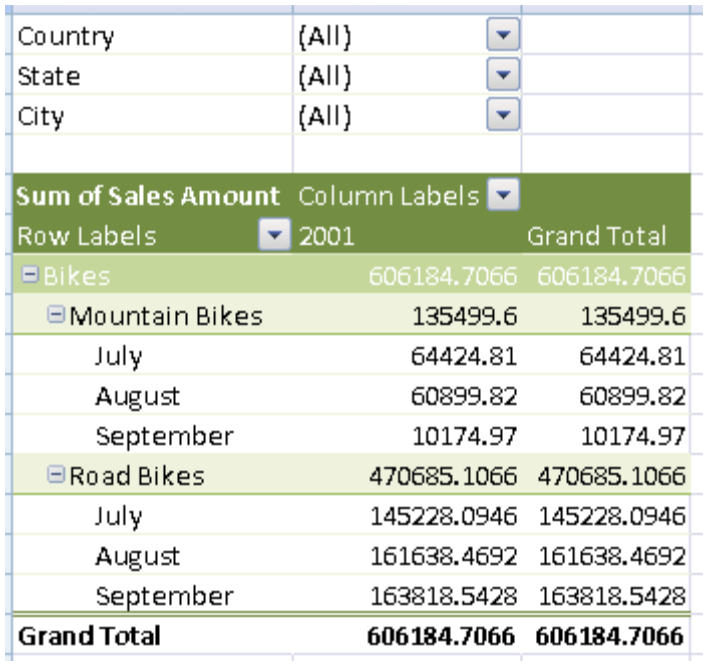
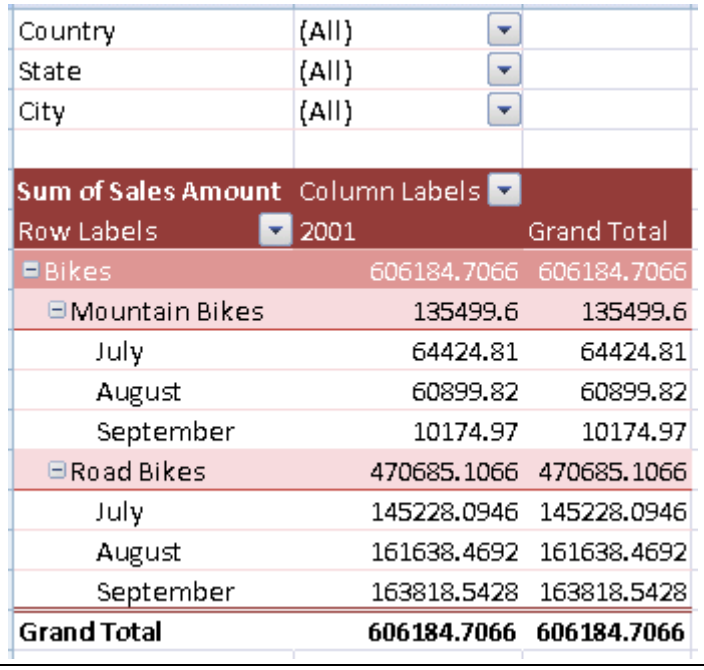
PivotTable Style	[*Example*: (informative)
PivotStyleMedium10	
PivotStyleMedium9	

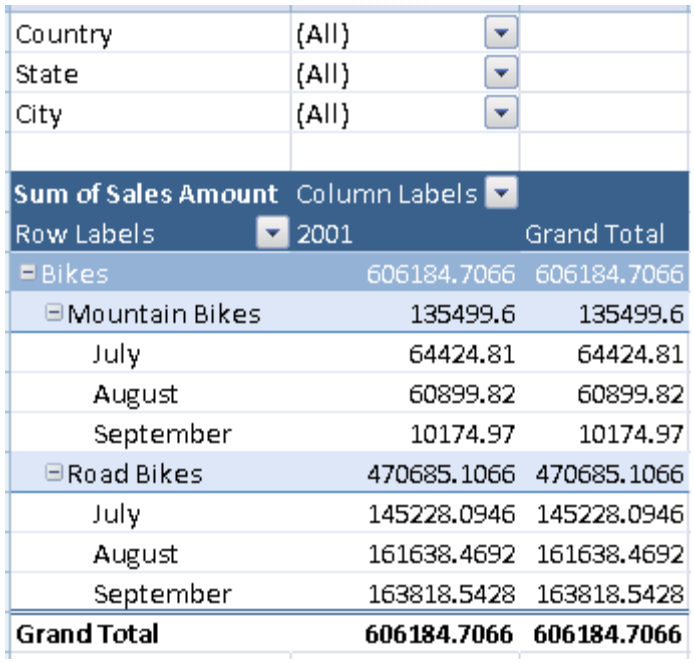
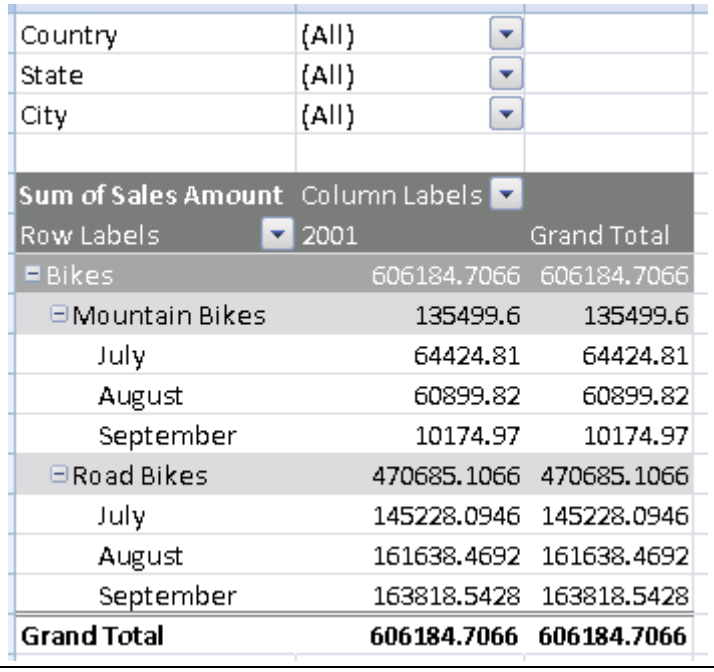
PivotTable Style	[*Example*: (informative)
PivotStyleMedium8	
PivotStyleMedium7	

PivotTable Style	[*Example*: (informative)
PivotStyleMedium6	
PivotStyleMedium5	

Country (All)
State (All)
City (All)

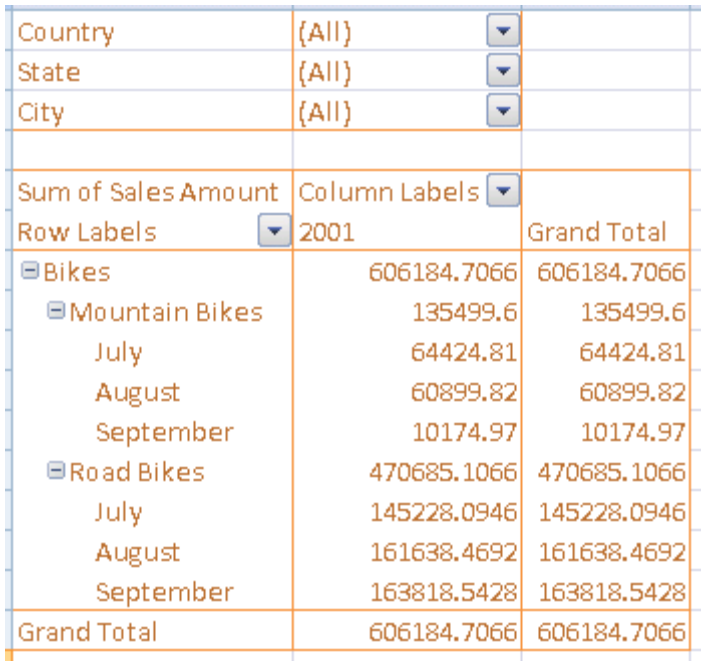
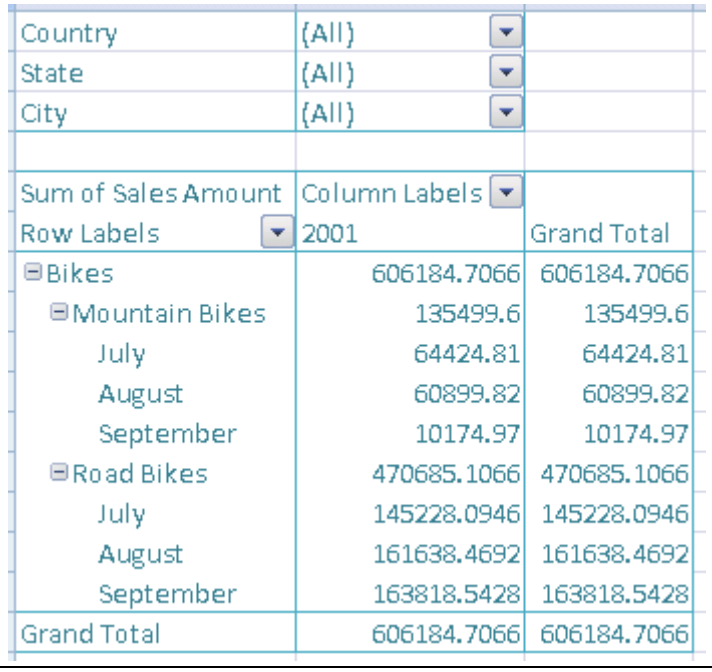
Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	606184.7066	606184.7066
Mountain Bikes	135499.6	135499.6
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	470685.1066	470685.1066
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	**606184.7066**	**606184.7066**

PivotTable Style	[*Example*: (informative)
PivotStyleMedium4	
PivotStyleMedium3	

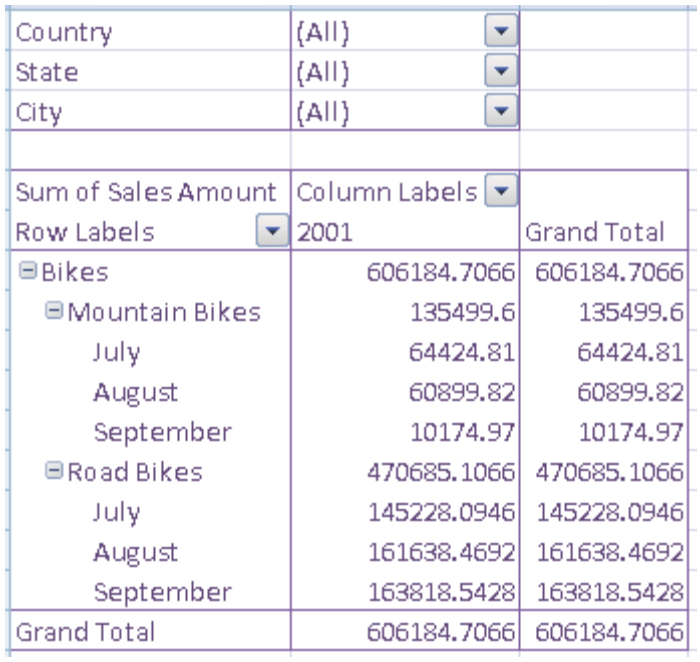
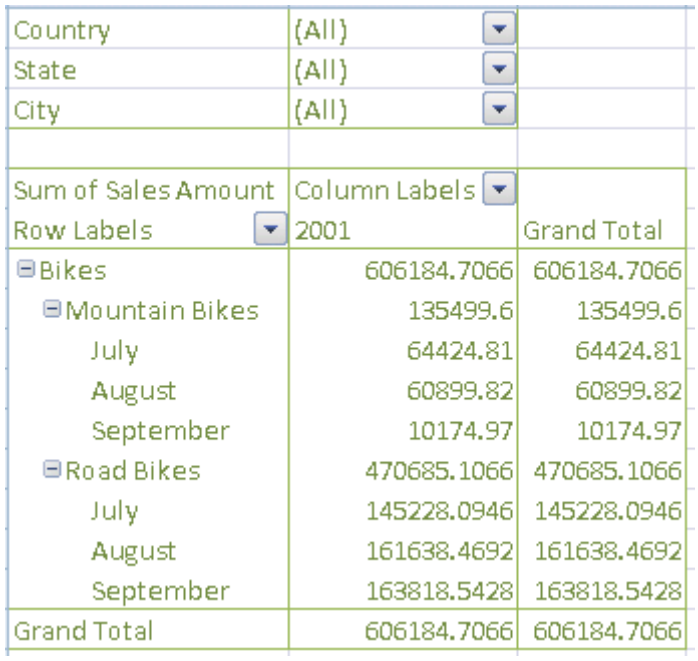
PivotTable Style	[*Example*: (informative)
PivotStyleMedium2	
PivotStyleMedium1	

Country	(All)	
State	(All)	
City	(All)	

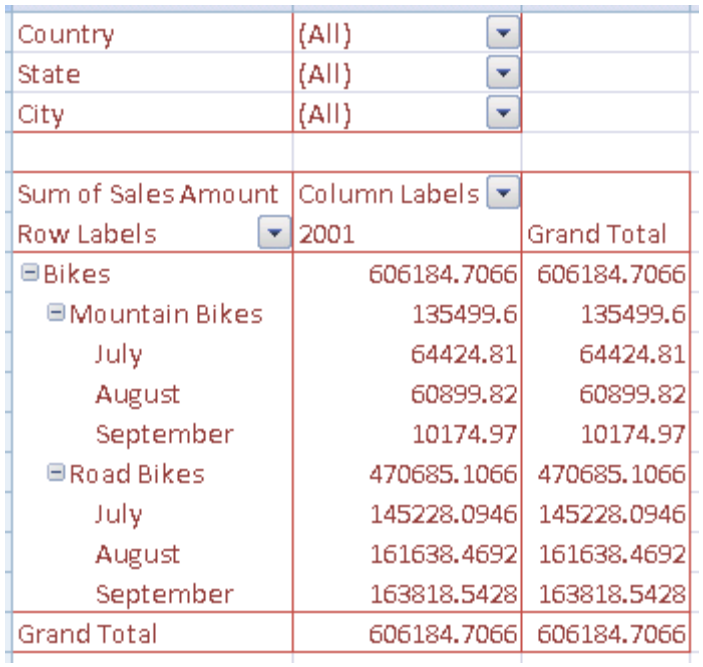
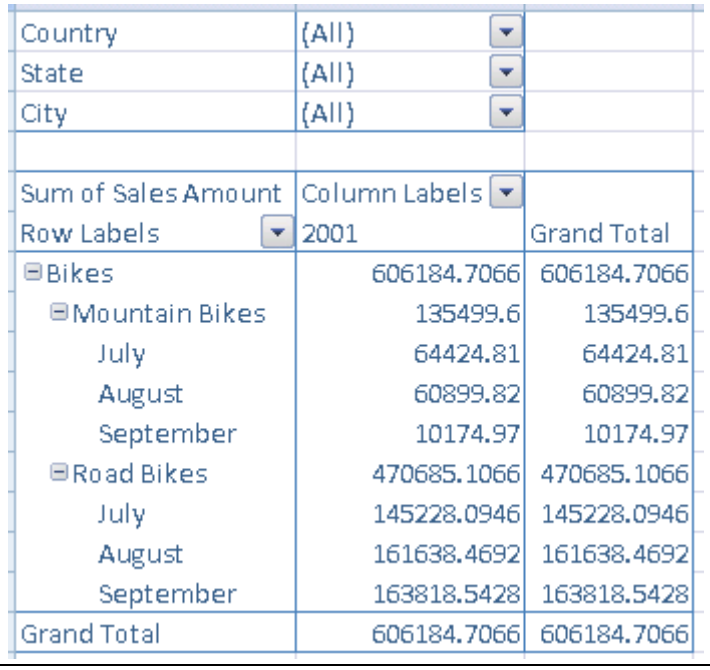
Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	606184.7066	606184.7066
Mountain Bikes	135499.6	135499.6
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	470685.1066	470685.1066
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	**606184.7066**	**606184.7066**

PivotTable Style	[*Example*: (informative)
PivotStyleLight28	
PivotStyleLight27	

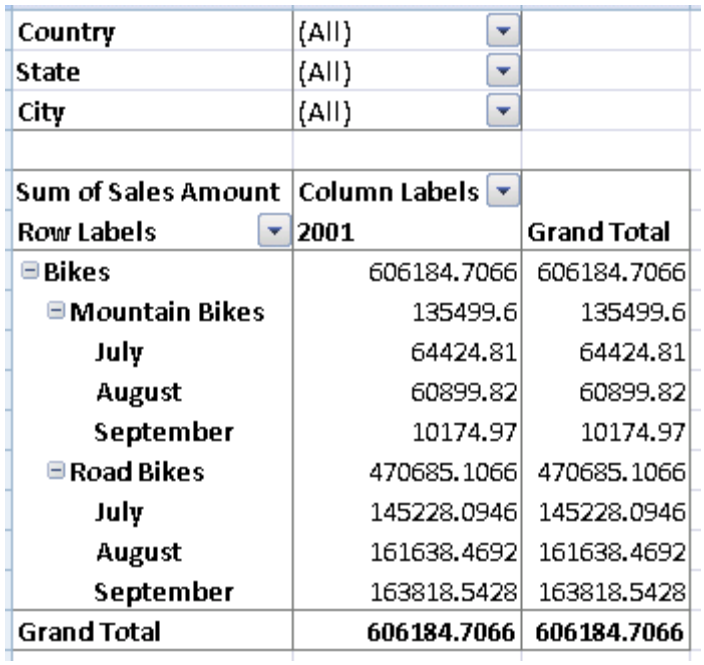
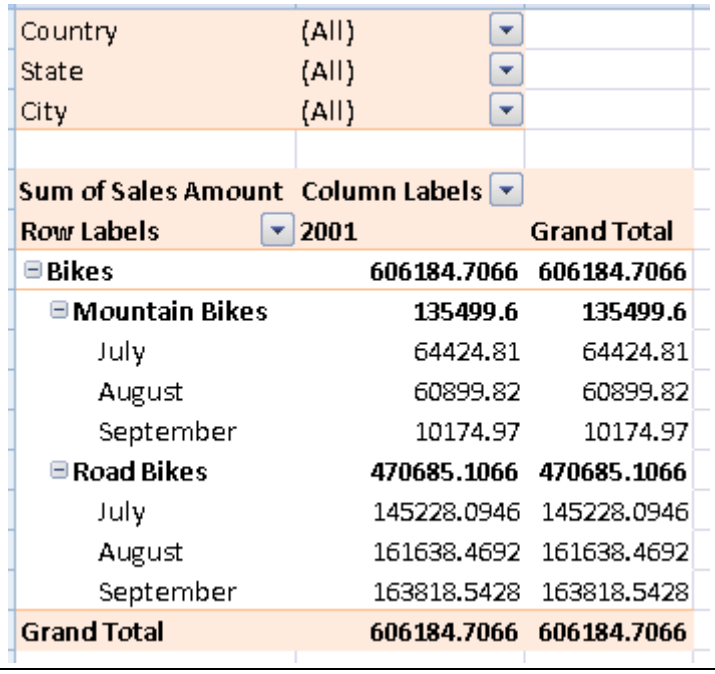
Country	(All)	
State	(All)	
City	(All)	
Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	606184.7066	606184.7066
Mountain Bikes	135499.6	135499.6
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	470685.1066	470685.1066
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	606184.7066	606184.7066

PivotTable Style	[*Example*: (informative)
PivotStyleLight26	
PivotStyleLight25	

Country	(All)	
State	(All)	
City	(All)	
Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	606184.7066	606184.7066
Mountain Bikes	135499.6	135499.6
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	470685.1066	470685.1066
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	606184.7066	606184.7066

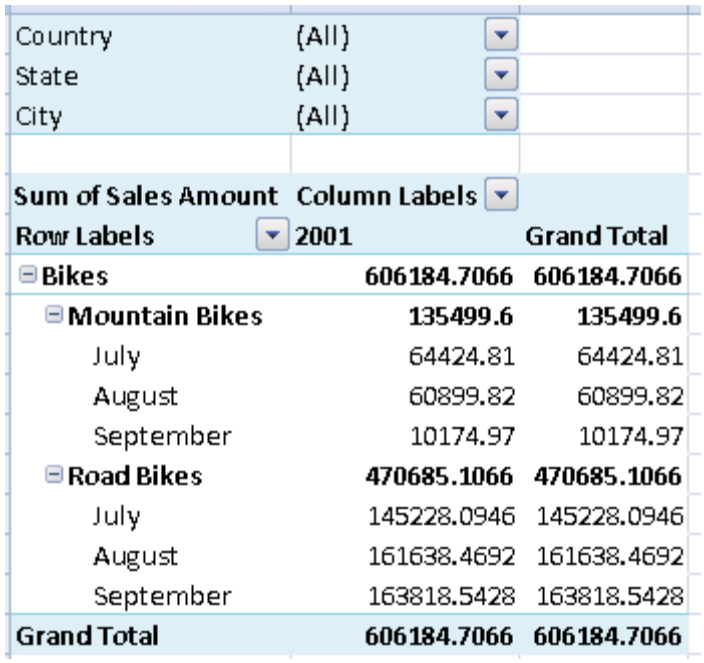
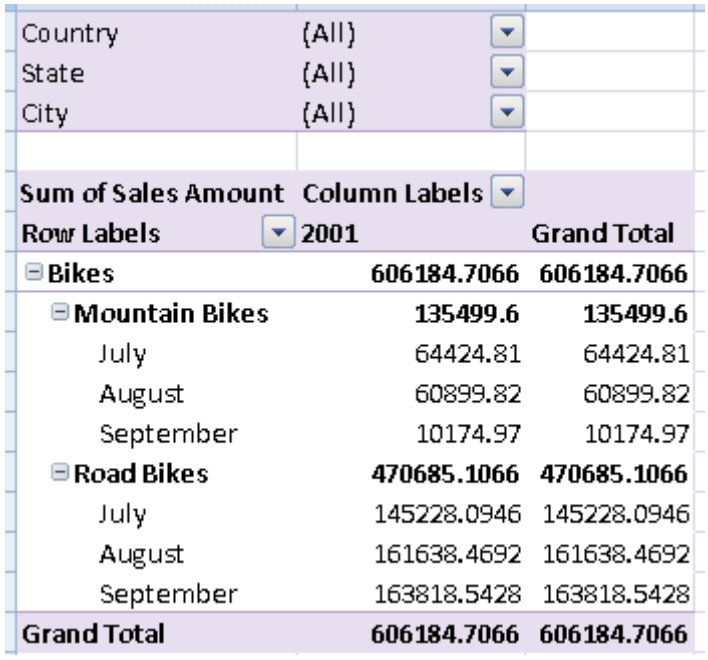
PivotTable Style	[*Example*: (informative)
PivotStyleLight24	
PivotStyleLight23	

Country	(All)	
State	(All)	
City	(All)	
Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	606184.7066	606184.7066
Mountain Bikes	135499.6	135499.6
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	470685.1066	470685.1066
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	606184.7066	606184.7066

PivotTable Style	[*Example*: (informative)
PivotStyleLight22	
PivotStyleLight21	

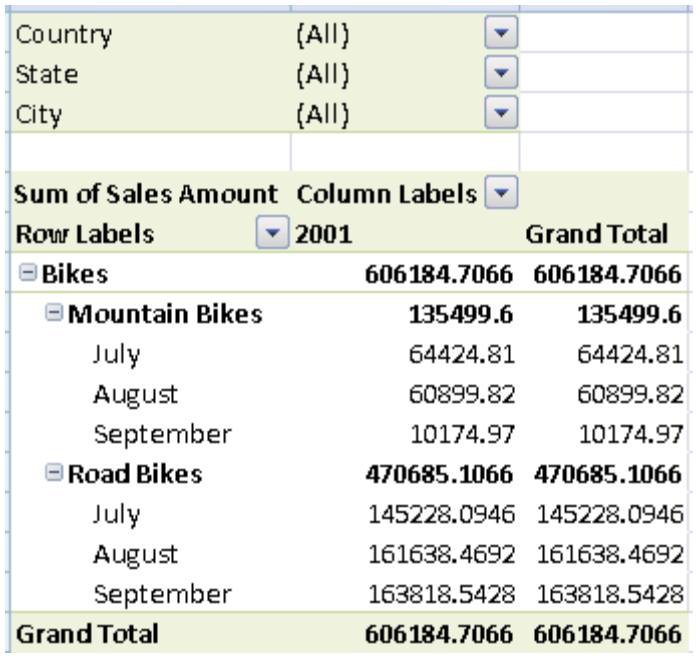
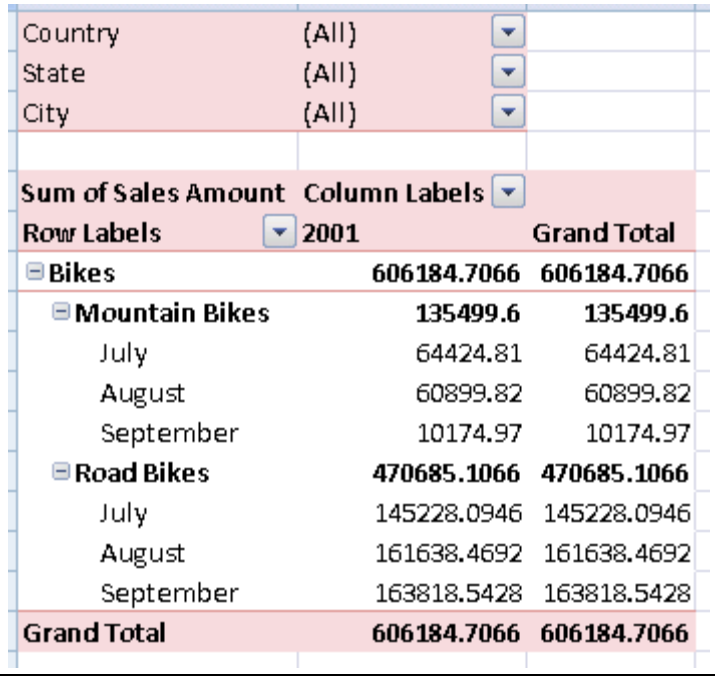
Country	(All)	
State	(All)	
City	(All)	
Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	606184.7066	606184.7066
Mountain Bikes	135499.6	135499.6
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	470685.1066	470685.1066
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	606184.7066	606184.7066

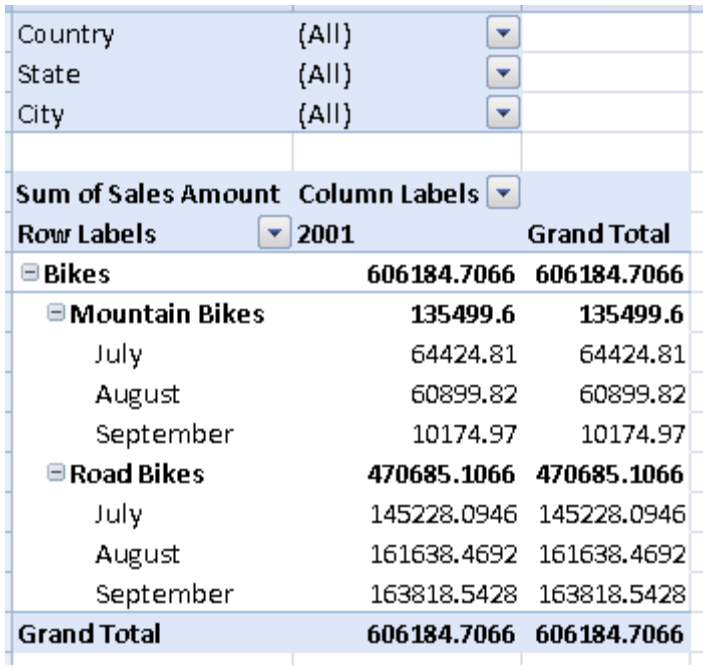
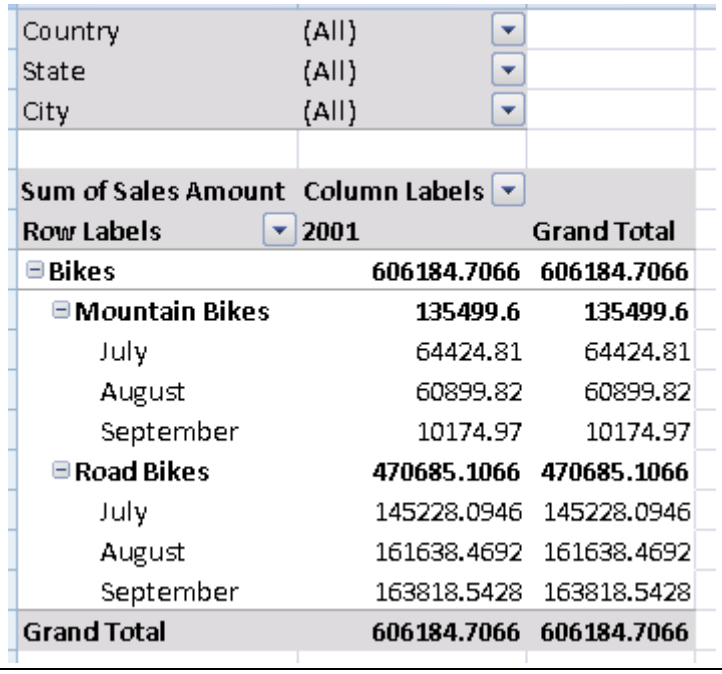
Country	(All)	
State	(All)	
City	(All)	
Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	606184.7066	606184.7066
Mountain Bikes	135499.6	135499.6
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	470685.1066	470685.1066
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	606184.7066	606184.7066

PivotTable Style	[*Example*: (informative)
PivotStyleLight20	
PivotStyleLight19	

Country (All)
State (All)
City (All)

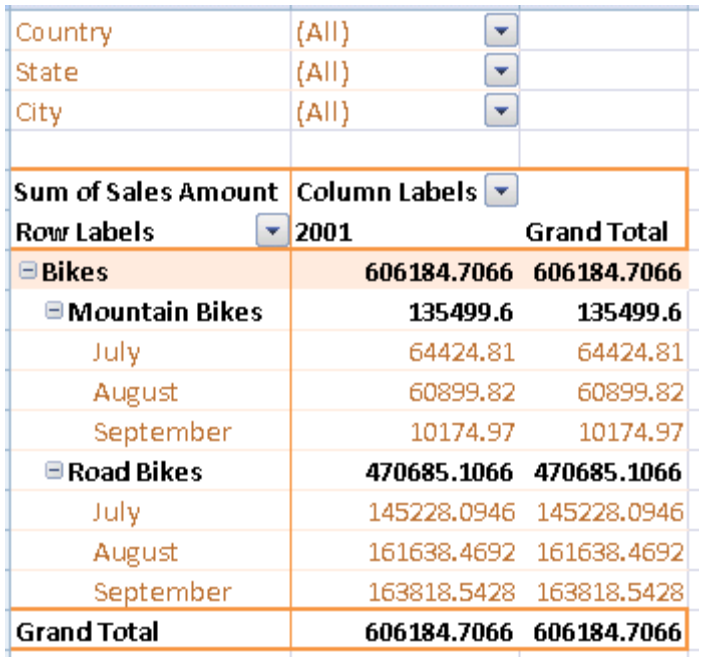
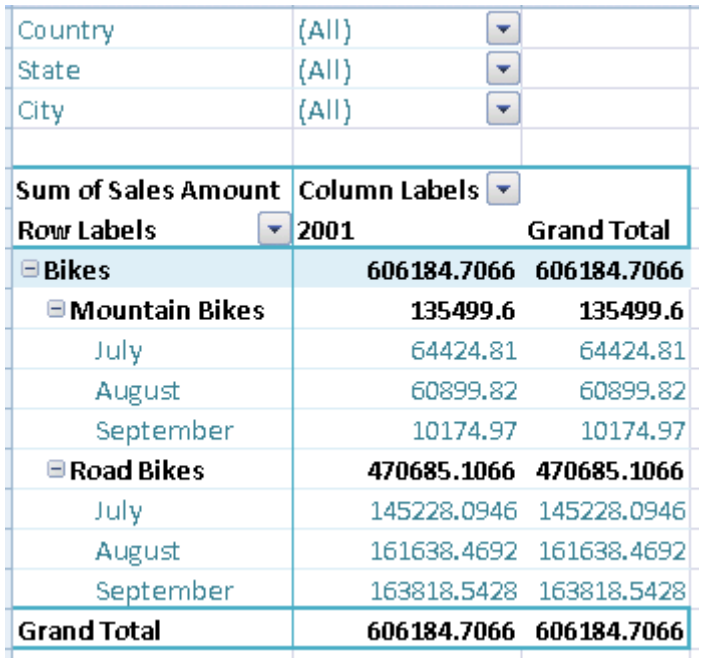
Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	**606184.7066**	**606184.7066**
Mountain Bikes	**135499.6**	**135499.6**
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	**470685.1066**	**470685.1066**
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	**606184.7066**	**606184.7066**

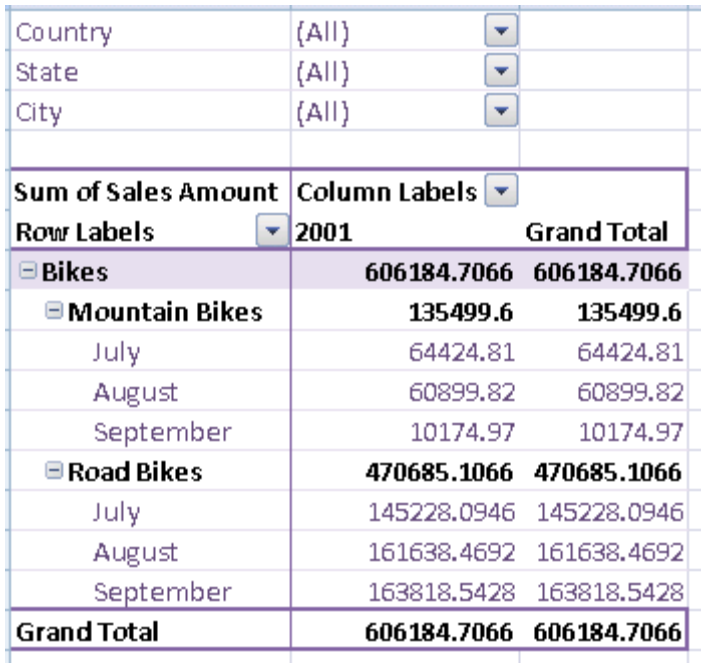
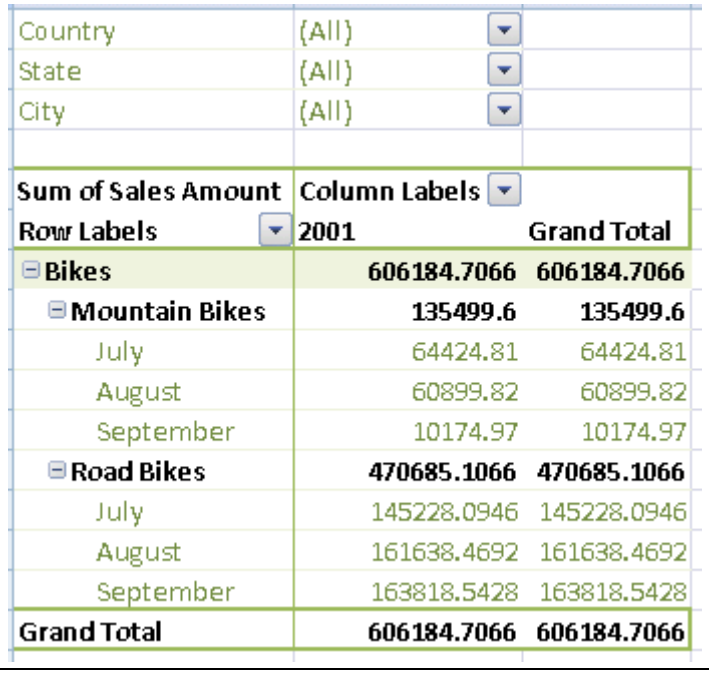
PivotTable Style	[*Example*: (informative)
PivotStyleLight18	
PivotStyleLight17	

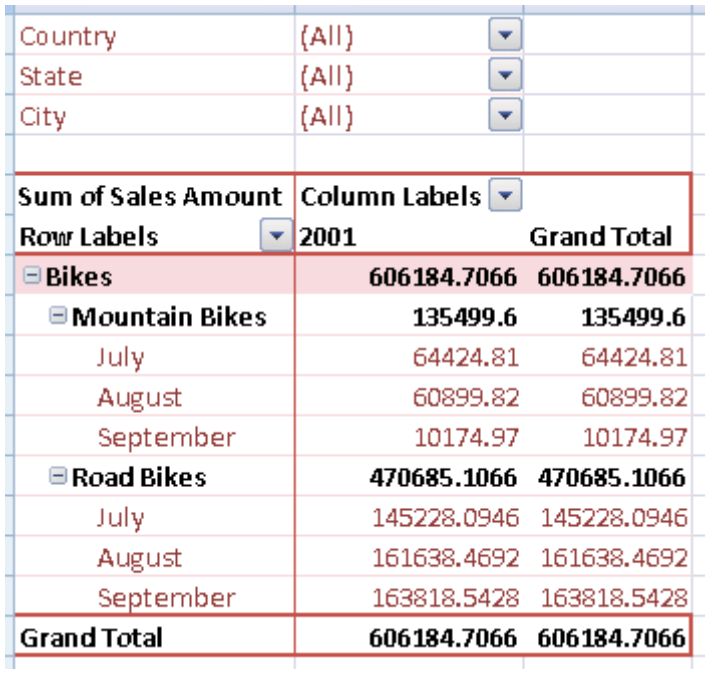
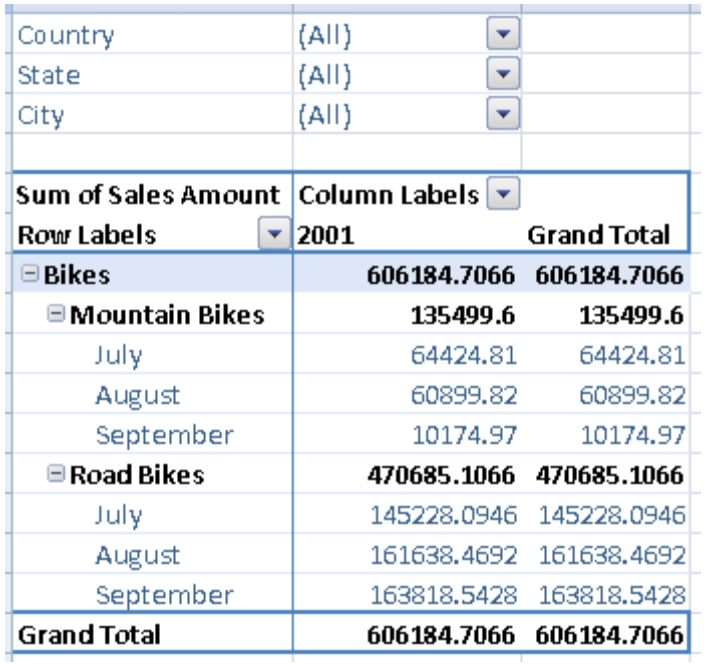
PivotTable Style	[*Example*: (informative)
PivotStyleLight16	
PivotStyleLight15	

Country (All)
State (All)
City (All)

Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	**606184.7066**	**606184.7066**
Mountain Bikes	**135499.6**	**135499.6**
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	**470685.1066**	**470685.1066**
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	**606184.7066**	**606184.7066**

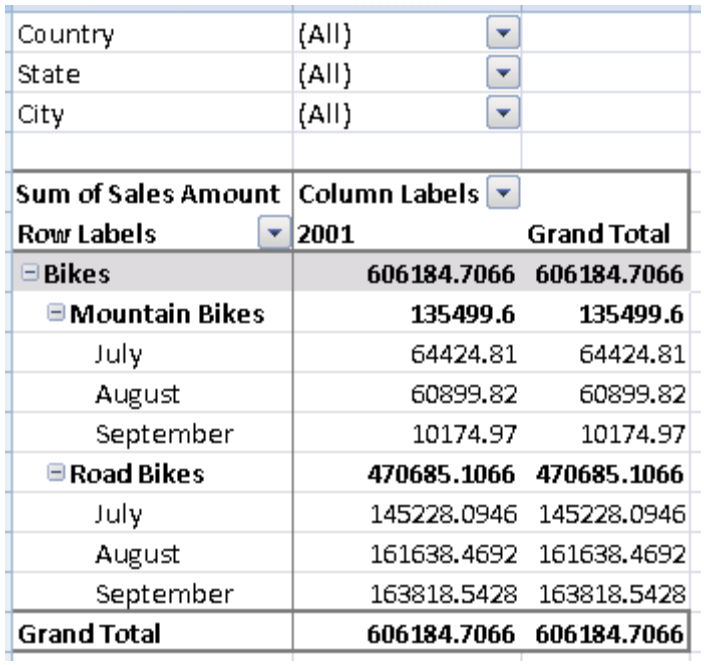
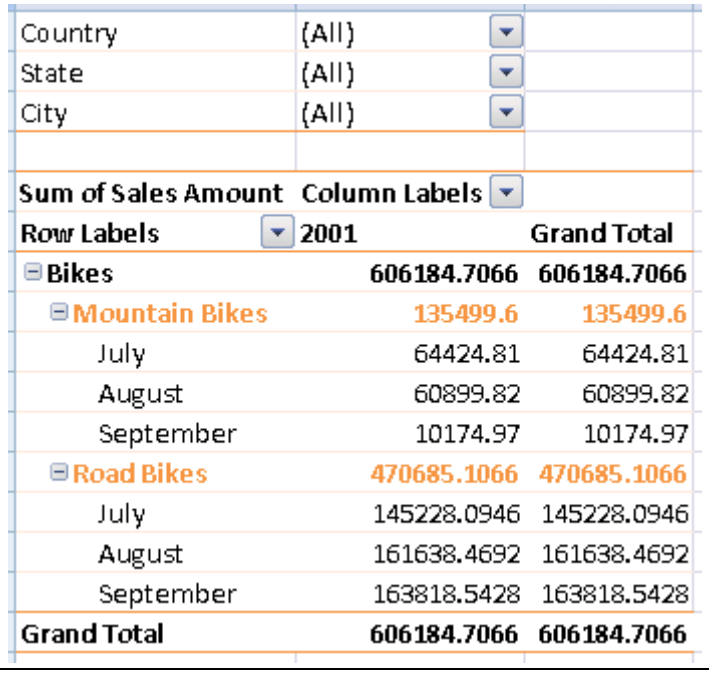
PivotTable Style	[*Example*: (informative)
PivotStyleLight14	
PivotStyleLight13	

PivotTable Style	[*Example*: (informative)
PivotStyleLight12	
PivotStyleLight11	

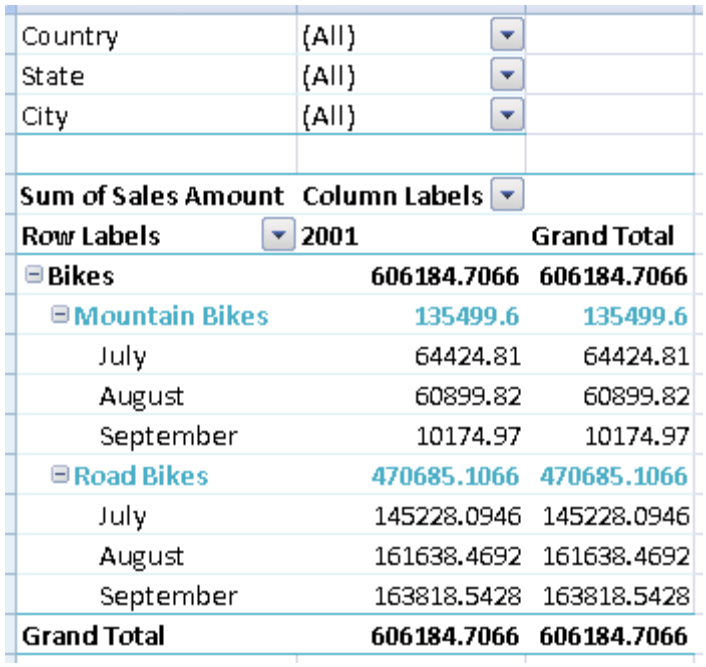
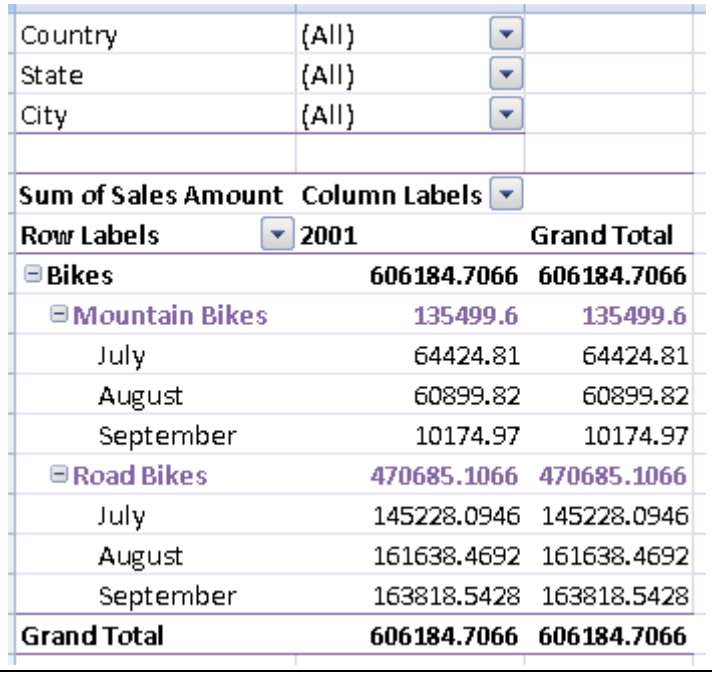
PivotTable Style	[*Example*: (informative)
PivotStyleLight10	
PivotStyleLight9	

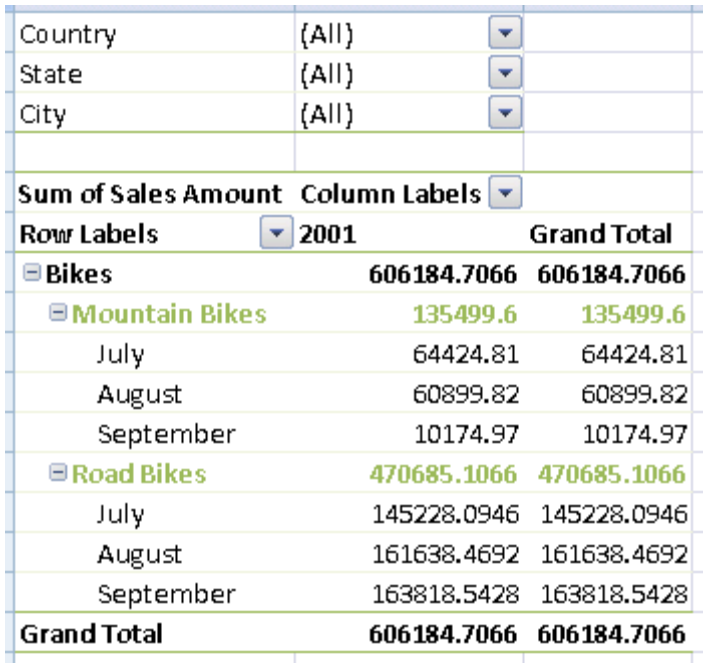
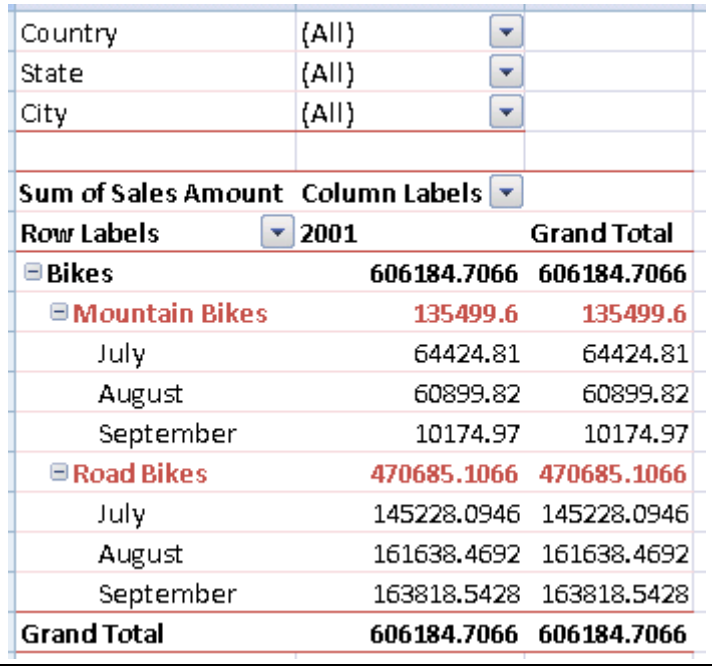
Country	(All)
State	(All)
City	(All)

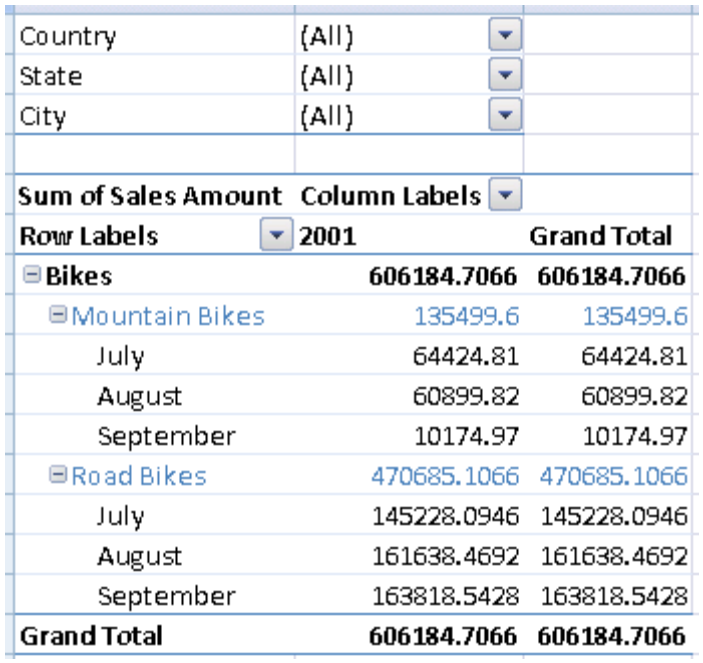
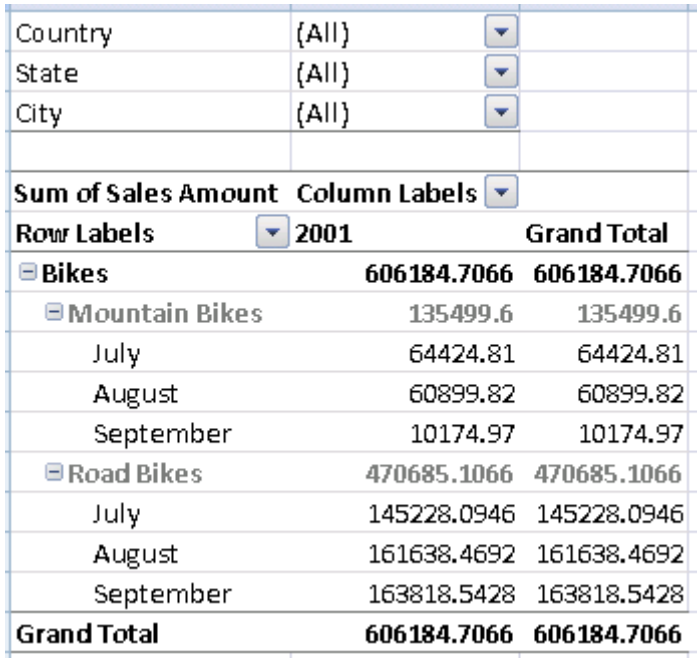
Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	606184.7066	606184.7066
Mountain Bikes	135499.6	135499.6
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	470685.1066	470685.1066
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	606184.7066	606184.7066

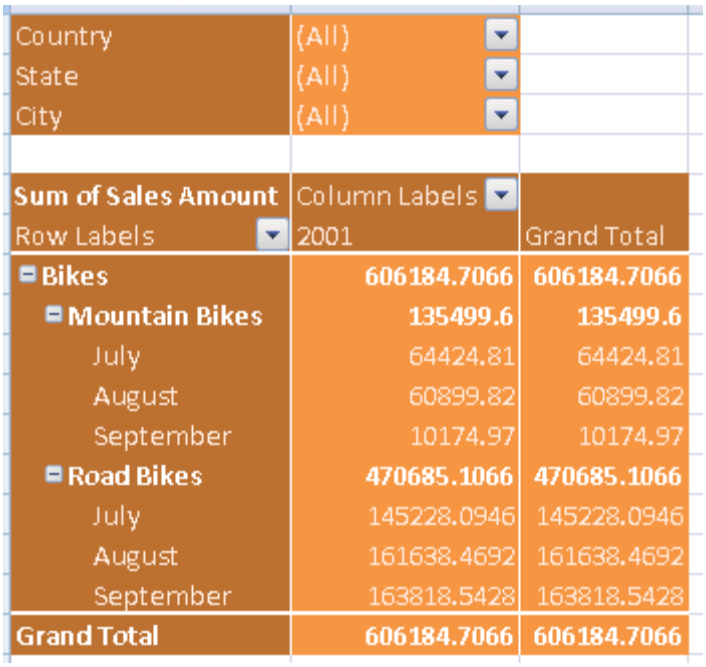
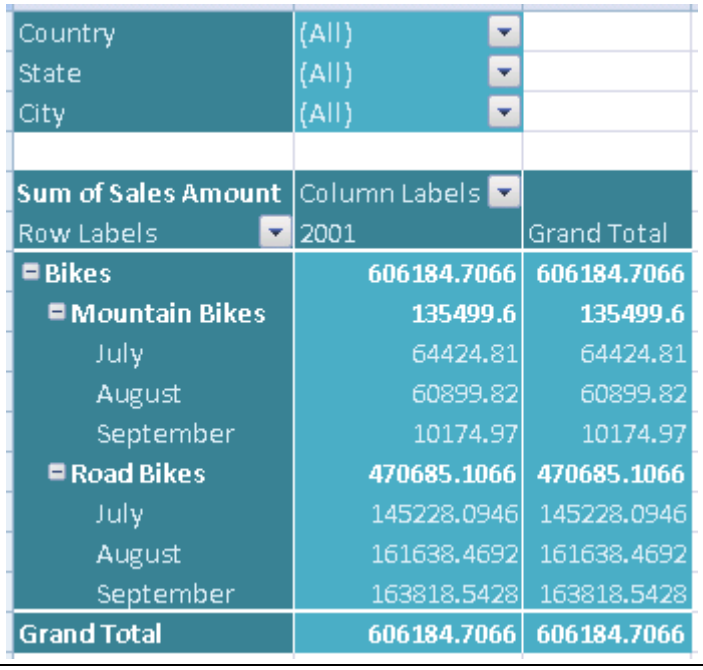
PivotTable Style	[*Example*: (informative)
PivotStyleLight8	
PivotStyleLight7	

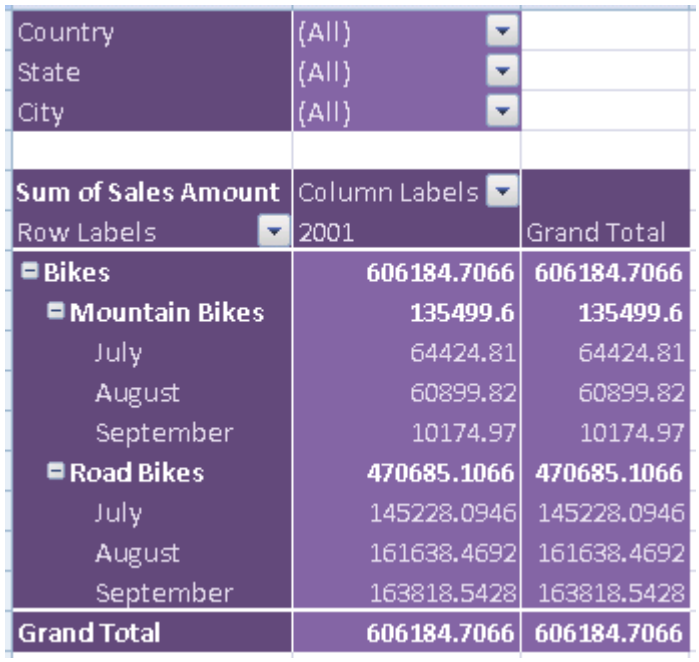
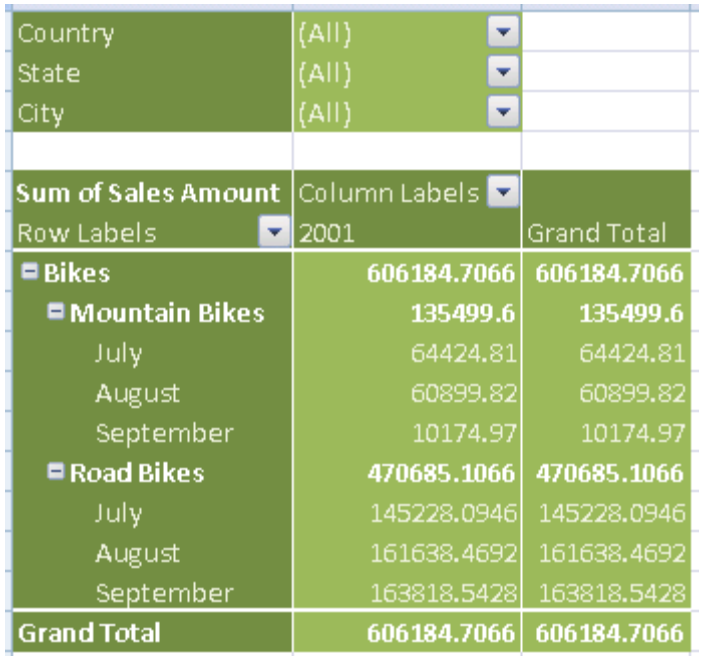
Country	(All)	
State	(All)	
City	(All)	
Sum of Sales Amount	**Column Labels**	
Row Labels	**2001**	**Grand Total**
Bikes	**606184.7066**	**606184.7066**
Mountain Bikes	**135499.6**	**135499.6**
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	**470685.1066**	**470685.1066**
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	**606184.7066**	**606184.7066**

PivotTable Style	[*Example*: (informative)
PivotStyleLight6	
PivotStyleLight5	

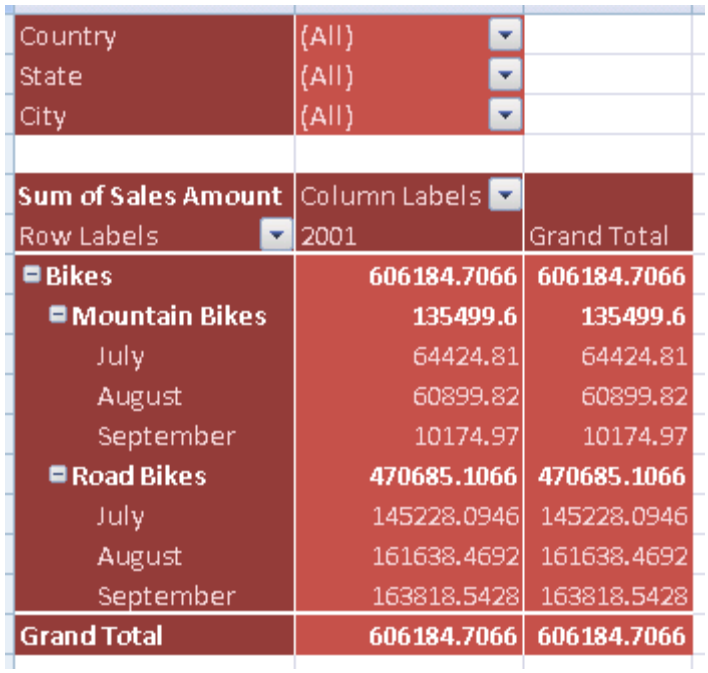
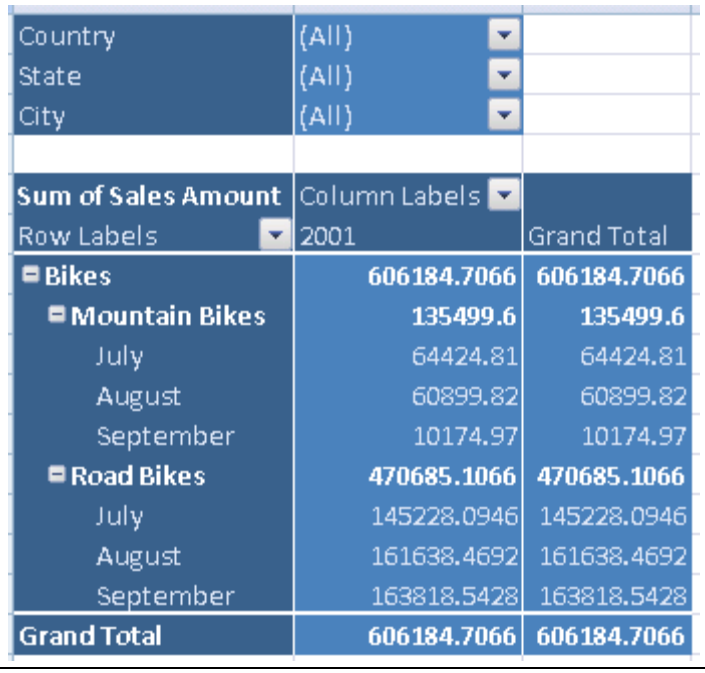
PivotTable Style	[*Example*: (informative)
PivotStyleLight4	
PivotStyleLight3	

PivotTable Style	[*Example*: (informative)
PivotStyleLight2	
PivotStyleLight1	

PivotTable Style	[*Example*: (informative)
PivotStyleDark28	
PivotStyleDark27	

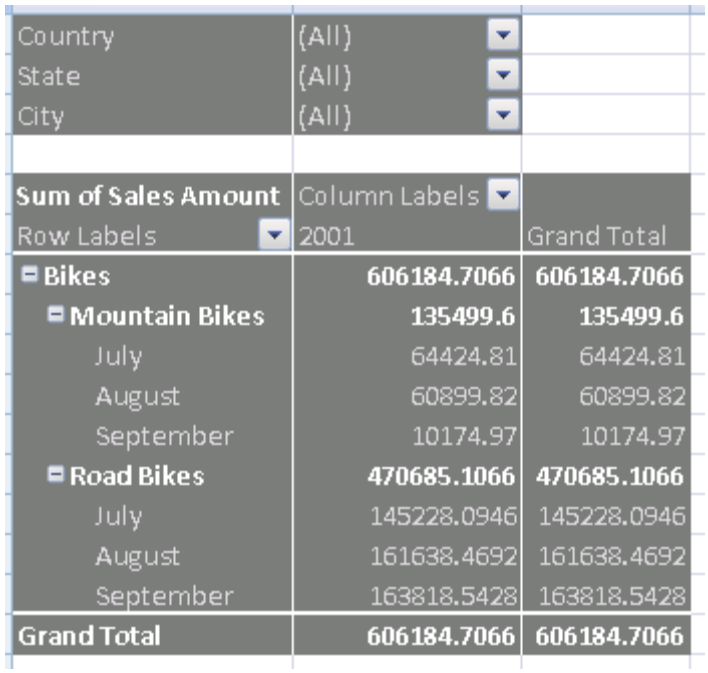
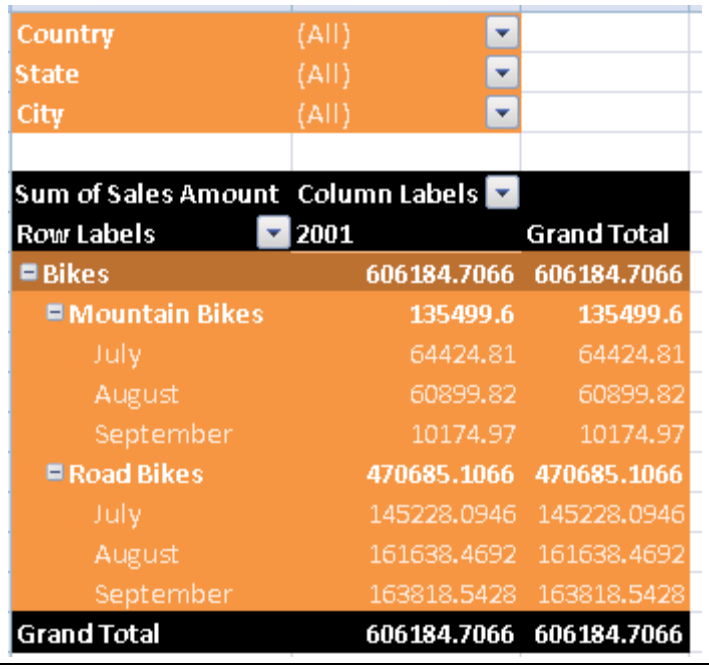
PivotTable Style	[*Example*: (informative)
PivotStyleDark26	
PivotStyleDark25	

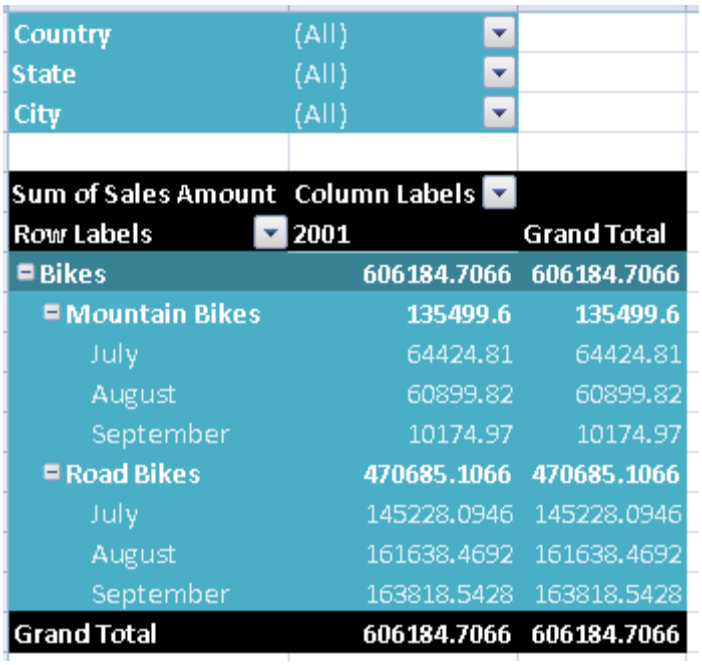
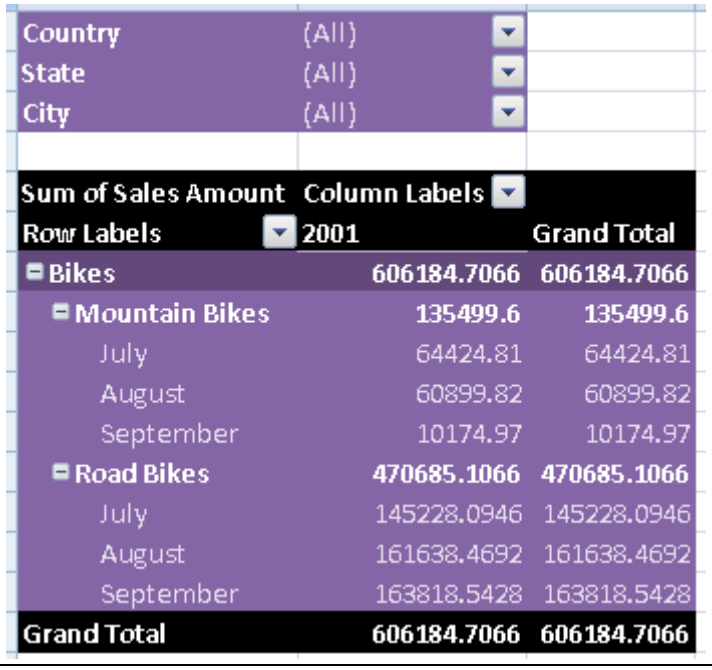
Country	(All)	
State	(All)	
City	(All)	
Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	**606184.7066**	**606184.7066**
Mountain Bikes	**135499.6**	**135499.6**
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	**470685.1066**	**470685.1066**
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	**606184.7066**	**606184.7066**

PivotTable Style	[*Example*: (informative)
PivotStyleDark24	
PivotStyleDark23	

Country (All)
State (All)
City (All)

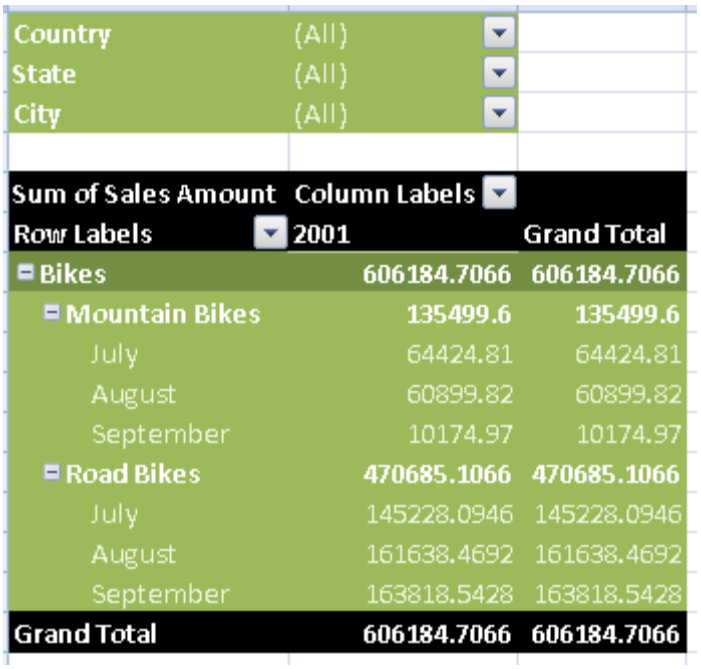
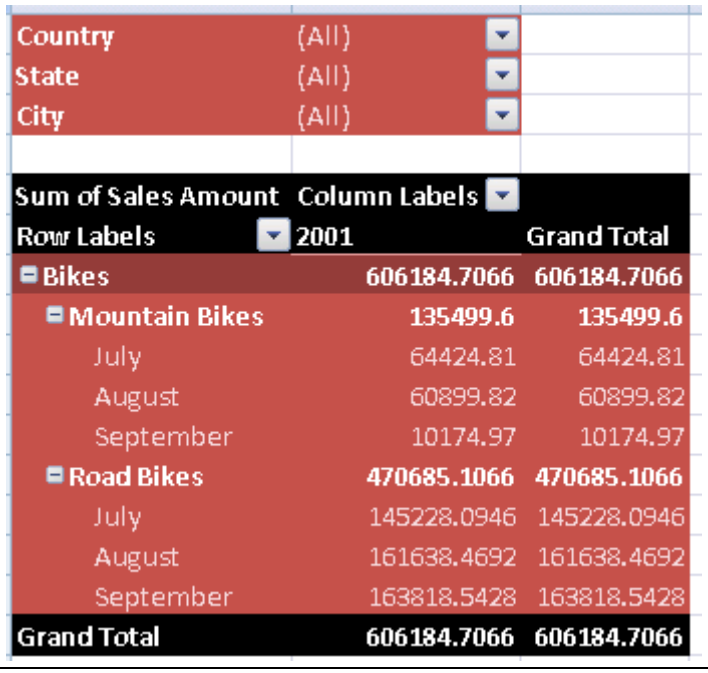
Sum of Sales Amount / Row Labels	Column Labels / 2001	Grand Total
Bikes	**606184.7066**	**606184.7066**
Mountain Bikes	**135499.6**	**135499.6**
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	**470685.1066**	**470685.1066**
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	**606184.7066**	**606184.7066**

PivotTable Style	[*Example*: (informative)
PivotStyleDark22	
PivotStyleDark21	

PivotTable Style	[*Example*: (informative)
PivotStyleDark20	
PivotStyleDark19	

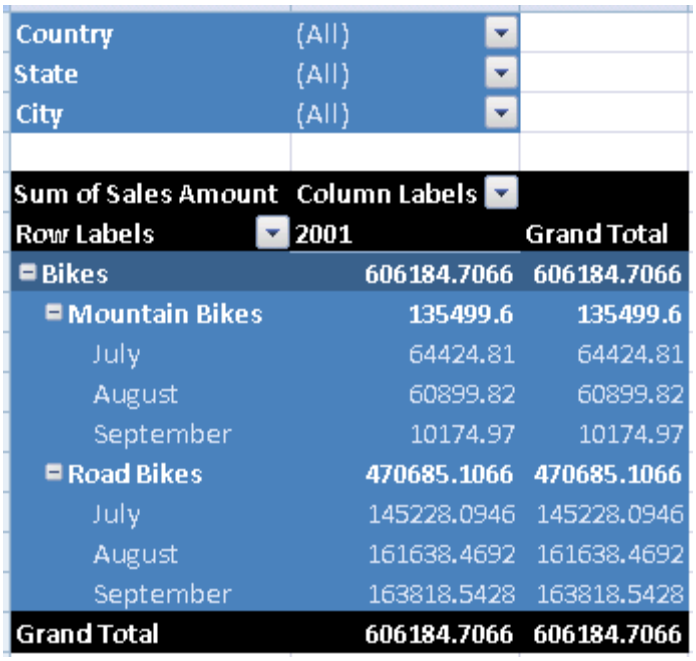
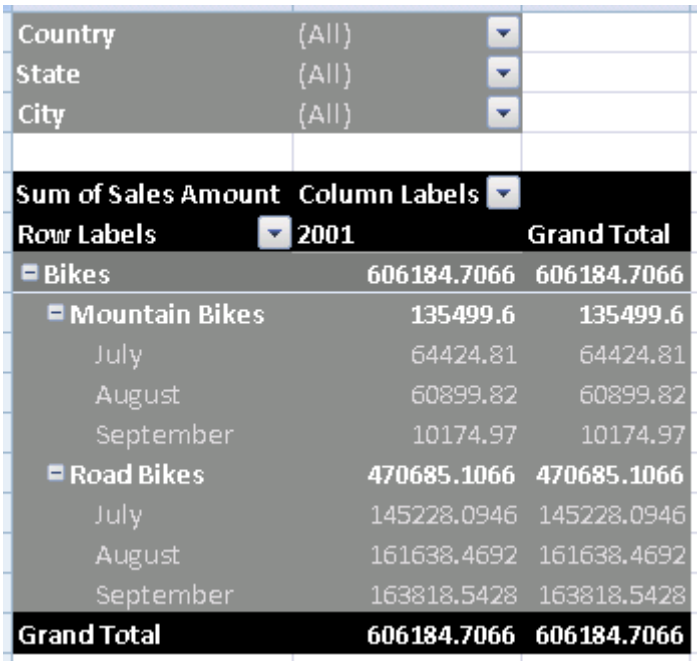
Country	(All)
State	(All)
City	(All)

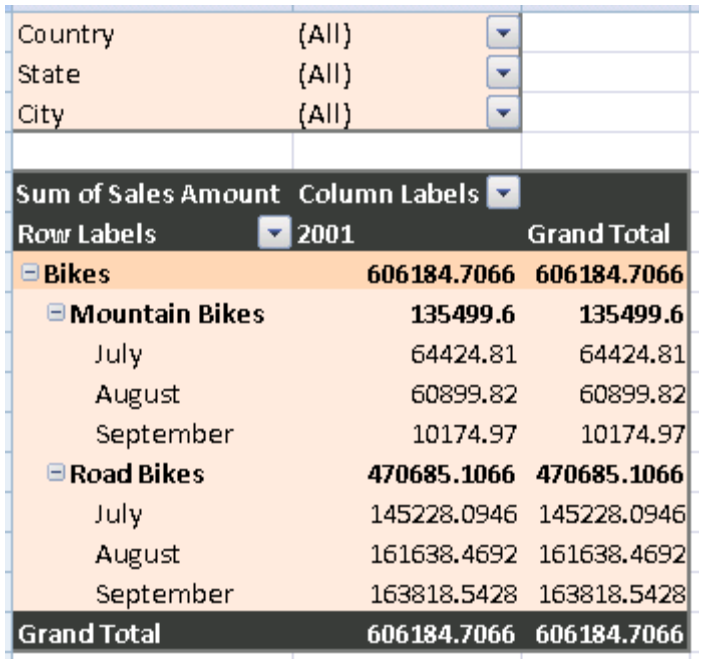
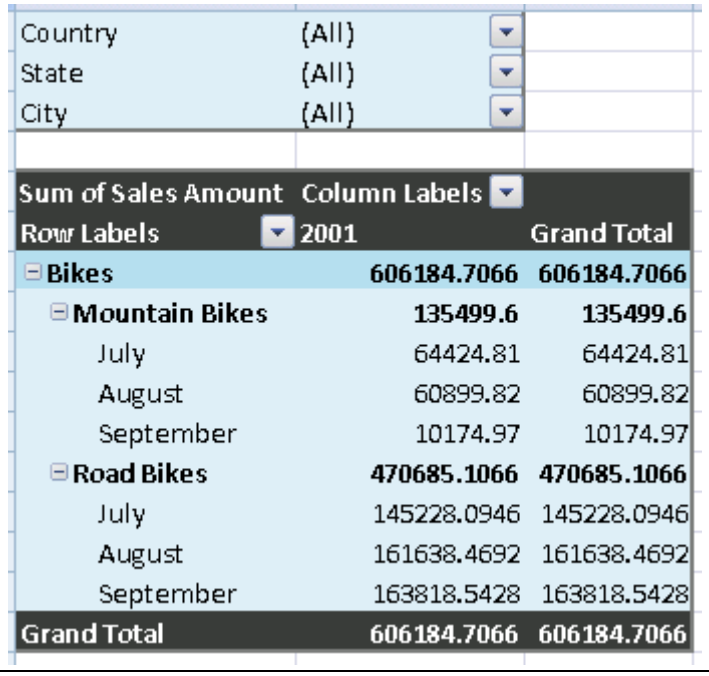
Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	606184.7066	606184.7066
Mountain Bikes	135499.6	135499.6
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	470685.1066	470685.1066
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	606184.7066	606184.7066

PivotTable Style	[*Example*: (informative)
PivotStyleDark18	
PivotStyleDark17	

Country	(All)
State	(All)
City	(All)

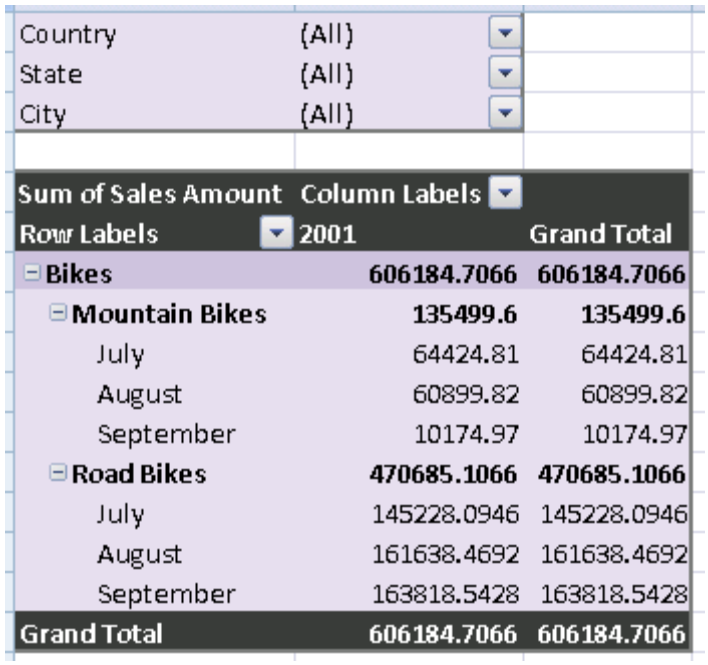
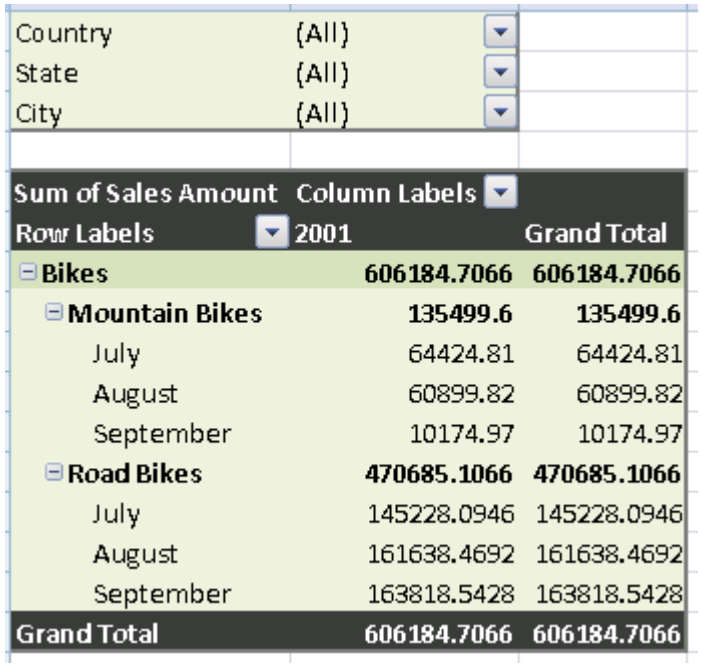
Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	606184.7066	606184.7066
Mountain Bikes	135499.6	135499.6
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	470685.1066	470685.1066
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	606184.7066	606184.7066

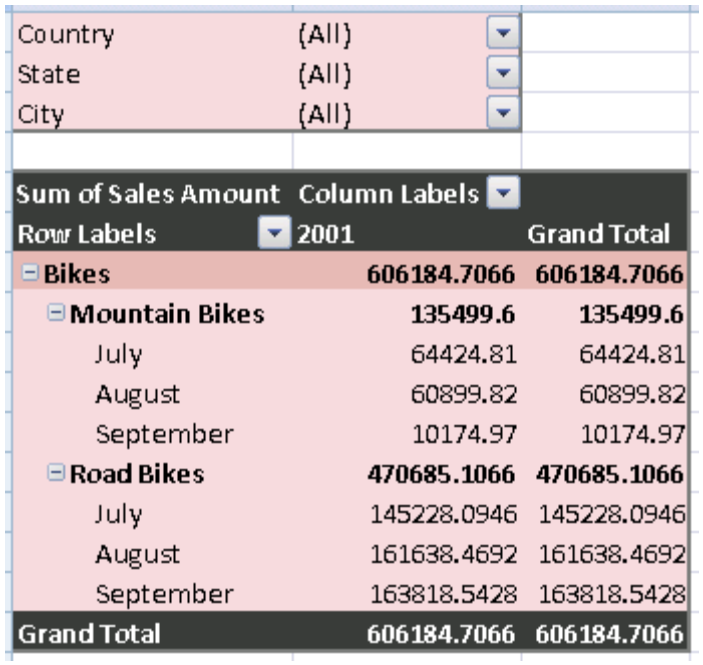
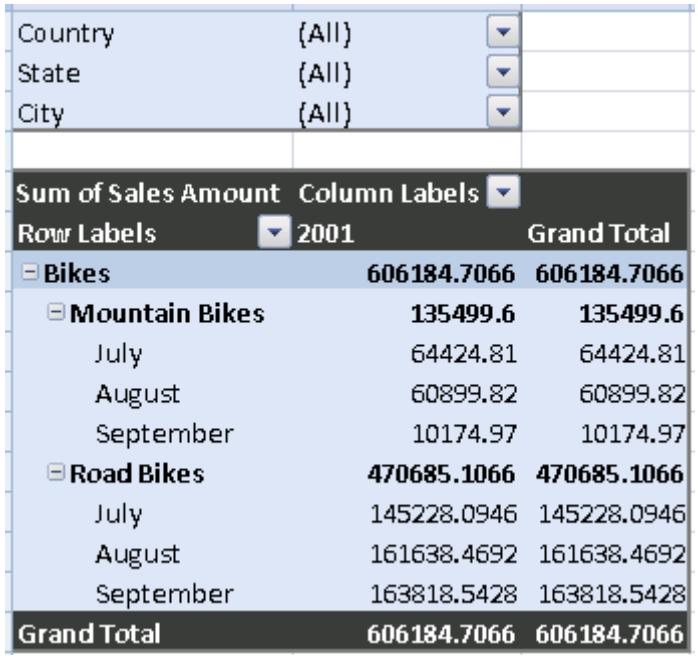
PivotTable Style	[*Example*: (informative)
PivotStyleDark16	
PivotStyleDark15	

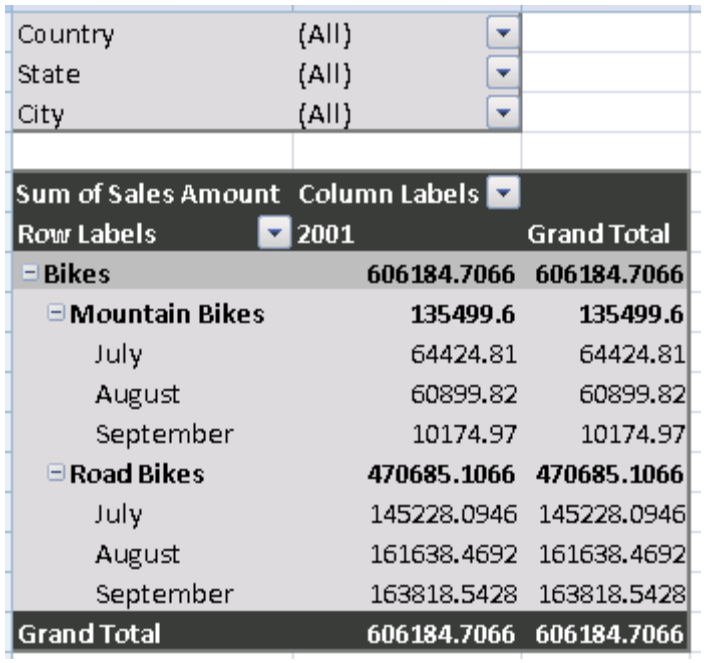
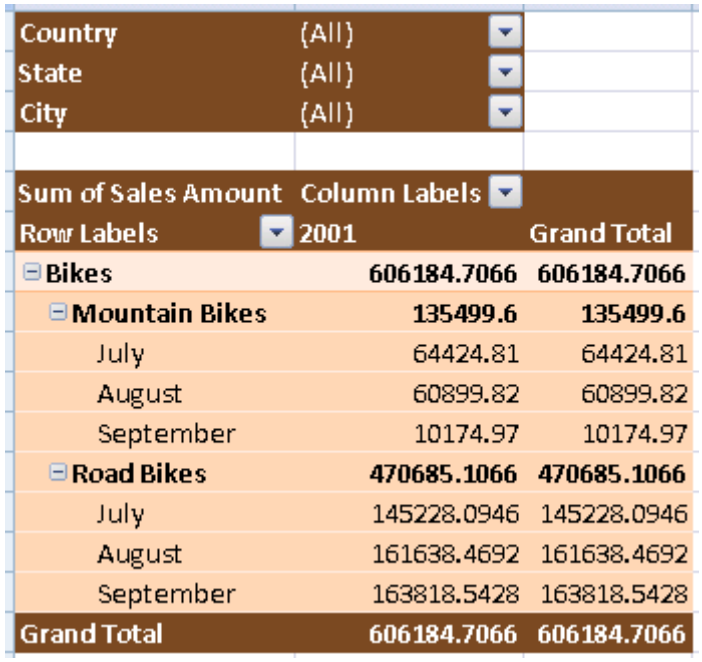
PivotTable Style	[*Example*: (informative)
PivotStyleDark14	
PivotStyleDark13	

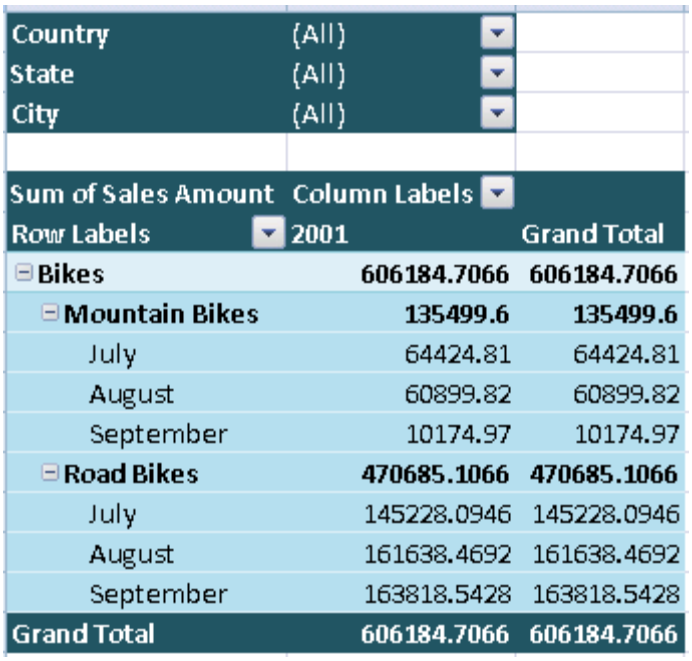
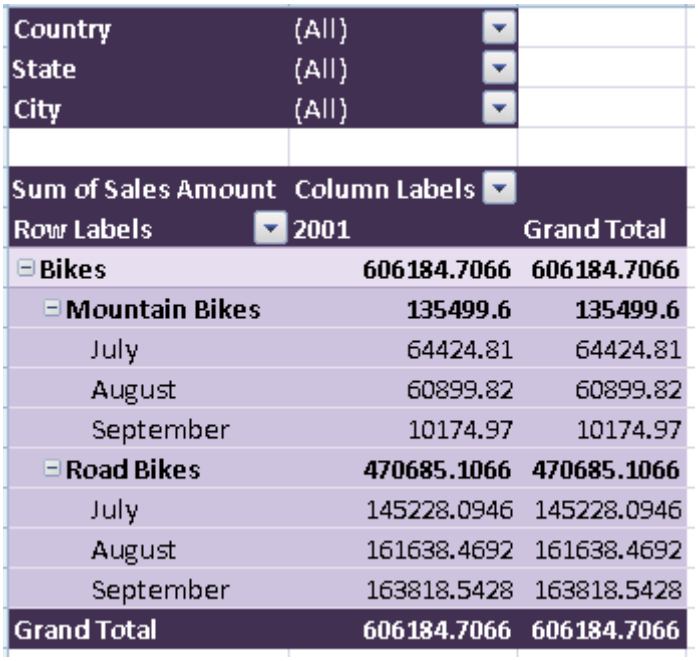
Country (All)
State (All)
City (All)

Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	**606184.7066**	**606184.7066**
Mountain Bikes	**135499.6**	**135499.6**
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	**470685.1066**	**470685.1066**
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	606184.7066	606184.7066

PivotTable Style	[*Example*: (informative)
PivotStyleDark12	
PivotStyleDark11	

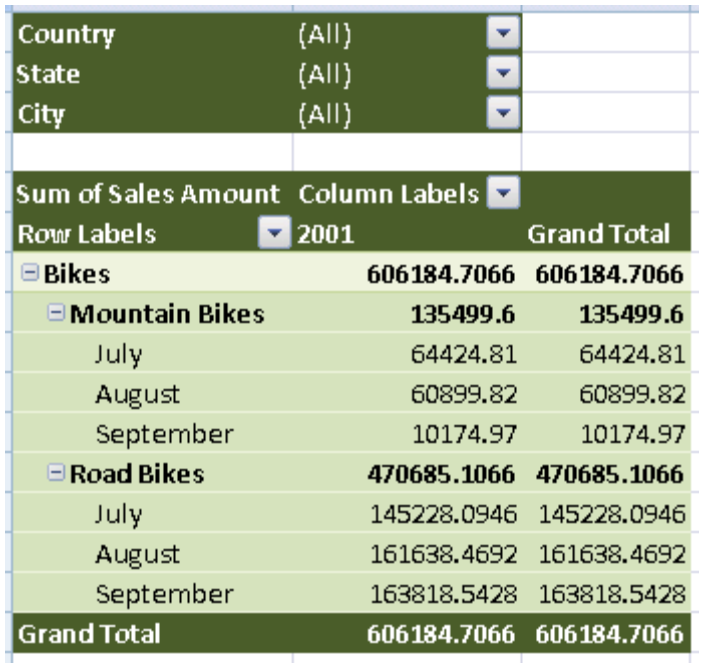
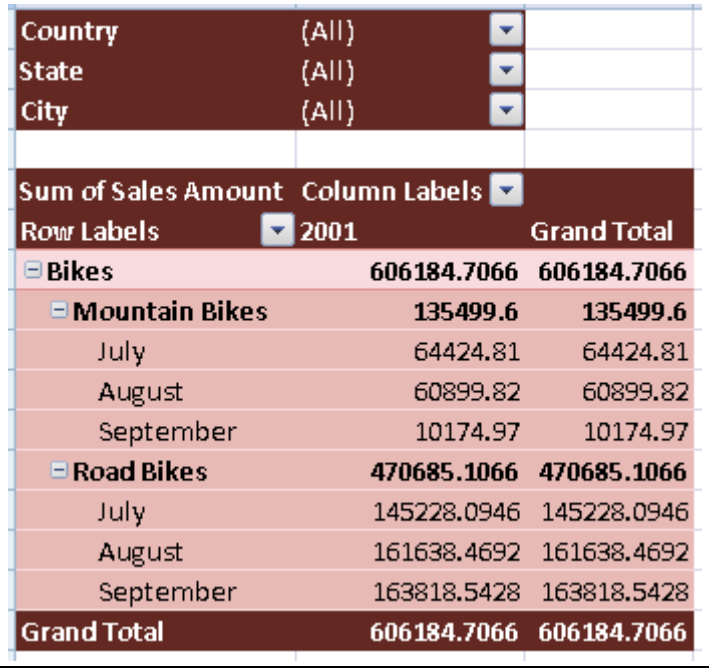
PivotTable Style	[*Example*: (informative)
PivotStyleDark10	
PivotStyleDark9	

PivotTable Style	[*Example*: (informative)
PivotStyleDark8	
PivotStyleDark7	

PivotTable Style	[*Example*: (informative)
PivotStyleDark6	
PivotStyleDark5	

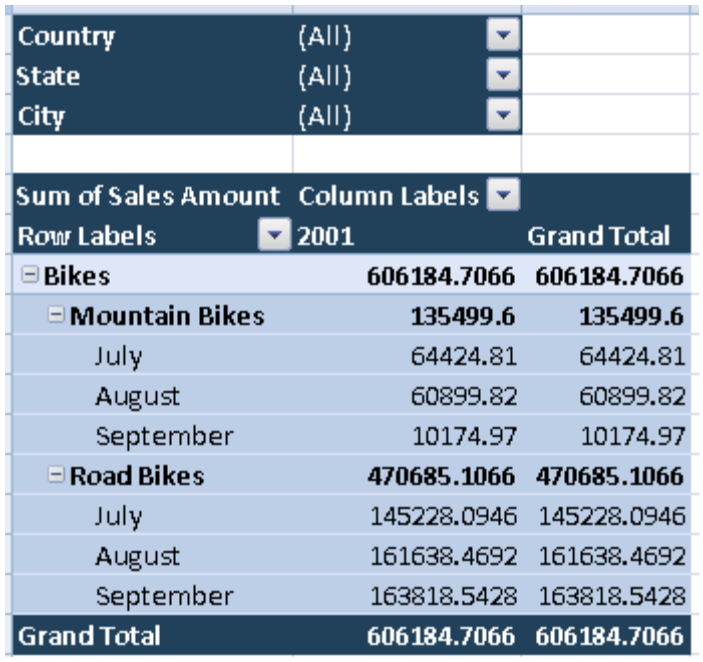
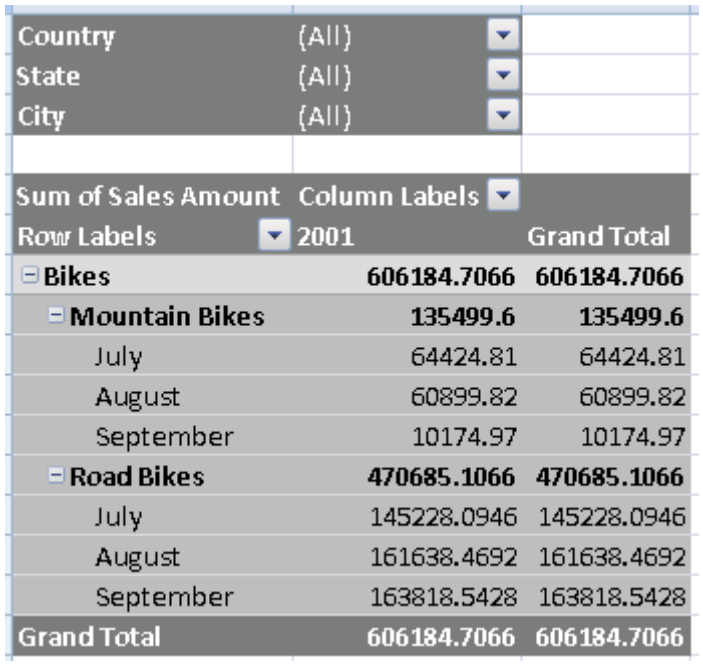
Country	(All)
State	(All)
City	(All)

Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	**606184.7066**	**606184.7066**
Mountain Bikes	**135499.6**	**135499.6**
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	**470685.1066**	**470685.1066**
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	**606184.7066**	**606184.7066**

PivotTable Style	[*Example*: (informative)
PivotStyleDark4	
PivotStyleDark3	

Country	(All)
State	(All)
City	(All)

Sum of Sales Amount	Column Labels	
Row Labels	2001	Grand Total
Bikes	**606184.7066**	**606184.7066**
Mountain Bikes	**135499.6**	**135499.6**
July	64424.81	64424.81
August	60899.82	60899.82
September	10174.97	10174.97
Road Bikes	**470685.1066**	**470685.1066**
July	145228.0946	145228.0946
August	161638.4692	161638.4692
September	163818.5428	163818.5428
Grand Total	**606184.7066**	**606184.7066**

PivotTable Style	[*Example*: (informative)
PivotStyleDark2	
PivotStyleDark1	

Parent Elements
tableStyles (§3.8.42)

Child Elements	Subclause
tableStyleElement (Table Style)	§3.8.41

Attributes	Description
count (Table Style Count)	Count of table style elements defined for this table style. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
name (Table Style Name)	Name of this table style. The possible values for this attribute are defined by the XML Schema string datatype.
pivot (Pivot Style)	'True' if this table style should be shown as an available pivot table style. Not mutually exclusive with table - both can be true. The possible values for this attribute are defined by the XML Schema boolean datatype.
table (Table)	True if this table style should be shown as an available table style. Not mutually exclusive with pivot - both can be true. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TableStyle">
   <sequence>
      <element name="tableStyleElement" type="CT_TableStyleElement" minOccurs="0"
        maxOccurs="unbounded"/>
   </sequence>
   <attribute name="name" type="xsd:string" use="required"/>
   <attribute name="pivot" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="table" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.8.41 tableStyleElement (Table Style)

This element specifies formatting for one area of a table or PivotTable. Together the sequence of these elements makes up one entire Table style or PivotTable style definition.

The order in which table style element formatting is applied is as follows:

Table Style Element Order

- Whole Table
- First Column Stripe
- Second Column Stripe
- First Row Stripe
- Second Row Stripe

- Last Column
- First Column
- Header Row
- Total Row
- First Header Cell
- Last Header Cell
- First Total Cell
- Last Total Cell

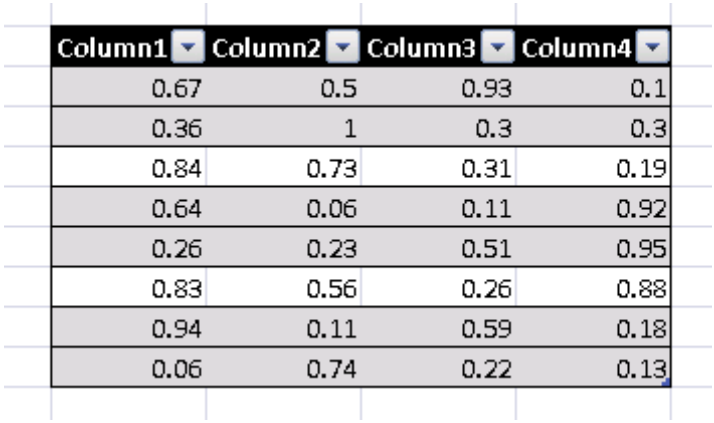
For instance, row stripe formatting 'wins' over column stripe formatting, and both 'win' over whole table formatting.

PivotTable Style Element Order

- Whole Table
- Page Field Labels
- Page Field Values
- First Column Stripe
- Second Column Stripe
- First Row Stripe
- Second Row Stripe
- First Column
- Header Row
- First Header Cell
- Subtotal Column 1
- Subtotal Column 2
- Subtotal Column 3
- Blank Row
- Subtotal Row 1
- Subtotal Row 2
- Subtotal Row 3
- Column Subheading 1
- Column Subheading 2
- Column Subheading 3
- Row Subheading 1
- Row Subheading 2
- Row Subheading 3
- Grand Total Column
- Grand Total Row

Parent Elements

Parent Elements
tableStyle (§3.8.40)

Attributes	Description
dxfId (Formatting Id)	Zero-based index to a dxf record in the dxfs collection, specifying differential formatting to use with this Table or PivotTable style element. The possible values for this attribute are defined by the ST_DxfId simple type (§3.18.26).
size (Band Size)	Number of rows or columns in a single band of striping. Applies only when type is `firstRowStripe`, `secondRowStripe`, `firstColumnStripe`, or `secondColumnStripe`. [*Example*: In this example, the `firstRowStripe` size is set to `2`, and the `secondRowStripe` size is set to `1`:  *end example*] The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
type (Table Style Type)	Identifies this table style element's type. The possible values for this attribute are defined by the ST_TableStyleType simple type (§3.18.79).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TableStyleElement">
   <attribute name="type" type="ST_TableStyleType" use="required"/>
   <attribute name="size" type="xsd:unsignedInt" use="optional" default="1"/>
   <attribute name="dxfId" type="ST_DxfId" use="optional"/>
</complexType>
```

3.8.42 tableStyles (Table Styles)

This element represents a collection of Table style definitions for Table styles and PivotTable styles used in this workbook. It consists of a sequence of tableStyle records, each defining a single Table style.

A Table style is a collection of formatting that applies to structured regions of a Table or PivotTable [*Example*: make the header row & totals bold face, and apply light gray fill to alternating rows in the data portion of the table to achieve striped or banded rows. *end example*]

See the enumeration values in ST_TableStyleType for a listing of structured Table regions to which formatting can be applied, and which together make up a single Table style definition.

Parent Elements
styleSheet (§3.8.39)

Child Elements	Subclause
tableStyle (Table Style)	§3.8.40

Attributes	Description
count (Table Style Count)	Count of table styles defined in this collection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
defaultPivotStyle (Default Pivot Style)	Name of the default table style to apply to new PivotTables. This can be set by the user interface. The possible values for this attribute are defined by the XML Schema string datatype.
defaultTableStyle (Default Table Style)	Name of default table style to apply to new Tables. This can be set by the user interface. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TableStyles">
   <sequence>
      <element name="tableStyle" type="CT_TableStyle" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
   <attribute name="defaultTableStyle" type="xsd:string" use="optional"/>
   <attribute name="defaultPivotStyle" type="xsd:string" use="optional"/>
</complexType>
```

3.8.43 top (Top Border)

This element specifies the color and line style for the top border of a cell.

Parent Elements
border (§3.8.4)

Child Elements	Subclause
color (Data Bar Color)	§3.3.1.14

Attributes	Description
style (Line Style)	The line style for this border. The possible values for this attribute are defined by the ST_BorderStyle simple type (§3.18.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BorderPr">
   <sequence>
      <element name="color" type="CT_Color" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="style" type="ST_BorderStyle" use="optional" default="none"/>
</complexType>
```

3.8.44 vertical (Vertical Inner Border)

This element specifies the color and line style for the vertical inner border(s) of a range of cells. Used in the context of dxf elements only. For example, see the borders definitions for **TableStyleMedium28**.

Parent Elements
border (§3.8.4)

Child Elements	Subclause
color (Data Bar Color)	§3.3.1.14

Attributes	Description
style (Line Style)	The line style for this border. The possible values for this attribute are defined by the ST_BorderStyle simple type (§3.18.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BorderPr">
   <sequence>
      <element name="color" type="CT_Color" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="style" type="ST_BorderStyle" use="optional" default="none"/>
</complexType>
```

3.8.45 xf (Format)

A single xf element describes all of the formatting for a cell.

Parent Elements
cellStyleXfs (§3.8.9); cellXfs (§3.8.10)

Child Elements	Subclause
alignment (Alignment)	§3.8.1
extLst (Future Feature Data Storage Area)	§3.2.10
protection (Protection Properties)	§3.8.33

Attributes	Description
applyAlignment (Apply Alignment)	A boolean value indicating whether the alignment formatting specified for this xf should be applied. The possible values for this attribute are defined by the XML Schema boolean datatype.
applyBorder (Apply Border)	A boolean value indicating whether the border formatting specified for this xf should be applied. The possible values for this attribute are defined by the XML Schema boolean datatype.
applyFill (Apply Fill)	A boolean value indicating whether the fill formatting specified for this xf should be applied. The possible values for this attribute are defined by the XML Schema boolean datatype.
applyFont (Apply Font)	A boolean value indicating whether the font formatting specified for this xf should be applied. The possible values for this attribute are defined by the XML Schema boolean datatype.
applyNumberFormat (Apply Number Format)	A boolean value indicating whether the number formatting specified for this xf should be applied. The possible values for this attribute are defined by the XML Schema boolean datatype.
applyProtection	A boolean value indicating whether the protection formatting specified for this xf should

Attributes	Description
(Apply Protection)	be applied. The possible values for this attribute are defined by the XML Schema boolean datatype.
borderId (Border Id)	Zero-based index of the border record used by this cell format. The possible values for this attribute are defined by the ST_BorderId simple type (§3.18.2).
fillId (Fill Id)	Zero-based index of the fill record used by this cell format. The possible values for this attribute are defined by the ST_FillId simple type (§3.18.31).
fontId (Font Id)	Zero-based index of the font record used by this cell format. The possible values for this attribute are defined by the ST_FontId simple type (§3.18.33).
numFmtId (Number Format Id)	Id of the number format (numFmt) record used by this cell format. The possible values for this attribute are defined by the ST_NumFmtId simple type (§3.18.49).
pivotButton (Pivot Button)	A boolean value indicating whether the cell rendering includes a pivot table dropdown button. The possible values for this attribute are defined by the XML Schema boolean datatype.
quotePrefix (Quote Prefix)	A boolean value indicating whether the text string in a cell should be prefixed by a single quote mark (e.g., 'text). In these cases, the quote is not stored in the Shared Strings Part. The possible values for this attribute are defined by the XML Schema boolean datatype.
xfId (Format Id)	For xf records contained in cellXfs this is the zero-based index of an xf record contained in cellStyleXfs corresponding to the cell style applied to the cell. Not present for xf records contained in cellStyleXfs. The possible values for this attribute are defined by the ST_CellStyleXfId simple type (§3.18.11).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Xf">
    <sequence>
        <element name="alignment" type="CT_CellAlignment" minOccurs="0" maxOccurs="1"/>
        <element name="protection" type="CT_CellProtection" minOccurs="0" maxOccurs="1"/>
        <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
    </sequence>
    <attribute name="numFmtId" type="ST_NumFmtId" use="optional"/>
    <attribute name="fontId" type="ST_FontId" use="optional"/>
    <attribute name="fillId" type="ST_FillId" use="optional"/>
    <attribute name="borderId" type="ST_BorderId" use="optional"/>
    <attribute name="xfId" type="ST_CellStyleXfId" use="optional"/>
    <attribute name="quotePrefix" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="pivotButton" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="applyNumberFormat" type="xsd:boolean" use="optional"/>
    <attribute name="applyFont" type="xsd:boolean" use="optional"/>
    <attribute name="applyFill" type="xsd:boolean" use="optional"/>
    <attribute name="applyBorder" type="xsd:boolean" use="optional"/>
    <attribute name="applyAlignment" type="xsd:boolean" use="optional"/>
    <attribute name="applyProtection" type="xsd:boolean" use="optional"/>
</complexType>
```

3.9 Metadata

A cell in a spreadsheet application can have metadata associated with it. Metadata is just a set of additional properties about the particular cell, and this metadata is stored in the metadata xml part.

There are two types of metadata: cell metadata and value metadata. Cell metadata contains information about the cell itself, and this metadata can be carried along with the cell as it moves (insert, shift, copy/paste, merge, unmerge, etc). Value metadata is information about the value of a particular cell. Value metadata properties can be propagated along with the value as it is referenced in formulas.

The file format is architected such that it supports both value and cell metadata, as well as even allowing for future extensions. Formulas, such as CUBEMEMBER() or CUBE*, shall make use of value metadata as part of the SpreadsheetML standard. So, only value metadata must be implemented as it is used by MDX cube functions for retrieving data from OLAP data sources. The other parts are allowed for future extensibility.

See the informative material for background information on OLAP and the various CUBE* functions.

[*Example*: The CUBEMEMBER() function is used to return a specific member from an OLAP cube. The metadata will express the connection name (used as a friendly identifier for the external data connection to the OLAP server and cube), the mdx statement retrieving that member, and a set of operational attributes of the metadata that sepcify how it behaves in the spreadsheet application (i.e., whether it propagates through formula assignment, is able to be copy/pasted, etc).

```
<metadata xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main">
```

```
  <metadataTypes count="1">
    <metadataType name="XLMDX" minSupportedVersion="120000" copy="1"
      pasteAll="1" pasteValues="1" merge="1" splitFirst="1" rowColShift="1"
      clearFormats="1" clearComments="1" assign="1" coerce="1"/>
  </metadataTypes>
  <metadataStrings count="2">
    <s v="My Connection"/>
    <s v="[Measures].[Internet Sales Amount]"/>
  </metadataStrings>
  <mdxMetadata count="1">
    <mdx n="0" f="m">
      <t c="1">
        <n x="1"/>
      </t>
    </mdx>
  </mdxMetadata>
  <valueMetadata count="1">
    <bk>
      <rc t="1" v="0"/>
    </bk>
  </valueMetadata>
</metadata>
```

As seen above, the metadata string table contains two entries: the name of the connection (My Connection), and the expression that returns the Internet Sales Amount member from the cube. The metadataType specifies that the metadata persists with assignment, cell merging, copy/pasting, shifting rows/columns, when the formatting or comments are deleted from the cell, and is assigned to the upper left most cell if a merged cell is split. In the valueMetadata collection, the metadata block specifies that the first `metadataType` is used, and indexes the first (0th) entry in the mdxMetadata collection. This mdx element in the mdxMetadata collection in turn specifies the cube function type (m= cube member) and an index into the string table that specifies the connection name. It also contains a tuple (t) element which specifies, via index into the string table, which tuple is returned. *end example*]

[*Note*: When copying a cell with metadata, and the cell contains an array formula, each pasted cell shall contain the value from the corresponding position in the array and should contain the metadata corresponding to that cell. *end note*]

3.9.1 bk (Metadata Block)

This element represents a block of metadata records.

Parent Elements
cellMetadata (§3.9.3); valueMetadata (§3.9.17)

Child Elements	Subclause
rc (Metadata Record)	§3.9.15

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MetadataBlock">
   <sequence>
      <element name="rc" type="CT_MetadataRecord" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.9.2 bk (Future Metadata Block)

This element represents a block of future metadata information. This is a location for storing feature extension information.

Parent Elements
futureMetadata (§3.9.4)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FutureMetadataBlock">
   <sequence>
      <element name="extLst" minOccurs="0" maxOccurs="1" type="CT_ExtensionList"/>
   </sequence>
</complexType>
```

3.9.3 cellMetadata (Cell Metadata)

This element represents cell metadata information. Cell metadata is information metadata about a specific cell, and it stays tied to that cell position.

[*Note*: Applications should not use this for storing metadata, but instead us `valueMetadata`. Cell metadata is included for storing information from future application. *end note*]

Parent Elements
metadata (§3.9.8)

Child Elements	Subclause
bk (Metadata Block)	§3.9.1

Attributes	Description

Attributes	Description
count (Metadata Block Count)	Number of blocks of metadata records. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MetadataBlocks">
    <sequence>
        <element name="bk" type="CT_MetadataBlock" minOccurs="1" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="count" type="xsd:unsignedInt" use="optional" default="0"/>
</complexType>
```

3.9.4 futureMetadata (Future Metadata)

This element represents future metadata information.

Future data storage areas are xml storage areas that a later version of the spreadsheet application can store data into. So a V2 spreadsheet application may store data for new features that don't exist in the V1 version in a future storage area when saving to a format that the V1 version can open. The V1 version may be able to open the file, but won't necessarily be able to understand data that is stored in a future storage area. So the V1 version may ignore this data, but still round trip it in the file format so that V2 and V1 users can collaborate on the same spreadsheet.

Parent Elements
metadata (§3.9.8)

Child Elements	Subclause
bk (Future Metadata Block)	§3.9.2
extLst (Future Feature Data Storage Area)	§3.2.10

Attributes	Description
count (Future Metadata Block Count)	Number of future metadata blocks. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
name (Metadata Type Name)	Metadata type name. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FutureMetadata">
   <sequence>
      <element name="bk" type="CT_FutureMetadataBlock" minOccurs="0" maxOccurs="unbounded"/>
      <element name="extLst" minOccurs="0" maxOccurs="1" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="name" type="ST_Xstring" use="required"/>
   <attribute name="count" type="xsd:unsignedInt" use="optional" default="0"/>
</complexType>
```

3.9.5 k (KPI MDX Metadata)

This element represents key performance indicator (KPI) MDX metadata. A KPI is typically an image that represents the state of some specific business measure at a given point in time. For instance, an image of a green traffic light indicating that customer satisfaction is good.

Parent Elements
mdx (§3.9.6)

Attributes	Description
n (Member Unique Name Index)	Index of member unique name in string store. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
np (KPI Index)	Index of key performance indicator name in string store. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
p (KPI Property)	Key performance indicator property. The possible values for this attribute are defined by the ST_MdxKPIProperty simple type (§3.18.47).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MdxKPI">
   <attribute name="n" type="xsd:unsignedInt" use="required"/>
   <attribute name="np" type="xsd:unsignedInt" use="required"/>
   <attribute name="p" type="ST_MdxKPIProperty" use="required"/>
</complexType>
```

3.9.6 mdx (MDX Metadata Record)

This element represents a single record of MDX metadata information which can express a tuple, KPI, set, or member property.

Parent Elements

Parent Elements
mdxMetadata (§3.9.7)

Child Elements	Subclause
k (KPI MDX Metadata)	§3.9.5
ms (Set MDX Metadata)	§3.9.12
p (Member Property MDX Metadata)	§3.9.14
t (Tuple MDX Metadata)	§3.9.16

Attributes	Description
f (Cube Function Tag)	This is an enumeration representing the type of the calling cube function from the spreadsheet. The possible values for this attribute are defined by the ST_MdxFunctionType simple type (§3.18.46).
n (Connection Name Index)	The zero based index of connection name in metadata string store, `metadataStrings`. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Mdx">
   <choice minOccurs="1" maxOccurs="1">
      <element name="t" type="CT_MdxTuple"/>
      <element name="ms" type="CT_MdxSet"/>
      <element name="p" type="CT_MdxMemeberProp"/>
      <element name="k" type="CT_MdxKPI"/>
   </choice>
   <attribute name="n" type="xsd:unsignedInt" use="required"/>
   <attribute name="f" type="ST_MdxFunctionType" use="required"/>
</complexType>
```

3.9.7 mdxMetadata (MDX Metadata Information)

This element represents a collection of specific MDX metadata records for the spreadsheet. This is used to build up the members, sets, tuples, KPIs, and member properties for the spreadsheet.

Parent Elements
metadata (§3.9.8)

Child Elements	Subclause

Child Elements	Subclause
mdx (MDX Metadata Record)	§3.9.6

Attributes	Description
count (MDX Metadata Record Count)	Number of MDX metadata metadata records. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MdxMetadata">
   <sequence>
      <element name="mdx" type="CT_Mdx" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional" default="0"/>
</complexType>
```

3.9.8 metadata (Metadata)

This element represents the root node for all metadata information in the spreadsheet.

Parent Elements
Root element of SpreadsheetML Metadata part

Child Elements	Subclause
cellMetadata (Cell Metadata)	§3.9.3
extLst (Future Feature Data Storage Area)	§3.2.10
futureMetadata (Future Metadata)	§3.9.4
mdxMetadata (MDX Metadata Information)	§3.9.7
metadataStrings (Metadata String Store)	§3.9.9
metadataTypes (Metadata Types Collection)	§3.9.11
valueMetadata (Value Metadata)	§3.9.17

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Metadata">
    <sequence>
        <element name="metadataTypes" type="CT_MetadataTypes" minOccurs="0" maxOccurs="1"/>
        <element name="metadataStrings" type="CT_MetadataStrings" minOccurs="0" maxOccurs="1"/>
        <element name="mdxMetadata" type="CT_MdxMetadata" minOccurs="0" maxOccurs="1"/>
        <element name="futureMetadata" type="CT_FutureMetadata" minOccurs="0" maxOccurs="unbounded"/>
        <element name="cellMetadata" type="CT_MetadataBlocks" minOccurs="0" maxOccurs="1"/>
        <element name="valueMetadata" type="CT_MetadataBlocks" minOccurs="0" maxOccurs="1"/>
        <element name="extLst" minOccurs="0" maxOccurs="1" type="CT_ExtensionList"/>
    </sequence>
</complexType>
```

3.9.9 metadataStrings (Metadata String Store)

This element represents the metadata string store. This is a collection of strings that are used as a resource for the rest of the metadata part. It contains all the required OLAP strings used in the spreadsheet including the connection name, as well as mdx expressions identifying specific members and sets. It is indexed from individual metadata records so that the records can use these strings to build up the necessary mdx statements to retrieve the correct data from the OLAP cube.

Parent Elements
metadata (§3.9.8)

Child Elements	Subclause
s (Character Value)	§3.13.8

Attributes	Description
count (MDX Metadata String Count)	Number of records in the string store. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MetadataStrings">
    <sequence>
        <element name="s" type="CT_XStringElement" minOccurs="1" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="count" type="xsd:unsignedInt" use="optional" default="0"/>
</complexType>
```

3.9.10 metadataType (Metadata Type Information)

This element represents information about metadata on cells - it defines a specific set of behaviors that the metadata shall adhere to when subject to other spreadsheet operations.

In general, many of these attributes represent operations that can be performed on a cell that allow the metadata to remain associated with the cell. Operations that are set to 0 or false, will cause the metadata to be disassociated from the cell when that operation is performed.

Parent Elements
metadataTypes (§3.9.11)

Attributes	Description
adjust (Adjust Metadata)	A Boolean flag indicating that metadata corresponding to a particular cell needs to be notified when that cell's location is changed. [*Note*: This is included in the file format for future extensibility.*end note*] The possible values for this attribute are defined by the XML Schema boolean datatype.
assign (Metadata Formula Assignment)	A Boolean flag indicating whether metadata is propagated by formula assignment operation. True when metadata should be propagated by assignment, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.
cellMeta (Cell Metadata)	A Boolean flag indicating whether metadata is cell metadata. True when the metadata is cell metadata, false otherwise - in the false case it is considered to be value metadata. The possible values for this attribute are defined by the XML Schema boolean datatype.
clearAll (Metadata Clear All)	A Boolean flag indicating whether metadata survives a "Clear: All" operation. True if the metadata persists after a clear all, false otherwise. The Clear operations can be implemented by the run time application to provide an easy way to allow users to delete everything from a cell (Clear: All), remove only comments (Clear: Comments), only remove formats (Clear: Formats), or only remove the contents but leave the comments and formatting (Clear: Contents). Note these operations can also be performed by the user manually deleting each item. The possible values for this attribute are defined by the XML Schema boolean datatype.
clearComments (Metadata Clear Comments)	A Boolean flag indicating whether metadata remains after comments have been cleared from the cell. True if the metadata persists after Clear:Comments, false otherwise. The Clear operations can be implemented by the run time application to provide an easy way to allow users to delete everything from a cell (Clear: All), remove only comments (Clear: Comments), only remove formats (Clear: Formats), or only remove the contents but leave the comments and formatting (Clear: Contents). Note these operations can also be performed by the user manually deleting each item. The possible values for this attribute are defined by the XML Schema boolean datatype.
clearContents	A Boolean flag indicating whether metadata remains after the contents of a cell are

Attributes	Description
(Metadata Clear Contents)	removed. True if metadata persists after a "Clear: Contents" action, false otherwise. The Clear operations can be implemented by the run time application to provide an easy way to allow users to delete everything from a cell (Clear: All), remove only comments (Clear: Comments), only remove formats (Clear: Formats), or only remove the contents but leave the comments and formatting (Clear: Contents). Note these operations can also be performed by the user manually deleting each item. The possible values for this attribute are defined by the XML Schema boolean datatype.
clearFormats (Metadata Clear Formats)	A Boolean flag indicating whether metadata remains after formatting is removed from a cell. True if metadata persists after a "Clear: Formats", false otherwise. The Clear operations can be implemented by the run time application to provide an easy way to allow users to delete everything from a cell (Clear: All), remove only comments (Clear: Comments), only remove formats (Clear: Formats), or only remove the contents but leave the comments and formatting (Clear: Contents). Note these operations can also be performed by the user manually deleting each item. The possible values for this attribute are defined by the XML Schema boolean datatype.
coerce (Metadata Coercion)	A Boolean flag indicating whether value metadata can be removed when this metadata data type is coerced to another type. True if the value metadata is removed upon coercion, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.
copy (Metadata Copy)	A Boolean flag indicating whether metadata is copied with a cell. True if the metadata is copied to other cells when this cell is copied, false otherwise. This shall be set to `true` if the paste attributes for the `metadataType` are going to be used. The possible values for this attribute are defined by the XML Schema boolean datatype.
delete (Metadata Cell Value Delete)	A Boolean flag indicating whether metadata survives deletion of a cell value. True when the metadata persists after the deletion of a cell value, false otherwise. This attribute is equivalent to the `clearContents` attribute. The possible values for this attribute are defined by the XML Schema boolean datatype.
edit (Metadata Edit)	A Boolean flag indicating whether metadata survives the editing of the cell's value. True if the metadata remains unchanged after the cell's value edit, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.
ghostCol (Metadata Ghost Column)	A Boolean flag indicating whether metadata is copied to/from a ghost column. True when the metadata is copied to/from a ghost column, false otherwise.

Attributes	Description
	A ghost column is a single column that exists for the row header. It is not displayed to the end user. It is used to store default formatting for an entire row (i.e. the row gets the formatting for the corresponding cell in the ghost column). For instance, when an entire row is selected and a cell color is applied, this is stored once for the cell in the ghost column instead of for each cell in the row. The possible values for this attribute are defined by the XML Schema boolean datatype.
ghostRow (Metadata Ghost Row)	A Boolean flag indicating whether metadata is copied to/from a ghost row. True when the metadata is copied to/from a ghost row, false otherwise. A ghost row is a single row that exists for the column header. It is not displayed to the end user. It is used to store default formatting for an entire column (i.e. the column gets the formatting for the corresponding cell in the ghost row). For instance, when an entire column is selected and a cell color is applied, this is stored once for the cell in the ghost row instead of for each cell in the column. The possible values for this attribute are defined by the XML Schema boolean datatype.
merge (Metadata Merge)	A Boolean flag indicating whether metadata survives cell merge. True if the metadata persists after a cell merge, false otherwise. It is up to the spreadsheet application on how to deal with conflicts when two cells that each have metadata are merged. The guidance here is to treat it the same as a 'regular' cell merge with the default behavior being that the data in the upper left cell wins. The possible values for this attribute are defined by the XML Schema boolean datatype.
minSupportedVersion (Minimum Supported Version)	The earliest version of the spreadsheet application that supports this metadata type. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
name (Metadata Type Name)	Represents the name of this particular metadata type. This name shall be unique amongst all other `metadataTypes`. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
pasteAll (Metadata Paste All)	A Boolean flag indicating whether metadata is populated to a new cell by "Paste: All". True when the metadata is populated on a Paste:All, false otherwise. Paste:All and regular paste should be implemented so that they are equivalent by the spreadsheet application. The `copy` flag shall be set to `true` for this paste behavior to be respected. Note: the spreadsheet application can implement special pasting behavior, such as pasting everything from a cell (paste all/normal paste), pasting only borders, pasting only comments, or pasting only any other specific cell property.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema boolean datatype.
pasteBorders (Metadata Paste Borders)	A Boolean flag indicating whether metadata is populated with Paste: Borders. True when the metadata is populated when only borders are pasted, false otherwise. The copy flag shall be set to `true` for this paste behavior to be respected. [*Note*: The spreadsheet application can implement special pasting behavior, such as pasting everything from a cell (paste all/normal paste), pasting only borders, pasting only comments, or pasting only any other specific cell property. *end note*] The possible values for this attribute are defined by the XML Schema boolean datatype.
pasteColWidths (Metadata Paste Column Widths)	A Boolean flag indicating whether metadata is populated by Paste: Column Widths. True if the metadata is populated when only column widths are pasted, false otherwise. The copy flag shall be set to `true` for this paste behavior to be respected. [*Note*: the spreadsheet application can implement special pasting behavior, such as pasting everything from a cell (paste all/normal paste), pasting only borders, pasting only comments, or pasting only any other specific cell property. *end note*] The possible values for this attribute are defined by the XML Schema boolean datatype.
pasteComments (Metadata Paste Comments)	A Boolean flag indicating whether metadata is populated by Paste: Comments. True when metadata is populated when only comments are pasted, false otherwise. The copy flag shall be set to `true` for this paste behavior to be respected. [*Note*: the spreadsheet application can implement special pasting behavior, such as pasting everything from a cell (paste all/normal paste), pasting only borders, pasting only comments, or pasting only any other specific cell property. *end note*] The possible values for this attribute are defined by the XML Schema boolean datatype.
pasteDataValidation (Metadata Paste Data Validation)	A Boolean flag indicating whether metadata is populated by Paste: Validation. True when metadata is populated when only data validation is pasted, false otherwise. The copy flag shall be set to `true` for this paste behavior to be respected. [*Note*: the spreadsheet application can implement special pasting behavior, such as pasting everything from a cell (paste all/normal paste), pasting only borders, pasting only comments, or pasting only any other specific cell property. *end note*] The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description
pasteFormats (Metadata Paste Formats)	A Boolean flag indicating whether metadata is populated by Paste Special: Formats. True when metadata is populated when only formatting is pasted, false otherwise. The copy flag shall be set to true for this paste behavior to be respected. [*Note*: the spreadsheet application can implement special pasting behavior, such as pasting everything from a cell (paste all/normal paste), pasting only borders, pasting only comments, or pasting only any other specific cell property. *end note*] The possible values for this attribute are defined by the XML Schema boolean datatype.
pasteFormulas (Metadata Paste Formulas)	A Boolean flag indicating whether metadata is populated by Paste: Formulas. True when the metadata is populated when only formulas are pasted, false otherwise. The copy flag shall be set to true for this paste behavior to be respected. [*Note*: the spreadsheet application can implement special pasting behavior, such as pasting everything from a cell (paste all/normal paste), pasting only borders, pasting only comments, or pasting only any other specific cell property. *end note*] The possible values for this attribute are defined by the XML Schema boolean datatype.
pasteNumberFormats (Metadata Paste Number Formats)	A Boolean flag indicating whether metadata is populated with Paste: Number Formats. True when metadata is populated when only number formatting is pasted, false otherwise. The copy flag shall be set to true for this paste behavior to be respected. [*Note*: the spreadsheet application can implement special pasting behavior, such as pasting everything from a cell (paste all/normal paste), pasting only borders, pasting only comments, or pasting only any other specific cell property. *end note*] The possible values for this attribute are defined by the XML Schema boolean datatype.
pasteValues (Metadata Paste Special Values)	A Boolean flag indicating whether metadata is populated by Paste: Values. The possible values for this attribute are defined by the XML Schema boolean datatype.
rowColShift (Metadata Insert Delete)	A Boolean flag indicating whether metadata survives shifting due to row/column insertion/deletion. True if the metadata persists after the cell has been shifted, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.
splitAll (Metadata Split All)	A Boolean flag indicating whether a merged cell split action has its metadata copied to all of the resulting cells. True if the metadata is copied to all new cells resulting from a split, false otherwise. If splitFirst is also set to true, splitAll wins - that is all the cells shall have the

Attributes	Description
	metadata copied to them. The possible values for this attribute are defined by the XML Schema boolean datatype.
splitFirst (Meatadata Split First)	A Boolean flag indicating whether when a merged cell is split its metadata is copied to only the first resulting cell. True when the metadata from a split cell is only copied to the first resulting cell, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MetadataType">
    <attribute name="name" type="ST_Xstring" use="required"/>
    <attribute name="minSupportedVersion" type="xsd:unsignedInt" use="required"/>
    <attribute name="ghostRow" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="ghostCol" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="edit" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="delete" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="copy" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="pasteAll" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="pasteFormulas" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="pasteValues" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="pasteFormats" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="pasteComments" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="pasteDataValidation" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="pasteBorders" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="pasteColWidths" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="pasteNumberFormats" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="merge" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="splitFirst" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="splitAll" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="rowColShift" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="clearAll" type="xsd:boolean" default="false"/>
    <attribute name="clearFormats" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="clearContents" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="clearComments" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="assign" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="coerce" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="adjust" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="cellMeta" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.9.11 metadataTypes (Metadata Types Collection)

This element is a collection of metadata types.

Parent Elements
metadata (§3.9.8)

Child Elements	Subclause
metadataType (Metadata Type Information)	§3.9.10

Attributes	Description
count (Metadata Type Count)	Number of metadata types. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MetadataTypes">
   <sequence>
      <element name="metadataType" type="CT_MetadataType" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional" default="0"/>
</complexType>
```

3.9.12 ms (Set MDX Metadata)

This element represents an MDX set.

Parent Elements
mdx (§3.9.6)

Child Elements	Subclause
n (Member Unique Name Index)	§3.9.13

Attributes	Description
c (Sort By Member Index Count)	Number of sort-by member indices. This is essentially the number of coordinates in the cube that this member is defined by. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
ns (Set Definition Index)	Zero based index of the set definition in the metadata string store. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
o (Set Sort Order)	An enumeration specifying what sort order is used to sort the set. The possible values for this attribute are defined by the ST_MdxSetOrder simple type (§3.18.48).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MdxSet">
    <sequence>
        <element name="n" type="CT_MetadataStringIndex" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="ns" type="xsd:unsignedInt" use="required"/>
    <attribute name="c" type="xsd:unsignedInt" use="optional" default="0"/>
    <attribute name="o" type="ST_MdxSetOrder" use="optional" default="u"/>
</complexType>
```

3.9.13 n (Member Unique Name Index)

This element represents an index of a member unique name in metadata string store that is used to define the sort-by set.

Parent Elements
ms (§3.9.12); t (§3.9.16)

Attributes	Description
s (String is a Set)	A Boolean flag indicating whether this string represents a set. True if the string represents a set, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.
x (Index Value)	Value of the zero based index. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MetadataStringIndex">
    <attribute name="x" type="xsd:unsignedInt" use="required"/>
    <attribute name="s" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.9.14 p (Member Property MDX Metadata)

This element represents an MDX member property.

Parent Elements
mdx (§3.9.6)

Attributes	Description
n (Member Unique Name Index)	The zero based index of member unique name in the metadata string store.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
np (Property Name Index)	The zero based index of the property name in metadata string store. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MdxMemeberProp">
   <attribute name="n" type="xsd:unsignedInt" use="required"/>
   <attribute name="np" type="xsd:unsignedInt" use="required"/>
</complexType>
```

3.9.15 rc (Metadata Record)

This element represents a reference to a specific metadata record.

Parent Elements
bk (§3.9.1)

Attributes	Description
t (Metadata Record Type Index)	A 1-based index to the metadata record type in `metadataTypes`. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
v (Metadata Record Value Index)	A zero based index to a specific metadata record. If the corresponding metadataType has `name`="XLMDX", then this is an index to a record in mdxMetadata, otherwise this is an index to a record in the futureMetadata section whose `name` matches the name of the metadataType. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MetadataRecord">
   <attribute name="t" type="xsd:unsignedInt" use="required"/>
   <attribute name="v" type="xsd:unsignedInt" use="required"/>
</complexType>
```

3.9.16 t (Tuple MDX Metadata)

This element represents an mdx tuple. A tuple is the intersection of two or more members of distinct dimensions in the cube. For instance, the three members (product, City, month) that are used to show the data point for how many products were sold.

The spreadsheet application should allow the values for the attributes of this element to be specified by the OLAP server.

Parent Elements
mdx (§3.9.6)

Child Elements	Subclause
n (Member Unique Name Index)	§3.9.13

Attributes	Description
b (Server Formatting Bold Font)	A Boolean flag indicating whether the bold style is applied. True if bold shall be applied, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.
bc (Server Formatting Background Color)	Specifies the background color in RGB values. It is in hex and is read in the form of 0x00RRGGBB. The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§3.18.86).
c (Member Index Count)	The number of member expressions in the tuple. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
ct (Server Formatting Culture Currency)	The culture tag to use for currency number format. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
fc (Server Formatting Foreground Color)	Represents the foreground color in RGB. It is in hex and is read in the form of 0x00RRGGBB. The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§3.18.86).
fi (Server Formatting Built-In Number Format Index)	Server formatting built-in number format index. This is an index into the spreadsheet application's built in number formats that is used to specify formatting. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
i (Server Formatting Italic Font)	A Boolean flag indicating that the italic formatting shall be applied. True if italic formatting is applied, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description
si (Server Formatting String Index)	Server formatting string index in the metadata string store, used to index to a string that contains information on how to format the number. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
st (Server Formatting Strikethrough Font)	A Boolean flag indicating whether the strikethrough font style is applied. True if strikethrough shall be applied, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.
u (Server Formatting Underline Font)	A Boolean flag indicating whether the underline font style is applied. True if underline shall be applied, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MdxTuple">
   <sequence>
      <element name="n" type="CT_MetadataStringIndex" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="c" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="ct" type="ST_Xstring" use="optional"/>
   <attribute name="si" type="xsd:unsignedInt" use="optional"/>
   <attribute name="fi" type="xsd:unsignedInt" use="optional"/>
   <attribute name="bc" type="ST_UnsignedIntHex" use="optional"/>
   <attribute name="fc" type="ST_UnsignedIntHex" use="optional"/>
   <attribute name="i" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="u" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="st" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="b" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.9.17 valueMetadata (Value Metadata)

This element represents the value metadata information for the spreadsheet. It is essentially a collection of block elements that each define the value metadata for a particular cell. Cells in the workbook index into this collection, and each block element in this collection in turn references the `mdxMetadata` records.

Parent Elements
metadata (§3.9.8)

Child Elements	Subclause
bk (Metadata Block)	§3.9.1

Attributes	Description
count (Metadata Block Count)	Number of blocks of metadata records. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

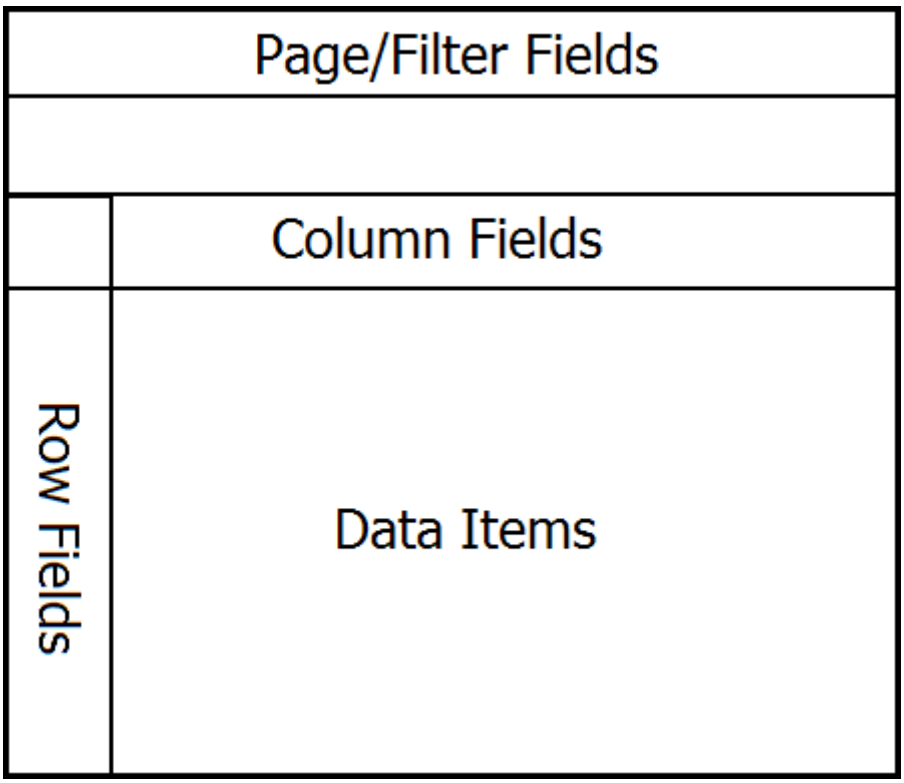
```
<complexType name="CT_MetadataBlocks">
    <sequence>
        <element name="bk" type="CT_MetadataBlock" minOccurs="1" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="count" type="xsd:unsignedInt" use="optional" default="0"/>
</complexType>
```

3.10 Pivot Tables

PivotTables display aggregated views of data easily and in an understandable layout. Hundreds or thousands of pieces of underlying information can be aggregated on row & column axes, revealing the meanings behind the data. PivotTable reports are used to organize and summarize your data in different ways. Creating a PivotTable report is about moving pieces of information around to see how they fit together. In a few gestures the pivot rows and columns can be moved into different arrangements and layouts.

A PivotTable object has a row axis area, a column axis area, a data area, and a page/report filter area. Additionally, PivotTables have a corresponding field list pane, or similar user interface, that displays all the fields of data that can be placed on one of the PivotTable areas. In SpreadsheetML, each PivotTable area maps to a collection of fields in the PivotTableDefinition that correspond to each area.

The following image shows the layout for the PivotTable areas.



[*Example*:

The following image shows a table of data in a worksheet.

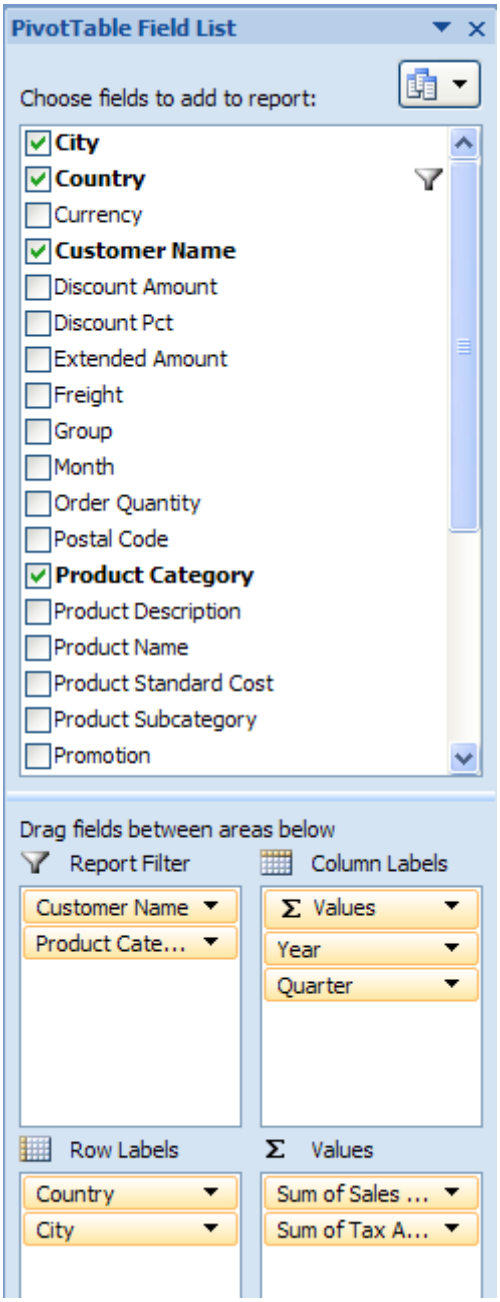
	A	C	F	H	I	O	P	Q	Z	AA	AB
1	Customer Name	Country	City	Product Category	Product Subcategory	Year	Quarter	Month	Sales Amount	Tax Amount	Freight
2	Michele Raman	Australia	Bendigo	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
3	Misty Raji	Australia	Bendigo	Bikes	Road Bikes	2001	3	July	3578.27	286.2616	89.4568
4	Tabitha E Arthur	Australia	Bendigo	Bikes	Road Bikes	2001	3	July	3578.27	286.2616	89.4568
5	Clarence D Rai	Australia	Bendigo	Bikes	Mountain Bikes	2001	3	July	3399.99	271.9992	84.9998
6	Jimmy L Moreno	Australia	Bendigo	Bikes	Mountain Bikes	2001	3	July	3399.99	271.9992	84.9998
7	Rob Verhoff	Australia	Bendigo	Bikes	Mountain Bikes	2001	3	July	3374.99	269.9992	84.3748
8	Levi Sai	Australia	Bendigo	Bikes	Road Bikes	2001	3	July	3578.27	286.2616	89.4568
9	Logan Gonzales	Australia	Brisbane	Bikes	Road Bikes	2001	3	July	3578.27	286.2616	89.4568
10	Dalton J Lee	Australia	Brisbane	Bikes	Road Bikes	2001	3	August	3578.27	286.2616	89.4568
11	Jessie J Ortega	Australia	Brisbane	Bikes	Road Bikes	2001	3	August	3578.27	286.2616	89.4568
12	Paul J. Shakespear	Australia	Caloundra	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
13	Joan R Martin	Australia	Caloundra	Bikes	Road Bikes	2001	3	September	699.0982	55.9279	17.4775
14	Casey Pal	Australia	Caloundra	Bikes	Road Bikes	2001	3	July	3578.27	286.2616	89.4568
15	Ethan G Coleman	Australia	Caloundra	Bikes	Road Bikes	2001	3	August	3578.27	286.2616	89.4568
16	Kendra Rubio	Australia	Caloundra	Bikes	Road Bikes	2001	3	August	3578.27	286.2616	89.4568
17	Bethany G Yuan	Australia	Cloverdal	Bikes	Mountain Bikes	2001	3	August	3399.99	271.9992	84.9998
18	Jasmine Wilson	Australia	Coffs Hart	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
19	Micah Wu	Australia	Coffs Hart	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
20	Warren L Zhang	Australia	Coffs Hart	Bikes	Road Bikes	2001	3	July	699.0982	55.9279	17.4775
21	Ariana Stewart	Australia	Coffs Hart	Bikes	Road Bikes	2001	3	August	3578.27	286.2616	89.4568
22	Suzanne K Lu	Australia	Coffs Hart	Bikes	Road Bikes	2001	3	August	3578.27	286.2616	89.4568
23	Randall M Rubio	Australia	Cranbourr	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
24	Deborah K Kumar	Australia	Cranbourr	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
25	Krystal Holt	Australia	Cranbourr	Bikes	Road Bikes	2001	3	July	3578.27	286.2616	89.4568
26	Patricia T Raman	Australia	Cranbourr	Bikes	Road Bikes	2001	3	August	3578.27	286.2616	89.4568
27	Wendy Dominguez	Australia	Cranbourr	Bikes	Mountain Bikes	2001	3	August	3374.99	269.9992	84.3748
28	Willie She	Australia	Darlinghu	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
29	Alan Zhu	Australia	Darlinghu	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
30	Dawn R Tang	Australia	Darlinghu	Bikes	Road Bikes	2001	3	July	3578.27	286.2616	89.4568

The following image shows a PivotTable summary of the worksheet table data.

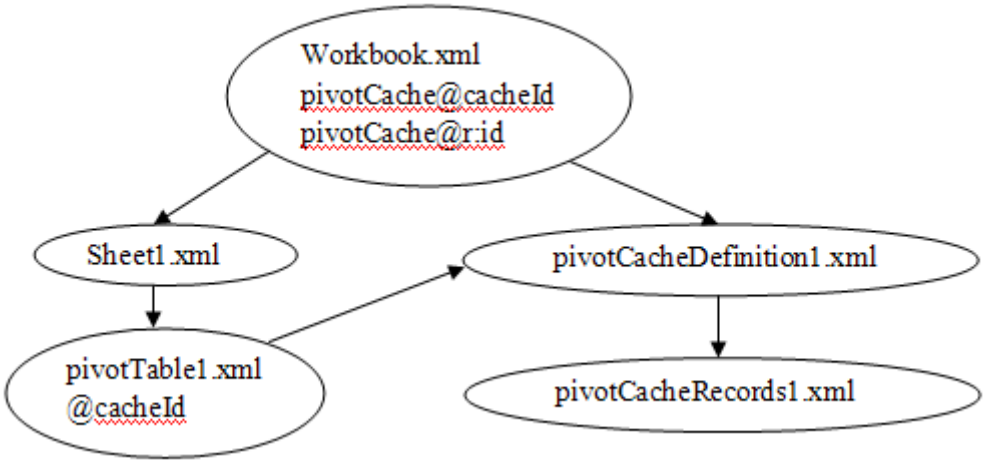
	A	B	C	D	E	F	G
1							
2		Country	(All)				
3		State	(All)				
4		City	(All)				
5							
6		Sum of Sales Amount	Column Labels				
7			2001				2001 Total
8			3			3 Total	
9		Row Labels	July	August	September		
10		Bikes	209652.9046	222538.2892	173993.5128	606184.7066	606184.7066
11		Mountain Bikes	64424.81	60899.82	10174.97	135499.6	135499.6
12		Road Bikes	145228.0946	161638.4692	163818.5428	470685.1066	470685.1066
13		Grand Total	209652.9046	222538.2892	173993.5128	606184.7066	606184.7066

The filter area consists of the "Country", "State", and "City" fields. The row area consists of the "Product Category" and "Product Subcategory" fields. "Bikes" belongs to the "Product Category" field and both "Mountain Bikes" and "Road Bikes" belong to the "Product Subcategory" field. The column consists of the "Year" ("2001"), "Quarter" ("3"), and "Month" ("July", "August", and "September") fields.

The following image shows the field list for the PivotTable in the previous image.



File Structure



The workbook points to (and owns the longevity of) the *pivotCacheDefinition* part, which in turn points to and owns the *pivotCacheRecords* part. The workbook also points to and owns the sheet part, which in turn points to and owns a *pivotTable* part definition, when a PivotTable is on the sheet. There can be multiple PivotTables on a sheet. The *pivotTable* part points to the appropriate *pivotCacheDefinition* which it is using. Since multiple PivotTables can use the same cache, the *pivotTable* part does not own the longevity of the *pivotCacheDefinition*.

The *pivotTable* part describes the particulars of the layout of the PivotTable on the sheet. It indicates what fields are on the row axis, the column axis, report filter, and values areas of the PivotTable. It also indicates formatting information about the PivotTable. If conditional formatting has been applied to the PivotTable, that is also expressed in the *pivotTable* part.

Outline of XML for pivotTableDefinition

```
<pivotTableDefinition>
  <location/>
  <pivotFields/>
  <rowFields/>
  <rowItems/>
  <colFields/>
  <colItems/>
  <pageFields/>
  <dataFields/>
  <conditionalFormats/>
  <pivotTableStyleInfo/>
</pivotTableDefinition>
```

The *pivotCacheRecords* part contains the underlying data to be aggregated. It is a cache of the source data.

Outline of XML for pivotCacheRecords

```
<pivotCacheRecords/>
  <r/>
</pivotCacheRecords>
```

The *pivotCacheDefinition* part defines each field in the *pivotCacheRecords* part, including field name and information about the data contained in the field. The *pivotCacheDefinition* part also defines pivot items that are shared among the *pivotTableDefinition* and *pivotCacheRecords* parts.

Outline of XML for pivotCacheDefinition

```
<pivotCacheDefinition>
  <cacheSource/>
  <cacheFields>
    <cacheField>
      <sharedItems>
        <d/>
      </sharedItems>
      <fieldGroup/>
    </cacheField>
  </cacheFields>
</pivotCacheDefinition>
```

3.10.1 Pivot Tables

This section describes the definition of PivotTables in SpreadsheetML.

3.10.1.1 autoSortScope (AutoSort Scope)

Represents the sorting scope for the PivotTable.

Parent Elements
pivotField (§3.10.1.69)

Child Elements	Subclause
pivotArea (Pivot Area)	§3.3.1.66

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AutoSortScope">
   <sequence>
      <element name="pivotArea" type="CT_PivotArea"/>
   </sequence>
</complexType>
```

3.10.1.2 b (Boolean)

Represents a boolean value for an item in the PivotTable.

Parent Elements
groupItems (§3.10.1.38); r (§3.10.1.77); sharedItems (§3.10.1.90)

Child Elements	Subclause
x (Member Property Index)	§3.10.1.96

Attributes	Description
c (Caption)	Specifies the caption for the item. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
cp (Member Property Count)	Specifies the number of property values for this item. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
f (Calculated Item)	Specifies a boolean value that indicates whether this item has a calculated value. A value of on, 1, or true indicates the item has a calculated value. A value of off, 0, or false indicates the item does not have a calculated value. The possible values for this attribute are defined by the XML Schema boolean datatype.
u (Unused Item)	Specifies a boolean value that indicates whether this is an unused item. The application marks an item as unused when an item is deleted from the data source. The item and associated metadata are retained in the cache until the threshold for unused items specified in missingItemsLimit is reached. A value of on, 1, or true indicates this item is unused. A value of off, 0, or false indicates this item is used. The possible values for this attribute are defined by the XML Schema boolean datatype.
v (Value)	Specifies the value of the item. This attribute is required. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <sequence>
      <element name="x" minOccurs="0" maxOccurs="unbounded" type="CT_X"/>
   </sequence>
   <attribute name="v" use="required" type="xsd:boolean"/>
   <attribute name="u" type="xsd:boolean"/>
   <attribute name="f" type="xsd:boolean"/>
   <attribute name="c" type="ST_Xstring"/>
   <attribute name="cp" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.3 cacheField (PivotCache Field)

Represent a single field in the PivotCache. This definition contains information about the field, such as its source, data type, and location within a level or hierarchy. The sharedItems complex type stores additional information about the data in this field. If there are no shared items, then values are stored directly in the pivotCacheRecords part.

[*Example*:

```
<cacheField name="Group" numFmtId="0">
  <sharedItems count="3">
    <s v="Pacific"/>
    <s v="North America"/>
    <s v="Europe"/>
  </sharedItems>
</cacheField>
```

end example]

Parent Elements
cacheFields (§3.10.1.4)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
fieldGroup (Field Group Properties)	§3.10.1.30
mpMap (Member Properties Map)	§3.10.1.58
sharedItems (Shared Items)	§3.10.1.90

Attributes	Description
caption (PivotCache Field Caption)	Specifies the caption of the cache field.

Attributes	Description
	The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
databaseField (Database Field)	Specifies a boolean value that indicates whether this field came from the source database rather having been created by the application. A value of on, 1, or true indicates the field is from the source database. A value of off, 0, or false indicates the field was created by the application. [*Note*: This attribute could be used for a defined grouped or calculated field. In this case, source database fields should precede defined grouped or calculated fields. *end note*] The possible values for this attribute are defined by the XML Schema boolean datatype.
formula (Calculated Field Formula)	Specifies the formula for the calculated field. This formula is specified by the end-user. Calculated fields can perform calculations by using the contents of other fields in the PivotTable. In formulas you create for calculated fields or calculated items, you can use operators and expressions as you do in other worksheet formulas. You can use constants and refer to data from the PivotTable, but you cannot use cell references or defined names. You cannot use worksheet functions that require cell references or defined names as arguments, and you cannot use array functions. Further behaviors and restrictions apply to formulas for calculated fields: • Formulas for calculated fields operate on the sum of the underlying data for any fields in the formula. For example, the formula =Sales * 1.2 multiplies the sum of the sales for each type and region by 1.2; it does not multiply each individual sale by 1.2 and then sum the multiplied amounts. • Formulas cannot refer to totals. For more information about formulas see §3.17 in Formulas. For more information about defined names see §3.2.6 in Workbook. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
hierarchy (Hierarchy)	Specifies the hierarchy that this field is part of. The possible values for this attribute are defined by the XML Schema int datatype.
level (Hierarchy Level)	Specifies the hierarchy level that this field is part of. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
mappingCount (Member Property	Specifies the number of property mappings for this field.

Attributes	Description
Count)	The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
memberPropertyField (Member Property Field)	Specifies a boolean value that indicates whether the field contains OLAP member property information. A value of on, 1, or true indicates this field contains OLAP member property information. A value of off, 0, or false indicates this field does not contain OLAP member property information. The possible values for this attribute are defined by the XML Schema boolean datatype.
name (PivotCache Field Name)	Specifies the name of the cache field. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
numFmtId (Number Format Id)	Specifies the number format that is applied to all items in the field. Number formats are written to the styles part. For more information see §3.8.31 in Styles. Note: Formatting information provided by cell table and by PivotTable need not agree. If the two formats differ, the cell-level formatting takes precedence. If you change the layout of the PivotTable, the PivotTable formatting will then take precedence. The possible values for this attribute are defined by the ST_NumFmtId simple type (§3.18.49).
propertyName (Property Name)	Specifies the name of the property if this field is an OLAP property field. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
serverField (Server-based Field)	Specifies a boolean value that indicates whether the field is a server-based page field. A value of on, 1, or true indicates this field is a server-based page field. A value of off, 0, or false indicates this field is not a server-based page field. Note: this attribute applies to ODBC sources only. The possible values for this attribute are defined by the XML Schema boolean datatype.
sqlType (SQL Data Type)	Specifies the SQL data type of the field. This attribute stores an ODBC data type and applies to ODBC data sources only. A value is supplied for this attribute only if it is provided to the application. The following are data types supported by ODBC. For a more information, see the ODBC specification. • 0 SQL_UNKNOWN_TYPE

Attributes	Description
	• 1 SQL_CHAR • 2 SQL_VARCHAR • -1 SQL_LONGVARCHAR • -8 SQL_WCHAR • -9 SQL_WVARCHAR • -10 SQL_WLONGVARCHAR • 3 SQL_DECIMAL • 2 SQL_NUMERIC • 5 SQL_SMALLINT • 4 SQL_INTEGER • 7 SQL_REAL • 6 SQL_FLOAT • 8 SQL_DOUBLE • -7 SQL_BIT • -6 SQL_TINYINT • -5 SQL_BIGINT • -2 SQL_BINARY • -3 SQL_VARBINARY • -4 SQL_LONGVARBINARY • 9 SQL_TYPE_DATE or SQL_DATE • 10 SQL_TYPE_TIME or SQL_TIME • 11 SQL_TYPE_TIMESTAMP or SQL_TIMESTAMP • 102 SQL_INTERVAL_MONTH • 101 SQL_INTERVAL_YEAR • 107 SQL_INTERVAL_YEAR_TO_MONTH • 103 SQL_INTERVAL_DAY • 104 SQL_INTERVAL_HOUR • 105 SQL_INTERVAL_MINUTE • 106 SQL_INTERVAL_SECOND • 108 SQL_INTERVAL_DAY_TO_HOUR • 109 SQL_INTERVAL_DAY_TO_MINUTE • 110 SQL_INTERVAL_DAY_TO_SECOND • 111 SQL_INTERVAL_HOUR_TO_MINUTE • 112 SQL_INTERVAL_HOUR_TO_SECOND • 113 SQL_INTERVAL_MINUTE_TO_SECOND • -11 SQL_GUID • -20 SQL_SIGNED_OFFSET • -22 SQL_UNSIGNED_OFFSET The possible values for this attribute are defined by the XML Schema int datatype.
uniqueList (Unique List Retrieved)	Specifies a boolean value that indicates whether the application was able to get a list of unique items for the field. The attribute only applies to PivotTables that use ODBC and is intended to be used in conjunction with optimization features in the application. For example, the application can optimize memory usage when populating PivotCache

Attributes	Description
	records if it has a list of unique items for a field before all the records are retrieved from ODBC. A value of on, 1, or true indicates the application was able to get a list of unique values for the field. A value of off, 0, or false indicates the application was unable to get a list of unique values for the field. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CacheField">
   <sequence>
      <element name="sharedItems" type="CT_SharedItems" minOccurs="0" maxOccurs="1"/>
      <element name="fieldGroup" minOccurs="0" type="CT_FieldGroup"/>
      <element name="mpMap" minOccurs="0" maxOccurs="unbounded" type="CT_X"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="name" type="ST_Xstring" use="required"/>
   <attribute name="caption" type="ST_Xstring" use="optional"/>
   <attribute name="propertyName" type="ST_Xstring" use="optional"/>
   <attribute name="serverField" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="uniqueList" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="numFmtId" type="ST_NumFmtId" use="optional"/>
   <attribute name="formula" type="ST_Xstring" use="optional"/>
   <attribute name="sqlType" type="xsd:int" use="optional" default="0"/>
   <attribute name="hierarchy" type="xsd:int" use="optional" default="0"/>
   <attribute name="level" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="databaseField" type="xsd:boolean" default="true"/>
   <attribute name="mappingCount" type="xsd:unsignedInt" use="optional"/>
   <attribute name="memberPropertyField" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.10.1.4 cacheFields (PivotCache Fields)

Represents the collection of field definitions in the source data.

[*Example*:

```
<cacheFields count="1">
  <cacheField name="Group" numFmtId="0">
    <sharedItems count="3">
      <s v="One"/>
      <s v="Two"/>
      <s v="Three"/>
    </sharedItems>
  </cacheField>
</cacheFields>
```

end example]

Parent Elements
pivotCacheDefinition (§3.10.1.67)

Child Elements	Subclause
cacheField (PivotCache Field)	§3.10.1.3

Attributes	Description
count (Field Count)	Specifies the number of fields in the cache. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CacheFields">
   <sequence>
      <element name="cacheField" type="CT_CacheField" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.5 cacheHierarchies (PivotCache Hierarchies)

Represents the collection of OLAP hierarchies in the PivotCache.

[*Example*:

```
<cacheHierarchies count="2">
  <cacheHierarchy uniqueName="[Account].[Account]" caption="Account"
    attribute="1" keyAttribute="1"
    defaultMemberUniqueName="[Account].[Account].[All Accounts]"
    allUniqueName="[Account].[Account].[All Accounts]"
    dimensionUniqueName="[Account]" count="0"/>
```

```
  <cacheHierarchy uniqueName="[Account].[Account Number]" caption="Account
    Number" attribute="1" defaultMemberUniqueName="[Account].[Account
    Number].[All Accounts]" allUniqueName="[Account].[Account Number].[All
    Accounts]" dimensionUniqueName="[Account]" count="0"/>
</cacheHierarchies>
```

Parent Elements
pivotCacheDefinition (§3.10.1.67)

Child Elements	Subclause
cacheHierarchy (PivotCache Hierarchy)	§3.10.1.6

Attributes	Description
count (Hierarchy Count)	Specifies the number of OLAP hierarchies in the cache. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CacheHierarchies">
   <sequence>
      <element name="cacheHierarchy" minOccurs="0" maxOccurs="unbounded" type="CT_CacheHierarchy"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.6 cacheHierarchy (PivotCache Hierarchy)

Represents an OLAP hierarchy in the PivotCache.

[*Example*:

```
<cacheHierarchy uniqueName="[Account].[Account Number]" caption="Account Number"
  attribute="1" defaultMemberUniqueName="[Account].[Account Number].[All
  Accounts]" allUniqueName="[Account].[Account Number].[All Accounts]"
  dimensionUniqueName="[Account]" count="0"/>
```

end example]

Parent Elements
cacheHierarchies (§3.10.1.5)

Child Elements	Subclause

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
fieldsUsage (Fields Usage)	§3.10.1.31
groupLevels (OLAP Grouping Levels)	§3.10.1.40

Attributes	Description
allCaption (Display Name of 'All')	Specifies the display name of the "all" member of this hierarchy. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
allUniqueName (Unique Name of 'All')	Specifies the unique name of the "all" member of this hierarchy. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
attribute (Attribute Hierarchy)	Specifies a boolean value that indicates whether this hierarchy is an attribute hierarchy. An attribute hierarchy is an OLAP member that is exposed as a flat, single-level hierarchy on the OLAP server. A value of on, 1, or true indicates this hierarchy is an attribute hierarchy. A value of off, 0, or false indicates this hierarchy is not an attribute hierarchy. The possible values for this attribute are defined by the XML Schema boolean datatype.
caption (Hierarchy Display Name)	Specifies the display name of the hierarchy. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
count (Levels Count)	Specifies the number of levels in this hierarchy. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
defaultMemberUniqueName (Default Member Unique Name)	Specifies the unique name of the default member of this hierarchy The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
dimensionUniqueName (Dimension Unique Name)	Specifies the unique name of the dimension to which this hierarchy belongs. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
displayFolder (Display Folder)	Specifies the display folder in which this hierarchy should be displayed. The possible values for this attribute are defined by the ST_Xstring simple type

Attributes	Description
	(§3.18.96).
hidden (Hidden)	Specifies a boolean value that indicates whether the hierarchy is hidden. A value of on, 1, or true indicates this hierarchy is hidden. The possible values for this attribute are defined by the XML Schema boolean datatype.
iconSet (KPI Icon Set)	Specifies the icon set to use to visualize a KPI trend or status expression. PivotTables use the icon sets available for conditional formatting in SpreadsheetML. See associated simple type definition for details. The following values are used by PivotTables: • no value: default iconset. For status KPI this corresponds to 3 traffic lights. For trend KPI this corresponds to 3-arrows. • 1: Variance Arrow - 3 arrow. • 2: 3 arrows • 3: Status Arrow Ascending - 5 arrows. • 4: Status Arrow Descending - 5 arrows • 5: Standard Arrow - 5 arrows gray. • 6: Traffic Light Single - 3 traffic lights 1. • 7: Traffic Light, Traffic Light Multiple - 3 traffic lights 2. • 8: Gauge Ascending - 5 quarters. • 9: Gauge Descending - 5 quarters. • 10: Thermometer, Cylinder, Smiley Face - 3 signs. • 11: Road Signs - 3 symbols. The possible values for this attribute are defined by the XML Schema int datatype.
keyAttribute (Key Attribute Hierarchy)	Specifies a boolean value that indicates whether this hierarchy is the key attribute hierarchy in an OLAP dimension. A value of on, 1, or true indicates this hierarchy is the key attribute hierarchy in an OLAP dimension. A value of off, 0, or false indicates this hierarchy is not a key attribute hierarchy. The possible values for this attribute are defined by the XML Schema boolean datatype.
measure (Measure Hierarchy)	Specifies a boolean value that indicates whether this hierarchy is a measure. A value of on, 1, or true indicates this hierarchy is a measure. A value of off, 0, or false indicates this hierarchy is not a measure. The possible values for this attribute are defined by the XML Schema boolean datatype.
measureGroup (Measure Group Name)	Specifies the name of the measure group to which this hierarchy belongs. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

Attributes	Description
measures (Measures)	Specifies a boolean value that indicates whether this hierarchy contains all the measures. A value of on, 1, or true indicates this hierarchy contains all the measures. A value of off, 0, or false indicates this hierarchy does not contain all the measures. The possible values for this attribute are defined by the XML Schema boolean datatype.
memberValueDatatype (Member Value Data Type)	Specifies the data type of the member value. This attribute stores an OLEDB data type. [*Note*: Data connectivity can use a number of different technologies. One example of potential values stored in this attribute can be found at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/oledb/htm/oledbtype_indicators.asp *end note*] `memberValueDataType` is stored for key attribute hierarchies in order to tell when the application will offer date filtering instead of label filtering in OLAP PivotTables. Date filtering is only offered when the data type is Date/Time. `memberValueDatatype="7"` indicates a date/time data type. The possible values for this attribute are defined by the XML Schema unsignedShort datatype.
oneField (One Field)	Specifies a boolean value that indicates whether this hierarchy is associated with only one field due to its position in the view. A value of on, 1, or true indicates this hierarchy is associated with only one field. A value of off, 0, or false indicates this field is not restricted to only one association due to its position in the user interface. The possible values for this attribute are defined by the XML Schema boolean datatype.
parentSet (Parent Set)	Specifies the parent hierarchy of the set. If the attribute is missing it means that the parent hierarchy is unknown or doesn't exist in the cache. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
set (Set)	Specifies a boolean value that indicates whether this hierarchy is a set. A value of on, 1, or true indicates this hierarchy is a set. A value of off, 0, or false indicates this hierarchy is not a set. The possible values for this attribute are defined by the XML Schema boolean datatype.
time (Time)	Specifies a boolean value that indicates whether this hierarchy is of type time.

Attributes	Description
	A value of on, 1, or true indicates this hierarchy is of type time. A value of off, 0, or false indicates is of a different type. The possible values for this attribute are defined by the XML Schema boolean datatype.
unbalanced (Unbalanced)	Specifies a boolean value that indicates whether this hierarchy is an unbalanced hierarchy. If value is not written, then this attribute either cannot be determined or does not apply to the current hierarchy. A value of on, 1, or true indicates this hierarchy is unbalanced. A value of off, 0, or false indicates is balanced. For more information on balanced hierarchies, see the documentation provided for your OLAP server. The possible values for this attribute are defined by the XML Schema boolean datatype.
unbalancedGroup (Unbalanced Group)	Specifies a boolean value that indicated whether the grouped version of this hierarchy is an unbalanced hierarchy. If value is not written, then this attribute either cannot be determined or does not apply to the current hierarchy. A value of on, 1, or true indicates this hierarchy is unbalanced when grouped. A value of off, 0, or false indicates is balanced when grouped. For more information on balanced hierarchies, see the documentation provided for your OLAP server. The possible values for this attribute are defined by the XML Schema boolean datatype.
uniqueName (Hierarchy Unique Name)	Specifies the unique name of the hierarchy. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CacheHierarchy">
    <sequence>
        <element name="fieldsUsage" minOccurs="0" type="CT_FieldsUsage"/>
        <element name="groupLevels" minOccurs="0" type="CT_GroupLevels"/>
        <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
    </sequence>
    <attribute name="uniqueName" use="required" type="ST_Xstring"/>
    <attribute name="caption" use="optional" type="ST_Xstring"/>
    <attribute name="measure" type="xsd:boolean" default="false"/>
    <attribute name="set" type="xsd:boolean" default="false"/>
    <attribute name="parentSet" type="xsd:unsignedInt" use="optional"/>
    <attribute name="iconSet" type="xsd:int" default="0"/>
    <attribute name="attribute" type="xsd:boolean" default="false"/>
    <attribute name="time" type="xsd:boolean" default="false"/>
    <attribute name="keyAttribute" type="xsd:boolean" default="false"/>
    <attribute name="defaultMemberUniqueName" type="ST_Xstring"/>
    <attribute name="allUniqueName" type="ST_Xstring"/>
    <attribute name="allCaption" type="ST_Xstring"/>
    <attribute name="dimensionUniqueName" type="ST_Xstring"/>
    <attribute name="displayFolder" type="ST_Xstring"/>
    <attribute name="measureGroup" type="ST_Xstring"/>
    <attribute name="measures" type="xsd:boolean" default="false"/>
    <attribute name="count" use="required" type="xsd:unsignedInt"/>
    <attribute name="oneField" type="xsd:boolean" default="false"/>
    <attribute name="memberValueDatatype" use="optional" type="xsd:unsignedShort"/>
    <attribute name="unbalanced" use="optional" type="xsd:boolean"/>
    <attribute name="unbalancedGroup" use="optional" type="xsd:boolean"/>
    <attribute name="hidden" type="xsd:boolean" default="false"/>
</complexType>
```

3.10.1.7 cacheSource (PivotCache Source Description)

Represents the description of data source whose data is stored in the pivot cache. The data source refers to the underlying rows or database records that provide the data for a PivotTable. You can create a PivotTable report from a SpreadsheetML table, an external database (including OLAP cubes), multiple SpreadsheetML worksheets, or another PivotTable.

Quarter	Region	Sport	Sales
Qtr1	East	Golf	$5,000
Qtr1	East	Safari	$9,000
Qtr1	East	Tennis	$1,500
Qtr2	East	Golf	$2,000
Qtr2	East	Safari	$6,000
Qtr2	East	Tennis	$500
Qtr1	West	Golf	$3,500
Qtr1	West	Tennis	$6,000
Qtr2	West	Golf	$2,500
Qtr2	West	Tennis	$3,200

Information about the data source is stored in the connection element and is retrieved using the connectionId attribute.

[*Example*:

```
<cacheSource type="external" connectionId="1"/>
```

end example]

OLAP data sources are distinguished from other data sources in SpreadsheetML. OLAP records are not stored in the *pivotCacheRecords* part, whereas all records for non-OLAP data sources are stored in the cache.

Parent Elements
pivotCacheDefinition (§3.10.1.67)

Child Elements	Subclause
consolidation (Consolidation Source)	§3.10.1.20
extLst (Future Feature Data Storage Area)	§3.2.10
worksheetSource (Worksheet PivotCache Source)	§3.10.1.95

Attributes	Description
connectionId (Connection Index)	Specifies the index to the workbook connection. This attribute is used when the cache type is 'External.' See §3.13.1 for more information about the connection element. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
type (Cache Type)	Specifies the cache type. The possible values for this attribute are defined by the ST_SourceType simple type (§3.18.77).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CacheSource">
   <choice minOccurs="0" maxOccurs="1">
      <element name="worksheetSource" type="CT_WorksheetSource" minOccurs="1" maxOccurs="1"/>
      <element name="consolidation" type="CT_Consolidation" minOccurs="1" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0"/>
   </choice>
   <attribute name="type" type="ST_SourceType" use="required"/>
   <attribute name="connectionId" type="xsd:unsignedInt" default="0" use="optional"/>
</complexType>
```

3.10.1.8 calculatedItem (Calculated Item)

Represents an item within a PivotTable field that uses a formula . The formula is specified in the formula attribute.

Calculations and options available for a PivotTable depend on whether the source data came from an OLAP database or another type of database. This complex type applies to non-OLAP external data or on worksheet data. See calculatedMember for information on calculations on OLAP data sources.

Parent Elements
calculatedItems (§3.10.1.9)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
pivotArea (Pivot Area)	§3.3.1.66

Attributes	Description
field (Field Index)	Specifies the index of the pivotField with which this calculated item is associated. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
formula (Calculated Item Formula)	Specifies the formula of the calculated item. In formulas you create for calculated items, you can use operators and expressions as you do in other worksheet formulas. You can use constants and refer to data from the PivotTable, but you cannot use cell references or defined names. You cannot use worksheet functions that require cell references or defined names as arguments, and you cannot use array functions. Further behaviors and restrictions apply to formulas for `calculatedItems`: • Formulas for calculated items operate on the individual records; the calculated item formula =Dairy *115% multiplies each individual sale of Dairy times 115%, after which the multiplied amounts are summarized together in the data area. • Formulas cannot refer to totals. • You can include the field name in a reference to an item. The item name must be in square brackets. Use this format to avoid #NAME? errors when two items in two different fields in a report have the same name. • You can refer to an item by its position in the PivotTable as currently sorted and displayed. The item referred to in this way can change whenever the positions of items change or different items are displayed or hidden. Hidden items are not counted in this index. • You can use relative positions to refer to items. The positions are determined relative to the calculated item that contains the formula. If the position you give is before the first item or after the last item in the field, the formula results in a #REF! error. For more information about formulas see §3.17 in Formulas. For more information about defined names see §3.2.6 in Workbook. The possible values for this attribute are defined by the ST_Xstring simple type

Attributes	Description
	(§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CalculatedItem">
    <sequence>
        <element name="pivotArea" type="CT_PivotArea"/>
        <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
    </sequence>
    <attribute name="field" type="xsd:unsignedInt" use="optional"/>
    <attribute name="formula" type="ST_Xstring"/>
</complexType>
```

3.10.1.9 calculatedItems (Calculated Items)

Represents the collection of calculated items.

Parent Elements
pivotCacheDefinition (§3.10.1.67)

Child Elements	Subclause
calculatedItem (Calculated Item)	§3.10.1.8

Attributes	Description
count (Calculated Item Formula Count)	Specifies the number of calculated item formulas in the cache. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CalculatedItems">
    <sequence>
        <element name="calculatedItem" maxOccurs="unbounded" type="CT_CalculatedItem"/>
    </sequence>
    <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.10 calculatedMember (Calculated Member)

Represents a calculated OLAP hierarchy. A calculated member is a member of an OLAP-based PivotTable whose value is calculated on the OLAP server. For PivotTables that are created from OLAP cubes the summarized values are precalculated on the OLAP server before the SpreadsheetML application displays the results. These fields appear in the PivotTable field list but cannot be changed from within the PivotTable. You cannot change the summary function used to calculate data fields or subtotals, or add calculated items.

Calculated members are defined by the Multidimensional Expressions (MDX) expression in the mdx attribute.

[*Example*:

```
<calculatedMembers count="1">
  <calculatedMember name="[Product].[Product Categories].[All
    Products].[Calculated Member]" mdx="'[Product].[Product Categories].[All
    Products].[Accessories]'" memberName="Calculated Member"
    hierarchy="[Product].[Product Categories]" parent="[Product].[Product
    Categories].[All Products]"/>
</calculatedMembers>
```

end example]

Parent Elements
calculatedMembers (§3.10.1.11)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10

Attributes	Description
hierarchy (Hierarchy Name)	Specifies the name of the hierarchy to which the calculated member belongs. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
mdx (Calculated Member MDX Formula)	Specifies the MDX formula for the calculated member. [*Note*: Data connectivity can use a number of different technologies. One example of potential values stored in this attribute can be found at: http://msdn2.microsoft.com/en-us/library/ms145595.aspx *end note*] The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
memberName (OLAP Calculated Member Name)	Specifies the OLAP member name for the calculated member. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
name (Calculated Member Name)	Specifies the name of the calculated member. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
parent (Parent	Specifies the name of the parent of the calculated member.

Attributes	Description
Name)	The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
set (Set)	Specifies a boolean value that indicates whether this calculated member describes a calculated set rather than a calculated member. A value of on, 1, or true indicates this is a calculated set. A value of off, 0, or false indicates this is a calculated member. The possible values for this attribute are defined by the XML Schema boolean datatype.
solveOrder (Calculated Members Solve Order)	Specifies the order in which this calculated member is calculated in relation to other calculated members. The possible values for this attribute are defined by the XML Schema int datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CalculatedMember">
   <sequence minOccurs="0">
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="name" use="required" type="ST_Xstring"/>
   <attribute name="mdx" use="required" type="ST_Xstring"/>
   <attribute name="memberName" type="ST_Xstring"/>
   <attribute name="hierarchy" type="ST_Xstring"/>
   <attribute name="parent" type="ST_Xstring"/>
   <attribute name="solveOrder" type="xsd:int" default="0"/>
   <attribute name="set" type="xsd:boolean" default="false"/>
</complexType>
```

3.10.1.11 calculatedMembers (Calculated Members)

Represents the collection of calculated members in an OLAP PivotTable.

[*Example*:

```
<calculatedMembers count="1">
  <calculatedMember name="[Product].[Product Categories].[All
    Products].[Calculated Member]" mdx="'[Product].[Product Categories].[All
    Products].[Accessories]'" memberName="Calculated Member"
    hierarchy="[Product].[Product Categories]" parent="[Product].[Product
    Categories].[All Products]"/>
</calculatedMembers>
```

end example]

Parent Elements
pivotCacheDefinition (§3.10.1.67)

Child Elements	Subclause
calculatedMember (Calculated Member)	§3.10.1.10

Attributes	Description
count (Calculated Members Count)	Specifies the number of calculated members. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CalculatedMembers">
   <sequence>
      <element name="calculatedMember" maxOccurs="unbounded" type="CT_CalculatedMember"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.12 chartFormat (PivotChart Format)

Represents the format defined in the PivotChart that is associated with this PivotTable.

[*Example*:

```
<sh:pivotTableDefinition xmlns:sh="…" name="PivotTable1" cacheId="0"
  applyNumberFormats="0" applyBorderFormats="0" applyFontFormats="0"
  applyPatternFormats="0" applyAlignmentFormats="0" applyWidthHeightFormats="1"
  dataCaption="Values" updatedVersion="3" minRefreshableVersion="3"
  showCalcMbrs="0" useAutoFormatting="1" colGrandTotals="0" itemPrintTitles="1"
  createdVersion="3" indent="0" outline="1" outlineData="1"
  multipleFieldFilters="0" chartFormat="1" fieldListSortAscending="1">
```

end example]

Parent Elements
chartFormats (§3.10.1.13)

Child Elements	Subclause
pivotArea (Pivot Area)	§3.3.1.66

Attributes	Description
chart (Chart Index)	Specifies the index of the chart part to which the formatting applies. For more information see the DrawingML specification for more information on the chart part. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
format (Pivot Format Id)	Specifies the index of the pivot format that is currently in use. This index corresponds to a dxf element in the Styles part. For more information see the Styles section (§3.8). The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
series (Series Format)	Specifies a boolean value that indicates whether format applies to a series. A value of on, 1, or true indicates this format applies to a series. A value of off, 0, or false indicates this format applies to a data point. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ChartFormat">
   <sequence>
      <element name="pivotArea" type="CT_PivotArea"/>
   </sequence>
   <attribute name="chart" use="required" type="xsd:unsignedInt"/>
   <attribute name="format" use="required" type="xsd:unsignedInt"/>
   <attribute name="series" type="xsd:boolean" default="false"/>
</complexType>
```

3.10.1.13 chartFormats (PivotChart Formats)

Represents the collection of formats applied to PivotChart.

[*Example*:

```
<sh:chartFormats count="4">
  <sh:chartFormat chart="0" format="0" series="1">
    <sh:pivotArea type="data" outline="0">
      <sh:references count="3">
        <sh:reference field="4294967294" count="1" selected="0">
          <sh:x v="0"/>
        </sh:reference>
        <sh:reference field="14" count="1" selected="0">
          <sh:x v="0"/>
        </sh:reference>
```

```
      <sh:reference field="15" count="1" selected="0">
        <sh:x v="2"/>
      </sh:reference>
    </sh:references>
  </sh:pivotArea>
</sh:chartFormat>
<sh:chartFormat chart="0" format="1" series="1">
  <sh:pivotArea type="data" outline="0">
    <sh:references count="3">
      <sh:reference field="4294967294" count="1" selected="0">
        <sh:x v="0"/>
      </sh:reference>
      <sh:reference field="14" count="1" selected="0">
        <sh:x v="0"/>
      </sh:reference>
      <sh:reference field="15" count="1" selected="0">
        <sh:x v="3"/>
      </sh:reference>
    </sh:references>
  </sh:pivotArea>
</sh:chartFormat>
<sh:chartFormat chart="0" format="2" series="1">
  <sh:pivotArea type="data" outline="0">
    <sh:references count="3">
      <sh:reference field="4294967294" count="1" selected="0">
        <sh:x v="1"/>
      </sh:reference>
      <sh:reference field="14" count="1" selected="0">
        <sh:x v="0"/>
      </sh:reference>
      <sh:reference field="15" count="1" selected="0">
        <sh:x v="2"/>
      </sh:reference>
    </sh:references>
  </sh:pivotArea>
</sh:chartFormat>
<sh:chartFormat chart="0" format="3" series="1">
  <sh:pivotArea type="data" outline="0">
    <sh:references count="3">
      <sh:reference field="4294967294" count="1" selected="0">
        <sh:x v="1"/>
      </sh:reference>
```

```
          <sh:reference field="14" count="1" selected="0">
            <sh:x v="0"/>
          </sh:reference>
          <sh:reference field="15" count="1" selected="0">
            <sh:x v="3"/>
          </sh:reference>
        </sh:references>
      </sh:pivotArea>
    </sh:chartFormat>
  </sh:chartFormats>
```

end example]

Parent Elements
pivotTableDefinition (§3.10.1.73)

Child Elements	Subclause
chartFormat (PivotChart Format)	§3.10.1.12

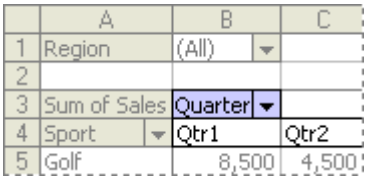
Attributes	Description
count (Format Count)	Specifies the number of formats in the collection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ChartFormats">
   <sequence>
      <element name="chartFormat" maxOccurs="unbounded" type="CT_ChartFormat"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" default="0"/>
</complexType>
```

3.10.1.14 colFields (Column Fields)

Represents the collection of fields that are on the column axis of the PivotTable.



	A	B	C
1	Region	(All)	
2			
3	Sum of Sales	Quarter	
4	Sport	Qtr1	Qtr2
5	Golf	8,500	4,500

In the image above, the blue field is a column field.

In the following SpreadsheetML example, "Year", "Quarter" and "Month" are on the column axis of the PivotTable, in that order.

[*Example*:

```
<colFields count="3">
  <field x="14"/>
  <field x="15"/>
  <field x="16"/>
</colFields>
```

end example]

Parent Elements
pivotTableDefinition (§3.10.1.73)

Child Elements	Subclause
field (Field)	§3.10.1.29

Attributes	Description
count (Repeated Items Count)	Specifies the number of items in this collection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ColFields">
   <sequence>
      <element name="field" maxOccurs="unbounded" type="CT_Field"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" default="0"/>
</complexType>
```

3.10.1.15 colHierarchiesUsage (Column OLAP Hierarchy References)

Represents the collection of references to OLAP hierarchies on the column axis of a PivotTable.

[*Example*:

```
<sh:colHierarchiesUsage count="2">
  <sh:colHierarchyUsage hierarchyUsage="33"/>
  <sh:colHierarchyUsage hierarchyUsage="-2"/>
</sh:colHierarchiesUsage>
```

end example]

Parent Elements
pivotTableDefinition (§3.10.1.73)

Child Elements	Subclause
colHierarchyUsage (Column OLAP Hierarchies)	§3.10.1.16

Attributes	Description
count (Items Count)	Specifies the number of items in the collection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ColHierarchiesUsage">
   <sequence>
      <element name="colHierarchyUsage" minOccurs="1" maxOccurs="unbounded"
        type="CT_HierarchyUsage"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.16 colHierarchyUsage (Column OLAP Hierarchies)

Represents the collection of references to OLAP Hierarchies on the column axis of a PivotTable.

[*Example*:

<sh:colHierarchyUsage hierarchyUsage="33"/>

end example]

Parent Elements
colHierarchiesUsage (§3.10.1.15)

Attributes	Description
hierarchyUsage (Hierarchy Usage)	Specifies the reference to an OLAP hierarchy in a PivotTable. The possible values for this attribute are defined by the XML Schema int datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HierarchyUsage">
   <attribute name="hierarchyUsage" type="xsd:int" use="required"/>
</complexType>
```

3.10.1.17 colItems (Column Items)

Represents the collection of column items of the PivotTable.

In the following SpreadsheetML example the item values are found in cells C6:H8. For example "2001" / "3" / "July" values are in C7:C9. Those are the first column item values and are referenced by the first <i> element below.

[*Example*:

```
<colItems count="5">
  <i>
    <x/>
    <x/>
    <x/>
  </i>
  <i r="2">
    <x v="1"/>
  </i>
  <i r="2">
    <x v="2"/>
  </i>
  <i t="default" r="1">
    <x/>
  </i>
  <i t="default">
    <x/>
  </i>
</colItems>
```

end example]

The first <i> collection represents all item values for the first column in the column axis area of the PivotTable. The first <x> in the first <i> corresponds to the first field in the columns area of the PivotTable, namely "Year". The implied index value of '0' on this <x> indicates that the item value for this first item in the column is the 0th item for this pivotField. The 0th item for this pivotField is itself an index to an item value into this field's shared items collection in the pivotCacheDefinition part, namely "2001".

The item values corresponding to the second and third <x> elements can be found in the same way, arriving at "3" for the second item value, and arriving at "July" for the third item value for this first column.

The second <i> collection expresses all 3 item values for the second column in the column axis area. The @r value of '2' indicates that the first two item values from the previous column will be repeated here, which means that the first item value for this second column will be "2001" again and the second item value for this second column will be "3". The third item value is expressed by the only <x> element under this second <i> element, and without further explanation is understood to reference the item value "August".

Parent Elements
pivotTableDefinition (§3.10.1.73)

Child Elements	Subclause
i (Row Items)	§3.10.1.44

Attributes	Description
count (Column Item Count)	Specifies the number of items on the column axis of the PivotTable. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_colItems">
   <sequence>
      <element name="i" maxOccurs="unbounded" type="CT_I"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.18 conditionalFormat (Conditional Formatting)

Represents the conditional formatting defined in the PivotTable.

Parent Elements
conditionalFormats (§3.10.1.19)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
pivotAreas (Pivot Areas)	§3.10.1.66

Attributes	Description
priority (Priority)	Specifies the priority of PivotTable conditional formatting rule. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
scope (Conditional Formatting Scope)	Specifies the scope of PivotTable conditional formatting rule.

Attributes	Description
	The possible values for this attribute are defined by the ST_Scope simple type (§3.18.69).
type (Conditional Formatting Rule Type)	Specifies the type of PivotTable conditional formatting rule. See associated simple type definition for details. The possible values for this attribute are defined by the ST_Type simple type (§3.18.84).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ConditionalFormat">
   <sequence>
      <element name="pivotAreas" type="CT_PivotAreas"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="scope" type="ST_Scope" default="selection"/>
   <attribute name="type" type="ST_Type" default="none"/>
   <attribute name="priority" use="required" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.19 conditionalFormats (Conditional Formats)

Represents the collection of conditional formats applied to a PivotTable.

[*Example*:

```
<sh:conditionalFormats count="1">
  <sh:conditionalFormat priority="1">
    <sh:pivotAreas count="1">
      <sh:pivotArea type="data" collapsedLevelsAreSubtotals="1">
        <sh:references count="5">
          <sh:reference field="4294967294" count="1" selected="0">
            <sh:x v="0"/>
          </sh:reference>
          <sh:reference field="2" count="1" selected="0">
            <sh:x v="0"/>
          </sh:reference>
          <sh:reference field="14" count="1" selected="0">
            <sh:x v="0"/>
          </sh:reference>
```

```
            <sh:reference field="15" count="2" selected="0">
              <sh:x v="2"/>
              <sh:x v="3"/>
            </sh:reference>
          </sh:references>
        </sh:pivotArea>
      </sh:pivotAreas>
    </sh:conditionalFormat>
  </sh:conditionalFormats>
```

end example]

Parent Elements
pivotTableDefinition (§3.10.1.73)

Child Elements	Subclause
conditionalFormat (Conditional Formatting)	§3.10.1.18

Attributes	Description
count (Conditional Format Count)	Specifies the number of conditional formats defined for the PivotTable. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ConditionalFormats">
   <sequence>
      <element name="conditionalFormat" maxOccurs="unbounded" type="CT_ConditionalFormat"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" default="0"/>
</complexType>
```

3.10.1.20 consolidation (Consolidation Source)

Represents the description of the PivotCache source using multiple consolidation ranges. This element is used when the source of the PivotTable is a collection of ranges in the workbook. The ranges are specified in the rangeSets collection. The logic for how the application consolidates the data in the ranges is application-specific. For example, the application may consolidate data based on its position in the worksheet that the end-user specifies.

Parent Elements
cacheSource (§3.10.1.7)

Child Elements	Subclause
pages (Page Item Values)	§3.10.1.65
rangeSets (Range Sets)	§3.10.1.80

Attributes	Description
autoPage (Auto Page)	Specifies a boolean value that indicates whether the application will automatically create one additional page field to describe/qualify the source ranges. A value of on, 1, or true indicates the application will create an additional page field. A value of off, 0, or false indicates will not create an additional page field. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Consolidation">
   <sequence>
      <element name="pages" type="CT_Pages" minOccurs="0" maxOccurs="1"/>
      <element name="rangeSets" type="CT_RangeSets" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="autoPage" type="xsd:boolean" default="true" use="optional"/>
</complexType>
```

3.10.1.21 d (Date Time)

Represents a date-time value in the PivotTable.

Parent Elements
groupItems (§3.10.1.38); r (§3.10.1.77); sharedItems (§3.10.1.90)

Child Elements	Subclause
x (Member Property Index)	§3.10.1.96

Attributes	Description
c (Caption)	Specifies the caption for the item. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
cp (Member Property Count)	Specifies the number of member property values. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

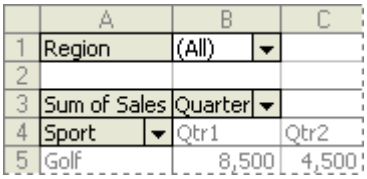
Attributes	Description
f (Calculated Item Value)	Specifies a boolean value that indicates whether this is a calculated item value. A value of on, 1, or true indicates this is a calculated item value. A value of off, 0, or false indicates this is not a calculated item value. The possible values for this attribute are defined by the XML Schema boolean datatype.
u (Unused Item)	Specifies a boolean value that indicates whether this is an unused item. The application marks an item as unused when an item is deleted from the data source. The item and associated metadata are retained in the cache until the threshold for unused items specified in missingItemsLimit is reached. A value of on, 1, or true indicates this is an unused item. A value of off, 0, or false indicates this item is used. The possible values for this attribute are defined by the XML Schema boolean datatype.
v (Value)	Specifies the value of the item. The possible values for this attribute are defined by the XML Schema dateTime datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DateTime">
   <sequence>
      <element name="x" minOccurs="0" maxOccurs="unbounded" type="CT_X"/>
   </sequence>
   <attribute name="v" use="required" type="xsd:dateTime"/>
   <attribute name="u" type="xsd:boolean"/>
   <attribute name="f" type="xsd:boolean"/>
   <attribute name="c" type="ST_Xstring"/>
   <attribute name="cp" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.22 dataField (Data Field Item)

Represents a field from a source list, table, or database that contains data that is summarized in a PivotTable.



A data field represents data that's derived from a field in the source list or database. The Sport field, for example, might come from a column in the source list that's labeled Sport and contains the names of various sports (Golf, Tennis) for which the source list has sales figures. Source data can be taken from an SpreadsheetML list or range, an external database or cube, or another PivotTable. Data fields use summary functions to

combine values from the underlying source data. You can also use custom calculations to compare data values, or add your own formulas that use elements of the report or other worksheet data.

[*Example*:

```
<dataFields count="1">
  <dataField name="Sum of Sales Amount" fld="25" baseField="0" baseItem="0"/>
</dataFields>
```

end example]

Parent Elements
dataFields (§3.10.1.23)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10

Attributes	Description
baseField ('Show Data As' Base Field)	Specifies the index to the base field when the ShowDataAs calculation is in use. The possible values for this attribute are defined by the XML Schema int datatype.
baseItem ('Show Data As' Base Setting)	Specifies the index to the base item when the ShowDataAs calculation is in use. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
fld (Field)	Specifies the index to the field (<r>) in the pivotCacheRecords part that this data item summarizes. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
name (Data Field Name)	Specifies the name of the data field. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
numFmtId (Number Format Id)	Specifies the index to the number format applied to this data field. Number formats are written to the styles part. See the Styles section(§3.8) for more information on number formats. Note: Formatting information provided by cell table and by PivotTable need not agree. If the two formats differ, the cell-level formatting takes precedence. If you change the layout the PivotTable, the PivotTable formatting will then take precedence.

Attributes	Description
	The possible values for this attribute are defined by the ST_NumFmtId simple type (§3.18.49).
showDataAs (Show Data As Display Format)	Specifies the display format for this data field. Note: Formatting information provided by cell table and by PivotTable need not agree. If the two formats differ, the cell-level formatting takes precedence. If you change the layout the PivotTable, the PivotTable formatting will then take precedence. The possible values for this attribute are defined by the ST_ShowDataAs simple type (§3.18.72).
subtotal (Subtotal)	Specifies the aggregation function that applies to this data field. The possible values for this attribute are defined by the ST_DataConsolidateFunction simple type (§3.18.18).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DataField">
   <sequence>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="name" use="optional" type="ST_Xstring"/>
   <attribute name="fld" type="xsd:unsignedInt" use="required"/>
   <attribute name="subtotal" type="ST_DataConsolidateFunction" default="sum"/>
   <attribute name="showDataAs" type="ST_ShowDataAs" default="normal"/>
   <attribute name="baseField" type="xsd:int" default="-1"/>
   <attribute name="baseItem" type="xsd:unsignedInt" default="1048832"/>
   <attribute name="numFmtId" type="ST_NumFmtId" use="optional"/>
</complexType>
```

3.10.1.23 dataFields (Data Fields)

Represents the collection of items in the data region of the PivotTable.

[*Example*:

```
<dataFields count="1">
  <dataField name="Sum of Sales Amount" fld="25" baseField="0" baseItem="0"/>
</dataFields>
```

end example]

Parent Elements
pivotTableDefinition (§3.10.1.73)

Child Elements	Subclause

Child Elements	Subclause
dataField (Data Field Item)	§3.10.1.22

Attributes	Description
count (Data Items Count)	Specifies the number of items in the data region of the PivotTable. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DataFields">
   <sequence>
      <element name="dataField" maxOccurs="unbounded" type="CT_DataField"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.24 dimension (OLAP Dimension)

Represents a PivotTable OLAP Dimension. A dimension is a field that organizes a single type of data into a hierarchy with levels of detail. For example, an OLAP database could contain a Time dimension providing data for levels Year, Month, Week, and Day, allowing you to create reports that let you compare day-to-day sales results or view a summary of your sales for an entire year.

Parent Elements
dimensions (§3.10.1.25)

Attributes	Description
caption (Dimension Display Name)	Specifies the display name of the dimension. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
measure (Measure)	Specifies a boolean value that indicates whether this is a measure dimension. A value of on, 1, or true indicates this dimension is a measure dimension. A value of off, 0, or false indicates this dimension is not a measure dimension. The possible values for this attribute are defined by the XML Schema boolean datatype.
name (Dimension Name)	Specifies the name of the dimension. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

Attributes	Description
uniqueName (Dimension Unique Name)	Specifies the unique name of the dimension. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PivotDimension">
   <attribute name="measure" type="xsd:boolean" default="false"/>
   <attribute name="name" use="required" type="ST_Xstring"/>
   <attribute name="uniqueName" use="required" type="ST_Xstring"/>
   <attribute name="caption" use="required" type="ST_Xstring"/>
</complexType>
```

3.10.1.25 dimensions (OLAP Dimensions)

Represents the collection of PivotTable OLAP dimensions.

[*Example*:

```
<dimensions count="22">
  <dimension name="Account" uniqueName="[Account]" caption="Account"/>
  <dimension name="Customer" uniqueName="[Customer]" caption="Customer"/>
  <dimension name="Date" uniqueName="[Date]" caption="Date"/>
  <dimension name="Delivery Date" uniqueName="[Delivery Date]"
    caption="Delivery Date"/>
  <dimension name="Department" uniqueName="[Department]" caption="Department"/>
  <dimension name="Destination Currency" uniqueName="[Destination Currency]"
    caption="Destination Currency"/>
  <dimension name="Employee" uniqueName="[Employee]" caption="Employee"/>
  <dimension name="Geography" uniqueName="[Geography]" caption="Geography"/>
  <dimension name="Internet Sales Order Details" uniqueName="[Internet Sales
    Order Details]" caption="Internet Sales Order Details"/>
  <dimension measure="1" name="Measures" uniqueName="[Measures]"
    caption="Measures"/>
  <dimension name="Organization" uniqueName="[Organization]"
    caption="Organization"/>
  <dimension name="Product" uniqueName="[Product]" caption="Product"/>
  <dimension name="Promotion" uniqueName="[Promotion]" caption="Promotion"/>
  <dimension name="Reseller" uniqueName="[Reseller]" caption="Reseller"/>
  <dimension name="Reseller Sales Order Details" uniqueName="[Reseller Sales
    Order Details]" caption="Reseller Sales Order Details"/>
  <dimension name="Sales Channel" uniqueName="[Sales Channel]" caption="Sales
    Channel"/>
  <dimension name="Sales Reason" uniqueName="[Sales Reason]" caption="Sales
    Reason"/>
```

```
    <dimension name="Sales Summary Order Details" uniqueName="[Sales Summary Order
      Details]" caption="Sales Summary Order Details"/>
    <dimension name="Sales Territory" uniqueName="[Sales Territory]"
      caption="Sales Territory"/>
    <dimension name="Scenario" uniqueName="[Scenario]" caption="Scenario"/>
    <dimension name="Ship Date" uniqueName="[Ship Date]" caption="Ship Date"/>
    <dimension name="Source Currency" uniqueName="[Source Currency]"
      caption="Source Currency"/>
  </dimensions>
```

end example]

Parent Elements
pivotCacheDefinition (§3.10.1.67)

Child Elements	Subclause
dimension (OLAP Dimension)	§3.10.1.24

Attributes	Description
count (OLAP Dimensions Count)	Specifies the number of OLAP dimensions in the PivotTable. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Dimensions">
   <sequence>
      <element name="dimension" minOccurs="0" maxOccurs="unbounded" type="CT_PivotDimension"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.26 discretePr (Discrete Grouping Properties)

Represents the collection of discrete grouping properties for a field group.

[*Example*:

```
  …
    <fieldGroup par="6" base="0">
      <rangePr groupBy="months" startDate="2002-01-01T00:00:00"
        endDate="2006-05-06T00:00:00"/>
```

```
    <groupItems count="14">
      <s v="&lt;1/1/2002"/>
      <s v="Jan"/>
      <s v="Feb"/>
      <s v="Mar"/>
      <s v="Apr"/>
      <s v="May"/>
      <s v="Jun"/>
      <s v="Jul"/>
      <s v="Aug"/>
      <s v="Sep"/>
      <s v="Oct"/>
      <s v="Nov"/>
      <s v="Dec"/>
      <s v="&gt;5/6/2006"/>
    </groupItems>
  </fieldGroup>
</cacheField>
<cacheField name="Name" numFmtId="0">
  <sharedItems count="4">
    <s v="Joe"/>
    <s v="John"/>
    <s v="Bob"/>
    <s v="Robert"/>
  </sharedItems>
  <fieldGroup par="4"/>
</cacheField>
<cacheField name="ProductID" numFmtId="0">
  <sharedItems containsSemiMixedTypes="0" containsString="0" containsNumber="1"
    containsInteger="1" minValue="1" maxValue="4" count="4">
    <n v="1"/>
    <n v="2"/>
    <n v="3"/>
    <n v="4"/>
  </sharedItems>
```

```
    <fieldGroup base="2">
      <rangePr startNum="1" endNum="4" groupInterval="2"/>
      <groupItems count="4">
        <s v="&lt;1"/>
        <s v="1-2"/>
        <s v="3-4"/>
        <s v="&gt;5"/>
      </groupItems>
    </fieldGroup>
  </cacheField>
```

end example]

Parent Elements
fieldGroup (§3.10.1.30)

Child Elements	Subclause
x (Shared Items Index)	§3.10.1.97

Attributes	Description
count (Mapping Index Count)	Specifies the number of mapping indexes for this grouped field. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DiscretePr">
   <sequence>
      <element name="x" maxOccurs="unbounded" type="CT_Index"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.27 e (Error Value)

Represents an error value. The use of this item indicates that an error value is present in the PivotTable source. The error is recorded in the `value` attribute.

Parent Elements
entries (§3.10.1.28); groupItems (§3.10.1.38); r (§3.10.1.77); sharedItems (§3.10.1.90)

Child Elements	Subclause

Child Elements	Subclause
tpls (Tuples)	§3.10.1.93
x (Member Property Index)	§3.10.1.96

Attributes	Description
b (Bold)	Specifies a boolean value that indicates whether the value contains bold formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of on, 1, or true indicates this value contains bold formatting on the server. The possible values for this attribute are defined by the XML Schema boolean datatype.
bc (background Color)	Specifies the background color for this value that was provided by the OLAP server. This attribute applies to OLAP-based PivotTables only. The color is specified as a HEX value in RGB space. The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§3.18.86).
c (Item Caption)	Specifies the item/member caption The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
cp (Member Property Count)	Specifies the number of member property values. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
f (Calculated Item)	Specifies a boolean value that indicates whether this is a calculated item value. A value of on, 1, or true indicates value is a calculated item value. A value of off, 0, or false indicates this value is not a calculated item value. The possible values for this attribute are defined by the XML Schema boolean datatype.
fc (Foreground Color)	Specifies the foreground color for this value that was provided by the OLAP server. This attribute applies to OLAP-based PivotTables only. The color is specified as a HEX value in RGB space. The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§3.18.86).
i (Italic)	Specifies a boolean value that indicates whether the value contains italic formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of on, 1, or true indicates this value contains italic formatting on the server.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema boolean datatype.
in (Format Index)	Specifies the index to the OLAP serverformat element where the format string for this entry is stored. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
st (Strikethrough)	Specifies a boolean value that indicates whether the value contains strikethrough formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of on, 1, or true indicates this value contains strikethrough formatting on the server. The possible values for this attribute are defined by the XML Schema boolean datatype.
u (Unused Item)	Specifies a boolean value that indicates whether this is an unused item. The application marks an item as unused when an item is deleted from the data source. The item and associated metadata are retained in the cache until the threshold for unused items specified in missingItemsLimit is reached. A value of on, 1, or true indicates this item is not used. A value of off, 0, or false indicates this item is used. The possible values for this attribute are defined by the XML Schema boolean datatype.
un (Underline)	Specifies a boolean value that indicates whether the value contains underline formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of on, 1, or true indicates this value contains underline formatting on the server. The possible values for this attribute are defined by the XML Schema boolean datatype.
v (Value)	Specifies the value of the item. This attribute depends on how the application records errors. [*Note*: While the error values are determined by the application, the following are some example error values that could be used: • #DIV/0! • #NAME? • #VALUE! • #NULL! • #NUM! • #REF! • #N/A • #GETTING_DATA *end note*] The possible values for this attribute are defined by the ST_Xstring simple type

Attributes	Description
	(§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Error">
   <sequence>
      <element name="tpls" minOccurs="0" type="CT_Tuples"/>
      <element name="x" minOccurs="0" maxOccurs="unbounded" type="CT_X"/>
   </sequence>
   <attribute name="v" use="required" type="ST_Xstring"/>
   <attribute name="u" type="xsd:boolean"/>
   <attribute name="f" type="xsd:boolean"/>
   <attribute name="c" type="ST_Xstring"/>
   <attribute name="cp" type="xsd:unsignedInt"/>
   <attribute name="in" type="xsd:unsignedInt" use="optional"/>
   <attribute name="bc" type="ST_UnsignedIntHex" use="optional"/>
   <attribute name="fc" type="ST_UnsignedIntHex" use="optional"/>
   <attribute name="i" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="un" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="st" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="b" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.10.1.28 entries (Entries)

Represents the collection of OLAP sheet data entries.

Parent Elements
tupleCache (§3.10.1.94)

Child Elements	Subclause
e (Error Value)	§3.10.1.27
m (No Value)	§3.10.1.50
n (Numeric)	§3.10.1.60
s (Character Value)	§3.10.1.85

Attributes	Description
count (Tuple Count)	Specifies the number of tuple entries. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PCDSDTCEntries">
   <choice maxOccurs="unbounded">
      <element name="m" type="CT_Missing"/>
      <element name="n" type="CT_Number"/>
      <element name="e" type="CT_Error"/>
      <element name="s" type="CT_String"/>
   </choice>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.29 field (Field)

Represents a generic field that can appear either on the column or the row region of the PivotTable. There will be as many <x> elements as there are item values in any particular column or row.

[*Example*:

```
<sh:field x="2"/>
```

end example]

Parent Elements
colFields (§3.10.1.14); rowFields (§3.10.1.81)

Attributes	Description
x (Field Index)	Specifies the index to a pivotField item value. There will be as many x elements as there are item values in any particular column. Note that these x elements sometimes are not explicitly written, but instead "inherited" from the previous column or i element, via the value of @r. The pivotField items don't list values explicitly, but instead reference a shared item value in the pivotCacheDefinition part. The first instance of x has no attribute value @v associated with it, so the default value for @v is assumed to be "0". The possible values for this attribute are defined by the XML Schema int datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Field">
   <attribute name="x" type="xsd:int" use="required"/>
</complexType>
```

3.10.1.30 fieldGroup (Field Group Properties)

Represents the collection of properties for a field group.

[*Example*:

```
…
  <fieldGroup par="6" base="0">
    <rangePr groupBy="months" startDate="2002-01-01T00:00:00"
      endDate="2006-05-06T00:00:00"/>
    <groupItems count="14">
      <s v="&lt;1/1/2002"/>
      <s v="Jan"/>
      <s v="Feb"/>
      <s v="Mar"/>
      <s v="Apr"/>
      <s v="May"/>
      <s v="Jun"/>
      <s v="Jul"/>
      <s v="Aug"/>
      <s v="Sep"/>
      <s v="Oct"/>
      <s v="Nov"/>
      <s v="Dec"/>
      <s v="&gt;5/6/2006"/>
    </groupItems>
  </fieldGroup>
</cacheField>
<cacheField name="Name" numFmtId="0">
  <sharedItems count="4">
    <s v="Joe"/>
    <s v="John"/>
    <s v="Bob"/>
    <s v="Robert"/>
  </sharedItems>
  <fieldGroup par="4"/>
</cacheField>
<cacheField name="ProductID" numFmtId="0">
  <sharedItems containsSemiMixedTypes="0" containsString="0" containsNumber="1"
    containsInteger="1" minValue="1" maxValue="4" count="4">
    <n v="1"/>
    <n v="2"/>
    <n v="3"/>
    <n v="4"/>
  </sharedItems>
```

```
    <fieldGroup base="2">
      <rangePr startNum="1" endNum="4" groupInterval="2"/>
      <groupItems count="4">
        <s v="&lt;1"/>
        <s v="1-2"/>
        <s v="3-4"/>
        <s v="&gt;5"/>
      </groupItems>
    </fieldGroup>
  …
```

end example]

Parent Elements
cacheField (§3.10.1.3)

Child Elements	Subclause
discretePr (Discrete Grouping Properties)	§3.10.1.26
groupItems (OLAP Group Items)	§3.10.1.38
rangePr (Range Grouping Properties)	§3.10.1.78

Attributes	Description
base (Field Base)	Specifies the base of this field, if any. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
par (Parent)	Specifies the parent of this field, if any. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FieldGroup">
   <sequence>
      <element name="rangePr" minOccurs="0" type="CT_RangePr"/>
      <element name="discretePr" minOccurs="0" type="CT_DiscretePr"/>
      <element name="groupItems" minOccurs="0" type="CT_GroupItems"/>
   </sequence>
   <attribute name="par" type="xsd:unsignedInt" use="optional"/>
   <attribute name="base" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.10.1.31 fieldsUsage (Fields Usage)

Represents the fields in the cache that are being used by this hierarchy.

[*Example*:

```
<fieldsUsage count="6">
  <fieldUsage x="-1"/>
  <fieldUsage x="2"/>
  <fieldUsage x="3"/>
  <fieldUsage x="4"/>
  <fieldUsage x="5"/>
  <fieldUsage x="6"/>
</fieldsUsage>
```

end example]

Parent Elements
cacheHierarchy (§3.10.1.6)

Child Elements	Subclause
fieldUsage (PivotCache Field Id)	§3.10.1.32

Attributes	Description
count (Field Count)	Specifies the number of fields that are being used by this hierarchy. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FieldsUsage">
   <sequence>
      <element name="fieldUsage" minOccurs="0" maxOccurs="unbounded" type="CT_FieldUsage"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.32 fieldUsage (PivotCache Field Id)

Represents a cache field used in this hierarchy.

[*Example*:

```
<fieldUsage x="-1"/>
```

end example]

Parent Elements
fieldsUsage (§3.10.1.31)

Attributes	Description
x (Field Index)	Specifies the index of a field. The possible values for this attribute are defined by the XML Schema int datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FieldUsage">
   <attribute name="x" use="required" type="xsd:int"/>
</complexType>
```

3.10.1.33 filter (PivotTable Advanced Filter)

Represents a PivotTable advanced filter.

[*Example*:

```
<sh:filter fld="3" type="count" id="1" iMeasureHier="187">
  <sh:autoFilter ref="A1">
    <sh:filterColumn colId="0">
      <sh:top10 val="5"/>
    </sh:filterColumn>
  </sh:autoFilter>
</sh:filter>
```

end example]

Parent Elements
filters (§3.10.1.34)

Child Elements	Subclause
autoFilter (AutoFilter Settings)	§3.3.1.1
extLst (Future Feature Data Storage Area)	§3.2.10

Attributes	Description
description (Pivot Filter Description)	Specifies the description of the pivot filter. The possible values for this attribute are defined by the ST_Xstring simple type

Attributes	Description
	(§3.18.96).
evalOrder (Evaluation Order)	Specifies the evaluation order of the pivot filter. This attribute is zero-based. The possible values for this attribute are defined by the XML Schema int datatype.
fld (Field Index)	Specifies the index of the field to which this pivot filter belongs. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
id (Pivot Filter Id)	Specifies the unique identifier of the pivot filter as assigned by the PivotTable. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
iMeasureFld (Measure Field Index)	Specifies the index of the measure field. This attribute is used only by filters in Relational pivots and specifies on which measure a value filter should apply. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
iMeasureHier (Measure Index)	Specifies the index of the measure cube field. This attribute is used only by filters in OLAP pivots and specifies on which measure a value filter should apply. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
mpFld (Member Property Field Id)	Specifies the index of the field representing the member property field on which this pivot filter is defined. This attribute is used only by label pivot filters. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
name (Pivot Filter Name)	Specifies the name of the pivot filter. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
stringValue1 (Label Pivot)	Specifies the string value "1" used by label pivot filters. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
stringValue2 (Label Pivot Filter String Value 2)	Specifies the string value "2" used by label pivot filters. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
type (Pivot Filter Type)	Specifies the type of the pivot filter. The possible values for this attribute are defined by the ST_PivotFilterType simple type

Attributes	Description
	(§3.18.61).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PivotFilter">
   <sequence>
      <element name="autoFilter" minOccurs="1" maxOccurs="1" type="CT_AutoFilter"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="fld" use="required" type="xsd:unsignedInt"/>
   <attribute name="mpFld" type="xsd:unsignedInt" use="optional"/>
   <attribute name="type" use="required" type="ST_PivotFilterType"/>
   <attribute name="evalOrder" use="optional" type="xsd:int" default="0"/>
   <attribute name="id" use="required" type="xsd:unsignedInt"/>
   <attribute name="iMeasureHier" use="optional" type="xsd:unsignedInt"/>
   <attribute name="iMeasureFld" use="optional" type="xsd:unsignedInt"/>
   <attribute name="name" type="ST_Xstring"/>
   <attribute name="description" type="ST_Xstring"/>
   <attribute name="stringValue1" type="ST_Xstring"/>
   <attribute name="stringValue2" type="ST_Xstring"/>
</complexType>
```

3.10.1.34 filters (Filters)

Represents the collection of filters that apply to this PivotTable.

[*Example*:

```
<sh:filters count="1">
  <sh:filter fld="3" type="count" id="1" iMeasureHier="187">
    <sh:autoFilter ref="A1">
      <sh:filterColumn colId="0">
        <sh:top10 val="5"/>
      </sh:filterColumn>
    </sh:autoFilter>
  </sh:filter>
</sh:filters>
```

end example]

Parent Elements
pivotTableDefinition (§3.10.1.73)

Child Elements	Subclause
filter (PivotTable Advanced Filter)	§3.10.1.33

Attributes	Description
count (Pivot Filter Count)	Specifies the number of pivot filters in the collection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PivotFilters">
   <sequence>
      <element name="filter" minOccurs="0" maxOccurs="unbounded" type="CT_PivotFilter"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" default="0"/>
</complexType>
```

3.10.1.35 format (PivotTable Format)

Represents the format defined in the PivotTable.

Parent Elements
formats (§3.10.1.36)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
pivotArea (Pivot Area)	§3.3.1.66

Attributes	Description
action (Format Action)	Specifies the formatting behavior for the area indicated in the pivotArea element. The default value for this attribute is "formatting," which indicates that the specified cells have some formatting applied. The format is specified in the dxfId attribute. If the formatting is cleared from the cells, then the value of this attribute becomes "blank." The possible values for this attribute are defined by the ST_FormatAction simple type (§3.18.35).
dxfId (Format Id)	Specifies the identifier of the format the application is currently using for the PivotTable. Formatting information is written to the styles part. See the Styles section (§3.8) for more information on formats. Note: Formatting information provided by cell table and by PivotTable need not agree. If the two formats differ, the cell-level formatting takes precedence. If you change the layout the PivotTable, the PivotTable formatting will then take precedence. The possible values for this attribute are defined by the ST_DxfId simple type (§3.18.26).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Format">
   <sequence>
      <element name="pivotArea" type="CT_PivotArea"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="action" type="ST_FormatAction" default="formatting"/>
   <attribute name="dxfId" type="ST_DxfId" use="optional"/>
</complexType>
```

3.10.1.36 formats (PivotTable Formats)

Represents the collection of formats applied to PivotTable.

Parent Elements
pivotTableDefinition (§3.10.1.73)

Child Elements	Subclause
format (PivotTable Format)	§3.10.1.35

Attributes	Description
count (Formats Count)	Specifies the number of formats in the collection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Formats">
   <sequence>
      <element name="format" maxOccurs="unbounded" type="CT_Format"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" default="0"/>
</complexType>
```

3.10.1.37 group (OLAP Group)

Represents an OLAP level group.

[*Example*:

```
<group name="CategoryXl_Grp_1" uniqueName="[Product].[Product Categories].
  [Product Categories1].[GROUPMEMBER.[CategoryXl_Grp_1]].[Product]].
  [Product Categories]].[All Products]]]" caption="Group1"
  uniqueParent="[Product].[Product Categories].[All Products]" id="1">
```

```
    <groupMembers count="2">
      <groupMember
        uniqueName="[Product].[Product Categories].[Category].&amp;[4]"/>
      <groupMember
        uniqueName="[Product].[Product Categories].[Category].&amp;[1]"/>
    </groupMembers>
  </group>
```

end example]

Parent Elements
groups (§3.10.1.43)

Child Elements	Subclause
groupMembers (OLAP Group Members)	§3.10.1.42

Attributes	Description
caption (Group Caption)	Specifies the caption for this group. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
id (Group Id)	Specifies the unique number for this group within the level. The possible values for this attribute are defined by the XML Schema int datatype.
name (Group Name)	Specifies the name of this group. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
uniqueName (Unique Group Name)	Specifies the unique name of this group. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
uniqueParent (Parent Unique Name)	Specifies the unique name of the parent of this group. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LevelGroup">
   <sequence>
      <element name="groupMembers" type="CT_GroupMembers"/>
   </sequence>
   <attribute name="name" use="required" type="ST_Xstring"/>
   <attribute name="uniqueName" use="required" type="ST_Xstring"/>
   <attribute name="caption" use="required" type="ST_Xstring"/>
   <attribute name="uniqueParent" type="ST_Xstring"/>
   <attribute name="id" type="xsd:int"/>
</complexType>
```

3.10.1.38 groupItems (OLAP Group Items)

Represents the collection of items in a field group.

[*Example*:

```
…
  <fieldGroup par="6" base="0">
    <rangePr groupBy="months" startDate="2002-01-01T00:00:00"
      endDate="2006-05-06T00:00:00"/>
    <groupItems count="14">
      <s v="&lt;1/1/2002"/>
      <s v="Jan"/>
      <s v="Feb"/>
      <s v="Mar"/>
      <s v="Apr"/>
      <s v="May"/>
      <s v="Jun"/>
      <s v="Jul"/>
      <s v="Aug"/>
      <s v="Sep"/>
      <s v="Oct"/>
      <s v="Nov"/>
      <s v="Dec"/>
      <s v="&gt;5/6/2006"/>
    </groupItems>
  </fieldGroup>
</cacheField>
```

```
<cacheField name="Name" numFmtId="0">
  <sharedItems count="4">
    <s v="Joe"/>
    <s v="John"/>
    <s v="Bob"/>
    <s v="Robert"/>
  </sharedItems>
  <fieldGroup par="4"/>
</cacheField>
<cacheField name="ProductID" numFmtId="0">
  <sharedItems containsSemiMixedTypes="0" containsString="0" containsNumber="1"
    containsInteger="1" minValue="1" maxValue="4" count="4">
    <n v="1"/>
    <n v="2"/>
    <n v="3"/>
    <n v="4"/>
  </sharedItems>
  <fieldGroup base="2">
    <rangePr startNum="1" endNum="4" groupInterval="2"/>
    <groupItems count="4">
      <s v="&lt;1"/>
      <s v="1-2"/>
      <s v="3-4"/>
      <s v="&gt;5"/>
    </groupItems>
  </fieldGroup>
…
```

end example]

Parent Elements
fieldGroup (§3.10.1.30)

Child Elements	Subclause
b (Boolean)	§3.10.1.2
d (Date Time)	§3.10.1.21
e (Error Value)	§3.10.1.27
m (No Value)	§3.10.1.50
n (Numeric)	§3.10.1.60
s (Character Value)	§3.10.1.85

Attributes	Description
count (Items Created Count)	Specifies the number of items created for this grouped field. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GroupItems">
   <choice maxOccurs="unbounded">
      <element name="m" type="CT_Missing"/>
      <element name="n" type="CT_Number"/>
      <element name="b" type="CT_Boolean"/>
      <element name="e" type="CT_Error"/>
      <element name="s" type="CT_String"/>
      <element name="d" type="CT_DateTime"/>
   </choice>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.39 groupLevel (OLAP Grouping Levels)

Represents the collection of OLAP grouping levels.

[*Example*:

```
<groupLevel uniqueName="[Product].[Product Categories].[Category]"
  caption="Category">
  <groups count="1">
    <group name="CategoryXl_Grp_1" uniqueName="[Product].[Product
      Categories].[Product Categories1].
      [GROUPMEMBER.[CategoryXl_Grp_1]].[Product]].[Product Categories]].
      [All Products]]]" caption="Group1" uniqueParent="[Product].
      [Product Categories].[All Products]" id="1">
      <groupMembers count="2">
        <groupMember
          uniqueName="[Product].[Product Categories].[Category].&amp;[4]"/>
        <groupMember
          uniqueName="[Product].[Product Categories].[Category].&amp;[1]"/>
      </groupMembers>
    </group>
  </groups>
</groupLevel>
```

end example]

Parent Elements
groupLevels (§3.10.1.40)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
groups (OLAP Level Groups)	§3.10.1.43

Attributes	Description
caption (Grouping Level Display Name)	Specifies the display name for this grouping level. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
customRollUp (Custom Roll Up)	Specifies a boolean value that indicates whether this group level has a custom roll up. A value of on, 1, or true indicates this group level has a custom roll up. A value of off, 0, or false indicates this group level does not have a custom roll up. The possible values for this attribute are defined by the XML Schema boolean datatype.
uniqueName (Unique Name)	Specifies the unique name for this grouping level. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
user (User-Defined Group Level)	Specifies a boolean value that indicates whether this is a user-defined group level. A value of on, 1, or true indicates this is a user-defined group. A value of off, 0, or false indicates this group is not user-defined. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GroupLevel">
   <sequence>
      <element name="groups" minOccurs="0" type="CT_Groups"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="uniqueName" use="required" type="ST_Xstring"/>
   <attribute name="caption" use="required" type="ST_Xstring"/>
   <attribute name="user" type="xsd:boolean" default="false"/>
   <attribute name="customRollUp" type="xsd:boolean" default="false"/>
</complexType>
```

3.10.1.40 groupLevels (OLAP Grouping Levels)

Represents the collection of OLAP grouping levels.

[*Example*:

```
<groupLevels count="5">
  <groupLevel uniqueName="[Product].[Product Categories].[(All)]"
    caption="(All)"/>
  <groupLevel uniqueName="[Product].[Product Categories].[Product Categories1]"
    caption="Product Categories1" user="1"/>
  <groupLevel uniqueName="[Product].[Product Categories].[Category]"
    caption="Category">
    <groups count="1">
      <group name="CategoryXl_Grp_1" uniqueName="[Product].[Product Categories].
        [Product Categories1].[GROUPMEMBER.[CategoryXl_Grp_1]].
        [Product]].[Product Categories]].[All Products]]]" caption="Group1"
        uniqueParent="[Product].[Product Categories].[All Products]" id="1">
        <groupMembers count="2">
          <groupMember
            uniqueName="[Product].[Product Categories].[Category].&amp;[4]"/>
          <groupMember
            uniqueName="[Product].[Product Categories].[Category].&amp;[1]"/>
        </groupMembers>
      </group>
    </groups>
  </groupLevel>
  <groupLevel uniqueName="[Product].[Product Categories].[Subcategory]"
    caption="Subcategory"/>
  <groupLevel uniqueName="[Product].[Product Categories].[Product]"
    caption="Product"/>
</groupLevels>
```

end example]

Parent Elements
cacheHierarchy (§3.10.1.6)

Child Elements	Subclause
groupLevel (OLAP Grouping Levels)	§3.10.1.39

Attributes	Description
count (Grouping Level Count)	Specifies the number of grouping levels. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GroupLevels">
    <sequence>
        <element name="groupLevel" maxOccurs="unbounded" type="CT_GroupLevel"/>
    </sequence>
    <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.41 groupMember (OLAP Group Member)

Represents an OLAP group member.

[*Example*:

```
<groupMember uniqueName="[Product].[Product Categories].[Category].&amp;[1]"/>
```

end example]

Parent Elements
groupMembers (§3.10.1.42)

Attributes	Description
group (Group)	Specifies a boolean value that indicates whether this member represents a group. A value of on, 1, or true indicates this member represents a group. A value of off, 0, or false indicates this member does not represent a group. The possible values for this attribute are defined by the XML Schema boolean datatype.
uniqueName (Group Member Unique Name)	Specifies the unique name of this group member. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GroupMember">
    <attribute name="uniqueName" use="required" type="ST_Xstring"/>
    <attribute name="group" type="xsd:boolean" default="false"/>
</complexType>
```

3.10.1.42 groupMembers (OLAP Group Members)

Represents the collection of OLAP group members.

[*Example*:

```
<groupMembers count="2">
  <groupMember uniqueName="[Product].[Product Categories].[Category].&amp;[4]"/>
  <groupMember uniqueName="[Product].[Product Categories].[Category].&amp;[1]"/>
</groupMembers>
```

end example]

Parent Elements
group (§3.10.1.37)

Child Elements	Subclause
groupMember (OLAP Group Member)	§3.10.1.41

Attributes	Description
count (Group Member Count)	Specifies the number of group members in the collection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GroupMembers">
   <sequence>
      <element name="groupMember" maxOccurs="unbounded" type="CT_GroupMember"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.43 groups (OLAP Level Groups)

Represents the collection of OLAP level groups.

[*Example*:

```
<groups count="1">
  <group name="CategoryXl_Grp_1" uniqueName="[Product].[Product Categories].
    [Product Categories1].[GROUPMEMBER.[CategoryXl_Grp_1]].[Product]].
    [Product Categories]].[All Products]]]" caption="Group1"
    uniqueParent="[Product].[Product Categories].[All Products]" id="1">
```

```
      <groupMembers count="2">
        <groupMember
          uniqueName="[Product].[Product Categories].[Category].&amp;[4]"/>
        <groupMember
          uniqueName="[Product].[Product Categories].[Category].&amp;[1]"/>
      </groupMembers>
    </group>
  </groups>
```

end example]

Parent Elements
groupLevel (§3.10.1.39)

Child Elements	Subclause
group (OLAP Group)	§3.10.1.37

Attributes	Description
count (Level Group Count)	Specifies the number of level groups in the collection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Groups">
   <sequence>
      <element name="group" maxOccurs="unbounded" type="CT_LevelGroup"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.44 i (Row Items)

Represents the collection of items in the row region of the PivotTable. In this example the item values are found in cells B10:B13. For example "Bikes" is in B10, and corresponds to the first <i> element below.

[*Example*:

```
  <rowItems count="4">
    <i>
      <x/>
    </i>
```

```
    <i r="1">
      <x/>
    </i>
    <i r="1">
      <x v="1"/>
    </i>
    <i t="grand">
      <x/>
    </i>
  </rowItems>
```

end example]

Parent Elements
colItems (§3.10.1.17); rowItems (§3.10.1.84)

Child Elements	Subclause
x (Member Property Index)	§3.10.1.96

Attributes	Description
i (Data Field Index)	Specifies a zero-based index indicating the referenced data item it in a data field with multiple data items. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
r (Repeated Items Count)	Specifies the number of items to repeat from the previous row item. Note: The first item has no @r explicitly written. Since a default of "0" is specified in the schema, for any item whose @r is missing, a default value of "0" is implied. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
t (Item Type)	Specifies the type of the item. Value of 'default' indicates a grand total as the last row item value The possible values for this attribute are defined by the ST_ItemType simple type (§3.18.45).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_I">
   <sequence>
      <element name="x" minOccurs="0" maxOccurs="unbounded" type="CT_X"/>
   </sequence>
   <attribute name="t" type="ST_ItemType" default="data"/>
   <attribute name="r" type="xsd:unsignedInt" default="0"/>
   <attribute name="i" type="xsd:unsignedInt" default="0"/>
</complexType>
```

3.10.1.45 item (PivotTable Field Item)

Represents a single item in PivotTable field.

[*Example*:

```
<sh:item x="66"/>
```

end example]

Parent Elements
items (§3.10.1.46)

Attributes	Description
c (Child Items)	Specifies a boolean value that indicates whether the approximate number of child items for this item is greater than zero. A value of on, 1, or true indicates the approximate number of child items for this item is greater than zero. A value of off, 0, or false indicates the approximate number of child items for this item is zero. The possible values for this attribute are defined by the XML Schema boolean datatype.
d (Expanded)	Specifies a boolean value that indicates whether this item has been expanded in the PivotTable view. A value of on, 1, or true indicates this item has been expanded. A value of off, 0, or false indicates this item is collapsed. The possible values for this attribute are defined by the XML Schema boolean datatype.
e (Drill Across Attributes)	Specifies a boolean value that indicates whether attribute hierarchies nested next to each other on a PivotTable row or column will offer drilling "across" each other or not. For example, if the application offers drill across for attribute hierarchies and not for user hierarchies, this attribute would only be written when two attribute hierarchies are

Attributes	Description
	placed next to each other on an axis. A value of on, 1, or true indicates there is a drill across attribute hierarchies positioned next to each other on a pivot axis. A value of off, 0, or false indicates there is not drill across attribute hierarchies. The possible values for this attribute are defined by the XML Schema boolean datatype.
f (Calculated Member)	Specifies a boolean value that indicates whether this item is a calculated member. A value of on, 1, or true indicates this item is a calculated member. A value of off, 0, or false indicates this item is not calculated. The possible values for this attribute are defined by the XML Schema boolean datatype.
h (Hidden)	Specifies a boolean value that indicates whether the item is hidden. A value of on, 1, or true indicates item is hidden. The possible values for this attribute are defined by the XML Schema boolean datatype.
m (Missing)	Specifies a boolean value that indicate whether the item has a missing value. A value of on, 1, or true indicates the item value is missing. The application should still retain the item settings in case the item reappears during a later refresh. The possible values for this attribute are defined by the XML Schema boolean datatype.
n (Item User Caption)	Specifies the user caption of the item. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
s (Character)	Specifies a boolean value that indicates whether the item has a character value. A value of on, 1, or true indicates the item has a string/character value. A value of off, 0, or false indicates item the item has a value of a different type. The possible values for this attribute are defined by the XML Schema boolean datatype.
sd (Hide Details)	Specifies a boolean value that indicates whether the details are hidden for this item. A value of on, 1, or true indicates item details are hidden. A value of off, 0, or false indicates item details are shown. The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description
t (Item Type)	Specifies the type of this item. A value of 'default' indicates the subtotal or total item. The possible values for this attribute are defined by the ST_ItemType simple type (§3.18.45).
x (Item Index)	Specifies the item index in pivotFields collection in the PivotCache. Applies only non-OLAP PivotTables. In the following example, "Product Category" and "Product Subcategory" are on the row axis of the PivotTable, in that order. *<rowFields count="2"> <field x="7"/> <x="8"/> </rowFieldsfield >* The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

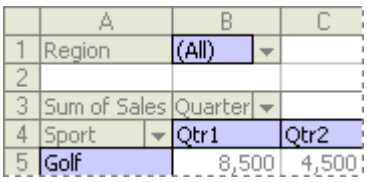
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Item">
   <attribute name="n" type="ST_Xstring"/>
   <attribute name="t" type="ST_ItemType" default="data"/>
   <attribute name="h" type="xsd:boolean" default="false"/>
   <attribute name="s" type="xsd:boolean" default="false"/>
   <attribute name="sd" type="xsd:boolean" default="true"/>
   <attribute name="f" type="xsd:boolean" default="false"/>
   <attribute name="m" type="xsd:boolean" default="false"/>
   <attribute name="c" type="xsd:boolean" default="false"/>
   <attribute name="x" type="xsd:unsignedInt" use="optional"/>
   <attribute name="d" type="xsd:boolean" default="false"/>
   <attribute name="e" type="xsd:boolean" default="true"/>
</complexType>
```

3.10.1.46 items (Field Items)

Represents the collection of items in a PivotTable field. The items in the collection are ordered by index. Items represent the unique entries from the field in the source data.

In the following image, the item Golf represents all rows of data in the source list for which the Sport field contains the entry Golf.



The order in which the items are listed is the order they would appear on a particular axis (row or column, for example).

In the following SpreadsheetML example, the first field is "Customer Name" and the first item referenced here is <item x="66"/> which references the value "Adam L Flores" in the pivotCacheDefinition. Therefore f you added "Customer Name" to the row axis, "Adam L Flores" would be the first row item listed.

[*Example*:

```
<pivotFields count="28">
  <pivotField showAll="0" includeNewItemsInFilter="1">
    <items count="8">
      <item x="66"/>
      <item x="133"/>
      <item x="74"/>
      <item x="27"/>
      <item x="118"/>
      <item x="63"/>
      <item x="141"/>
      <item t="default"/>
    </items>
  </pivotField>
  <pivotField showAll="0" includeNewItemsInFilter="1"/>
  <pivotField axis="axisPage" showAll="0" includeNewItemsInFilter="1">
    <items count="2">
      <item x="0"/>
      <item t="default"/>
    </items>
  </pivotField>
<pivotField showAll="0" includeNewItemsInFilter="1"/>
```

end example]

Parent Elements
pivotField (§3.10.1.69)

Child Elements	Subclause
item (PivotTable Field Item)	§3.10.1.45

Attributes	Description
count (Field Count)	Specifies the number of fields in the PivotTable.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Items">
   <sequence>
      <element name="item" maxOccurs="unbounded" type="CT_Item"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.47 kpi (OLAP KPI)

Represents the KPI defined on the OLAP server and stored in the PivotCache.

[*Example*:

```
<kpi uniqueName="Growth in Customer Base" caption="Growth in Customer Base"
  displayFolder="Customer Perspective\Expand Customer Base"
  measureGroup="Internet Sales" value="[Measures].[Growth in Customer Base]"
  goal="[Measures].[Growth in Customer Base Goal]"
  status="[Measures].[Growth in Customer Base Status]"
  trend="[Measures].[Growth in Customer Base Trend]"/>
```

end example]

Parent Elements
kpis (§3.10.1.48)

Attributes	Description
caption (KPI Display Name)	Specifies the display name of the KPI. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
displayFolder (KPI Display Folder)	Specifies the folder where this KPI will be displayed in a list of fields for the PivotTable. This attribute depends on how the application exposes a list of fields in the user interface. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
goal (KPI Goal Unique Name)	Specifies the unique name of the KPI goal measure. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

Attributes	Description
measureGroup (KPI Measure Group Name)	Specifies the name of the measure group to which this KPI belongs. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
parent (Parent KPI)	Specifies the name of the parent KPI for this KPI. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
status (KPI Status Unique Name)	Specifies the unique name of the KPI status measure. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
time (Time Member KPI Unique Name)	Specifies the unique name of the KPI current time member. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
trend (KPI Trend Unique Name)	Specifies the unique name of the KPI trend measure. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
uniqueName (KPI Unique Name)	Specifies the unique name of the KPI. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
value (KPI Value Unique Name)	Specifies the unique name of the KPI value measure. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
weight (KPI Weight Unique Name)	Specifies the unique name of the KPI weight measure. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PCDKPI">
   <attribute name="uniqueName" use="required" type="ST_Xstring"/>
   <attribute name="caption" use="optional" type="ST_Xstring"/>
   <attribute name="displayFolder" type="ST_Xstring"/>
   <attribute name="measureGroup" type="ST_Xstring"/>
   <attribute name="parent" type="ST_Xstring"/>
   <attribute name="value" use="required" type="ST_Xstring"/>
   <attribute name="goal" type="ST_Xstring"/>
   <attribute name="status" type="ST_Xstring"/>
   <attribute name="trend" type="ST_Xstring"/>
   <attribute name="weight" type="ST_Xstring"/>
   <attribute name="time" type="ST_Xstring"/>
</complexType>
```

3.10.1.48 kpis (OLAP KPIs)

Represents the collection of Key Performance Indicators (KPIs) defined on the OLAP server and stored in the PivotCache.

[*Example*:

```
<kpis count="3">
  <kpi uniqueName="Growth in Customer Base" caption="Growth in Customer Base"
    displayFolder="Customer Perspective\Expand Customer Base"
    measureGroup="Internet Sales" value="[Measures].[Growth in Customer Base]"
    goal="[Measures].[Growth in Customer Base Goal]"
    status="[Measures].[Growth in Customer Base Status]"
    trend="[Measures].[Growth in Customer Base Trend]"/>
  <kpi uniqueName="Net Income" caption="Net Income"
    displayFolder="Financial Perspective\Maintain Overall Margins"
    measureGroup="Financial Reporting" value="[Measures].[Net Income Value]"
    goal="[Measures].[Net Income Goal]" status="[Measures].[Net Income Status]"
    trend="[Measures].[Net Income Trend]"/>
  <kpi uniqueName="Operating Profit" caption="Operating Profit"
    displayFolder="Financial Perspective\Maintain Overall Margins"
    measureGroup="Financial Reporting" parent="Net Income"
    value="[Measures].[Operating Profit Value]"
    goal="[Measures].[Operating Profit Goal]"
    status="[Measures].[Operating Profit Status]"
    trend="[Measures].[Operating Profit Trend]"/>
…
```

end example]

Parent Elements
pivotCacheDefinition (§3.10.1.67)

Child Elements	Subclause
kpi (OLAP KPI)	§3.10.1.47

Attributes	Description
count (KPI Count)	Specifies the number of KPIs stored in the PivotCache. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PCDKPIs">
   <sequence>
      <element name="kpi" minOccurs="0" maxOccurs="unbounded" type="CT_PCDKPI"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.49 location (PivotTable Location)

Represents location information for the PivotTable.

[*Example*:

```
<location ref="B6:G13" firstHeaderRow="1" firstDataRow="4" firstDataCol="1"
  rowPageCount="3" colPageCount="1"/>
```

end example]

Parent Elements
pivotTableDefinition (§3.10.1.73)

Attributes	Description
colPageCount (Columns Per Page)	Specifies the number of columns per page for this PivotTable that the filter area will occupy. By default there is a single column of filter fields per page and the fields occupy as many rows as there are fields. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
firstDataCol (First Data Column)	Specifies the first column of the PivotTable data, relative to the top left cell in the `ref` value. The possible values for this attribute are defined by the XML Schema unsignedInt

Attributes	Description
	datatype.
firstDataRow (PivotTable Data First Row)	Specifies the first row of the PivotTable data, relative to the top left cell in the `ref` value. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
firstHeaderRow (First Header Row)	Specifies the first row of the PivotTable header, relative to the top left cell in the `ref` value. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
ref (Reference)	Specifies the first row of the PivotTable. The possible values for this attribute are defined by the ST_Ref simple type (§3.18.64).
rowPageCount (Rows Per Page Count)	Specifies the number of rows per page for this PivotTable that the filter area will occupy. By default there is a single column of filter fields per page and the fields occupy as many rows as there are fields. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Location">
   <attribute name="ref" use="required" type="ST_Ref"/>
   <attribute name="firstHeaderRow" use="required" type="xsd:unsignedInt"/>
   <attribute name="firstDataRow" use="required" type="xsd:unsignedInt"/>
   <attribute name="firstDataCol" use="required" type="xsd:unsignedInt"/>
   <attribute name="rowPageCount" type="xsd:unsignedInt" default="0"/>
   <attribute name="colPageCount" type="xsd:unsignedInt" default="0"/>
</complexType>
```

3.10.1.50 m (No Value)

Represents a value that was not specified.

[*Example*:

```
<sharedItems containsString="0" containsBlank="1" count="1">
  <m/>
</sharedItems>
```

end example]

Parent Elements
entries (§3.10.1.28); groupItems (§3.10.1.38); r (§3.10.1.77); sharedItems (§3.10.1.90)

Child Elements	Subclause
tpls (Tuples)	§3.10.1.93
x (Member Property Index)	§3.10.1.96

Attributes	Description
b (Bold)	Specifies a boolean value that indicates whether the value contains bold formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of on, 1, or true indicates this value contains bold formatting on the server. The possible values for this attribute are defined by the XML Schema boolean datatype.
bc (background Color)	Specifies the background color for this value that was provided by the OLAP server. This attribute applies to OLAP-based PivotTables only. The color is specified as a HEX value in RGB space. The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§3.18.86).
c (Caption)	Specifies the caption for this item. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
cp (Member Property Count)	Specifies the number of member property values for this item. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
f (Calculated Item)	Specifies a boolean value that indicates whether this is a calculated item value. A value of on, 1, or true indicates this item is a calculated value. A value of off, 0, or false indicates this item is not calculated. The possible values for this attribute are defined by the XML Schema boolean datatype.
fc (Foreground Color)	Specifies the foreground color for this value that was provided by the OLAP server. This attribute applies to OLAP-based PivotTables only. The color is specified as a HEX value in RGB space. The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§3.18.86).
i (Italic)	Specifies a boolean value that indicates whether the value contains italic formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of on, 1, or true indicates this value contains underline formatting on the server.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema boolean datatype.
in (Format Index)	Specifies the index to the OLAP serverformat element where the format string for this entry is stored. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
st (Strikethrough)	Specifies a boolean value that indicates whether the value contains strikethrough formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of on, 1, or true indicates this value contains underline formatting on the server. The possible values for this attribute are defined by the XML Schema boolean datatype.
u (Unused Item)	Specifies a boolean value that indicates whether this is an unused item. The application marks an item as unused when an item is deleted from the data source. The item and associated metadata are retained in the cache until the threshold for unused items specified in missingItemsLimit is reached. A value of on, 1, or true indicates this item is unused. A value of off, 0, or false indicates this item is used. The possible values for this attribute are defined by the XML Schema boolean datatype.
un (Underline)	Specifies a boolean value that indicates whether the value contains underline formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of on, 1, or true indicates this value contains underline formatting on the server. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Missing">
   <sequence>
      <element name="tpls" minOccurs="0" maxOccurs="unbounded" type="CT_Tuples"/>
      <element name="x" minOccurs="0" maxOccurs="unbounded" type="CT_X"/>
   </sequence>
   <attribute name="u" type="xsd:boolean"/>
   <attribute name="f" type="xsd:boolean"/>
   <attribute name="c" type="ST_Xstring"/>
   <attribute name="cp" type="xsd:unsignedInt"/>
   <attribute name="in" type="xsd:unsignedInt" use="optional"/>
   <attribute name="bc" type="ST_UnsignedIntHex" use="optional"/>
   <attribute name="fc" type="ST_UnsignedIntHex" use="optional"/>
   <attribute name="i" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="un" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="st" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="b" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.10.1.51 map (OLAP Measure Group)

Represents a PivotTable OLAP measure group - Dimension map.

[*Example*:

```
<map measureGroup="0" dimension="2"/>
```

end example]

Parent Elements
maps (§3.10.1.52)

Attributes	Description
dimension (Dimension Id)	Specifies the identifier for the dimension. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
measureGroup (Measure Group Id)	Specifies the identifier of the measure group. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MeasureDimensionMap">
   <attribute name="measureGroup" use="optional" type="xsd:unsignedInt"/>
   <attribute name="dimension" use="optional" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.52 maps (OLAP Measure Group)

Represents the PivotTable OLAP measure group - Dimension maps.

[*Example*:

```
<maps count="3">
  <map measureGroup="0" dimension="2"/>
  <map measureGroup="1" dimension="19"/>
  <map measureGroup="2" dimension="8"/>
</maps>
```

end example]

Parent Elements
pivotCacheDefinition (§3.10.1.67)

Child Elements	Subclause
map (OLAP Measure Group)	§3.10.1.51

Attributes	Description
count (Measure Group Count)	Specifies the number of measure groups, or dimension maps, in the PivotTable. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MeasureDimensionMaps">
    <sequence>
        <element name="map" minOccurs="0" maxOccurs="unbounded" type="CT_MeasureDimensionMap"/>
    </sequence>
    <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.53 measureGroup (OLAP Measure Group)

Represents a PivotTable OLAP measure group.

[*Example*:

```
<measureGroup name="Sales Orders" caption="Sales Orders"/>
```

end example]

Parent Elements

Parent Elements
measureGroups (§3.10.1.54)

Attributes	Description
caption (Measure Group Display Name)	Specifies the display name of the measure group. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
name (Measure Group Name)	Specifies the name of the measure group. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MeasureGroup">
   <attribute name="name" use="required" type="ST_Xstring"/>
   <attribute name="caption" use="required" type="ST_Xstring"/>
</complexType>
```

3.10.1.54 measureGroups (OLAP Measure Groups)

Represents the collection of PivotTable OLAP measure groups.

[*Example*:

```
<measureGroups count="11">
  <measureGroup name="Exchange Rates" caption="Exchange Rates"/>
  <measureGroup name="Financial Reporting" caption="Financial Reporting"/>
  <measureGroup name="Internet Customers" caption="Internet Customers"/>
  <measureGroup name="Internet Orders" caption="Internet Orders"/>
  <measureGroup name="Internet Sales" caption="Internet Sales"/>
  <measureGroup name="Reseller Orders" caption="Reseller Orders"/>
  <measureGroup name="Reseller Sales" caption="Reseller Sales"/>
  <measureGroup name="Sales Orders" caption="Sales Orders"/>
  <measureGroup name="Sales Reasons" caption="Sales Reasons"/>
  <measureGroup name="Sales Summary" caption="Sales Summary"/>
  <measureGroup name="Sales Targets" caption="Sales Targets"/>
</measureGroups>
```

end example]

Parent Elements
pivotCacheDefinition (§3.10.1.67)

Child Elements	Subclause
measureGroup (OLAP Measure Group)	§3.10.1.53

Attributes	Description
count (Measure Group Count)	Specifies the number of measure groups in the PivotTable. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MeasureGroups">
   <sequence>
      <element name="measureGroup" minOccurs="0" maxOccurs="unbounded" type="CT_MeasureGroup"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.55 member (Member)

Represents an item that may be included or excluded.

Parent Elements
members (§3.10.1.56)

Attributes	Description
name (Hidden Item Name)	Specifies the name of a hidden item. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Member">
   <attribute name="name" use="required" type="ST_Xstring"/>
</complexType>
```

3.10.1.56 members (Members)

Represents the collection of items that may be included or excluded.

Parent Elements
pivotHierarchy (§3.10.1.72)

Child Elements	Subclause

Child Elements	Subclause
member (Member)	§3.10.1.55

Attributes	Description
count (Item Count)	Specifies the number of items in the collection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
level (Hierarchy Level)	Specifies the hierarchy level with which these items are associated. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Members">
   <sequence>
      <element name="member" maxOccurs="unbounded" type="CT_Member"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
   <attribute name="level" use="optional" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.57 mp (OLAP Member Property)

Represents an OLAP member property.

[*Example*:

```
<sh:mp field="7"/>
```

end example]

Parent Elements
mps (§3.10.1.59)

Attributes	Description
field (Field Index)	Specifies the index of the field with which this member property is associated. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
level (Level Index)	Specifies the index of the level to which this member property applies. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

Attributes	Description
name (OLAP Member Property Unique Name)	Specifies the unique name of the OLAP member property. The following attributes depend on the name attribute: • nameLen • pLen • pPos These attributes consist of metadata about a member in an OLAP cube and are usually displayed in a tooltip or mechanism in the user interface. For example, if the value for name equals "[Store].[Store Name].[Store Manager]": • nameLen will equal 20. This would refer to "[Store].[Store Name]" • pPos will equal 22. This would refer to starting character of "Store Manager" • pLen will equal 13. This would to length of "Store Manager" The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
nameLen (Name Length)	Specifies the length of the unique name portion of name. For example, if the value for name equals "[Store].[Store Name].[Store Manager]", nameLen will equal 20. This would refer to "[Store].[Store Name]". The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
pLen (Property Name Length)	Specifies the length of the property name portion of name. For example, if the value for name equals "[Store].[Store Name].[Store Manager]", pLen will equal 13. This would to length of "Store Manager". The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
pPos (Property Name Character Index)	Specifies the index of the character where the property name portion begins in name. For example, if the value for name equals "[Store].[Store Name].[Store Manager]", pPos will equal 22. This would refer to starting character of "Store Manager". The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
showAsCaption (Show As Caption)	Specifies a boolean value that indicates whether to show the property a member caption. A value of on, 1, or true indicates member property value will be shown in as a caption. A value of off, 0, or false indicates member property value will not be shown. The possible values for this attribute are defined by the XML Schema boolean datatype.
showCell (Show Cell)	Specifies a boolean value that indicates whether to show the member property value in a PivotTable cell.

Attributes	Description
	A value of on, 1, or true indicates member property value will be shown in a cell. A value of off, 0, or false indicates member property value will not be shown. The possible values for this attribute are defined by the XML Schema boolean datatype.
showTip (Show Tooltip)	Specifies a boolean value that indicates whether to show the member property value in a tooltip on the appropriate PivotTable view cells. A value of on, 1, or true indicates member property value will be shown in a tooltip. A value of off, 0, or false indicates member property value will not be shown. This attribute depends on whether the application employs tooltips or similar mechanism in the user interface. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MemberProperty">
    <attribute name="name" type="ST_Xstring" use="optional"/>
    <attribute name="showCell" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="showTip" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="showAsCaption" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="nameLen" type="xsd:unsignedInt" use="optional"/>
    <attribute name="pPos" type="xsd:unsignedInt" use="optional"/>
    <attribute name="pLen" type="xsd:unsignedInt" use="optional"/>
    <attribute name="level" type="xsd:unsignedInt" use="optional"/>
    <attribute name="field" use="required" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.58 mpMap (Member Properties Map)

Represents a mapping to cached member properties.

[*Example*:

```
<mpMap v="7"/>
```

end example]

Parent Elements
cacheField (§3.10.1.3)

Attributes	Description
v (Shared Items Index)	Specifies the index into the shared items table in the PivotCache that identifies this item.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema int datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_X">
   <attribute name="v" type="xsd:int" default="0"/>
</complexType>
```

3.10.1.59 mps (OLAP Member Properties)

Represents the collection of OLAP member property. Member properties contain additional information that's available about the items in an OLAP dimension field. For example, if a Geography dimension has property fields Population and Average Income available, you could create a PivotTable report that displays the sales figures for cities where your products are selling well. By displaying and analyzing the population and income figures for these cities, you could target cities with similar demographics for your marketing campaign.

[*Example*:

```
<sh:mps count="3">
  <sh:mp field="7"/>
  <sh:mp field="8"/>
  <sh:mp field="9"/>
</sh:mps>
```

end example]

Parent Elements
pivotHierarchy (§3.10.1.72)

Child Elements	Subclause
mp (OLAP Member Property)	§3.10.1.57

Attributes	Description
count (OLAP Member Properties Count)	Specifies the number of OLAP member properties in the collection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MemberProperties">
   <sequence>
      <element name="mp" maxOccurs="unbounded" type="CT_MemberProperty"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.60 n (Numeric)

Represents a numeric value in the PivotTable.

[*Example*:

```
<sharedItems containsSemiMixedTypes="0" containsString="0" containsNumber="1"
  containsInteger="1" minValue="3" maxValue="3" count="1">
  <n v="3"/>
</sharedItems>
```

end example]

Parent Elements
entries (§3.10.1.28); groupItems (§3.10.1.38); r (§3.10.1.77); sharedItems (§3.10.1.90)

Child Elements	Subclause
tpls (Tuples)	§3.10.1.93
x (Member Property Index)	§3.10.1.96

Attributes	Description
b (Bold)	Specifies a boolean value that indicates whether this value contains bold formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of on, 1, or true indicates this value contains italic formatting on the server. The possible values for this attribute are defined by the XML Schema boolean datatype.
bc (Background Color)	Specifies the background color for this value that was provided by the OLAP server. This attribute applies to OLAP-based PivotTables only. The color is specified as a HEX value in RGB space. The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§3.18.86).
c (Caption)	Specifies the caption for this item. The possible values for this attribute are defined by the ST_Xstring simple type

Attributes	Description
	(§3.18.96).
cp (Member Property Count)	Specifies the number of member property values for this item. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
f (Calculated Item)	Specifies a boolean value that indicates whether this is a calculated item value. A value of on, 1, or true indicates this item is a calculated value. A value of off, 0, or false indicates this item is not calculated. The possible values for this attribute are defined by the XML Schema boolean datatype.
fc (Foreground Color)	Specifies the foreground color for this value that was provided by the OLAP server. This attribute applies to OLAP-based PivotTables only. The color is specified as a HEX value in RGB space. The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§3.18.86).
i (Italic)	Specifies a boolean value that indicates whether the value contains italic formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of on, 1, or true indicates this value contains italic formatting on the server. The possible values for this attribute are defined by the XML Schema boolean datatype.
in (Format Index)	Specifies the index to the OLAP serverformat element where the format string for this entry is stored. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
st (Strikethrough)	Specifies a boolean value that indicates whether the value contains strikethrough formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of on, 1, or true indicates this value contains strikethrough formatting on the server. The possible values for this attribute are defined by the XML Schema boolean datatype.
u (Unused Item)	Specifies a boolean value that indicates whether this is an unused item. The application marks an item as unused when an item is deleted from the data source. The item and associated metadata are retained in the cache until the threshold for unused items specified in missingItemsLimit is reached. A value of on, 1, or true indicates this item is not used. A value of off, 0, or false indicates this item is used.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema boolean datatype.
un (Underline)	Specifies a boolean value that indicates whether the value contains underline formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of on, 1, or true indicates this value contains underline formatting on the server. The possible values for this attribute are defined by the XML Schema boolean datatype.
v (Value)	Specified the value of this item. The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Number">
   <sequence>
      <element name="tpls" minOccurs="0" maxOccurs="unbounded" type="CT_Tuples"/>
      <element name="x" minOccurs="0" maxOccurs="unbounded" type="CT_X"/>
   </sequence>
   <attribute name="v" use="required" type="xsd:double"/>
   <attribute name="u" type="xsd:boolean"/>
   <attribute name="f" type="xsd:boolean"/>
   <attribute name="c" type="ST_Xstring"/>
   <attribute name="cp" type="xsd:unsignedInt"/>
   <attribute name="in" type="xsd:unsignedInt" use="optional"/>
   <attribute name="bc" type="ST_UnsignedIntHex" use="optional"/>
   <attribute name="fc" type="ST_UnsignedIntHex" use="optional"/>
   <attribute name="i" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="un" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="st" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="b" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.10.1.61 page (Page Items)

Represents the collection of page item values for a page field.

Parent Elements
pages (§3.10.1.65)

Child Elements	Subclause
pageItem (Page Item)	§3.10.1.64

Attributes	Description
count (Page Item	Specifies the number of page item strings in the collectoin.

Attributes	Description
String Count)	The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PCDSCPage">
   <sequence>
      <element name="pageItem" type="CT_PageItem" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.10.1.62 pageField (Page Field)

Represents a field on the page or report filter of the PivotTable.

	A	B	C
1	Region	(All)	
2			
3	Sum of Sales	Quarter	
4	Sport	Qtr1	Qtr2
5	Golf	8,500	4,500

In the image above, the blue field is a page or report filter field. Page/filter fields allow you to filter the entire PivotTable to display data for a single item or all the items.

[*Example*:

```
<sh:pageField fld="43" hier="103"
  name="[Product].[Product Categories].[All Products]" cap="All Products"/>
```

end example]

Parent Elements
pageFields (§3.10.1.63)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10

Attributes	Description
cap (Hierarchy Display Name)	Specifies the display name of the hierarchy. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

Attributes	Description
fld (Field)	Specifies the index of the field that appears on the page or filter report area of the PivotTable. The possible values for this attribute are defined by the XML Schema int datatype.
hier (OLAP Hierarchy Index)	Specifies the index of the OLAP hierarchy to which this item belongs. The possible values for this attribute are defined by the XML Schema int datatype.
item (Item Index)	Specifies the index of the item in the PivotCache. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
name (Hierarchy Unique Name)	Specifies the unique name of the hierarchy. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PageField">
   <sequence minOccurs="0">
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="fld" use="required" type="xsd:int"/>
   <attribute name="item" use="optional" type="xsd:unsignedInt"/>
   <attribute name="hier" type="xsd:int"/>
   <attribute name="name" type="ST_Xstring"/>
   <attribute name="cap" type="ST_Xstring"/>
</complexType>
```

3.10.1.63 pageFields (Page Field Items)

Represents the collection of items in the page or report filter region of the PivotTable.

[*Example*:

```
<sh:pageFields count="2">
  <sh:pageField fld="43" hier="103"
    name="[Product].[Product Categories].[All Products]" cap="All Products"/>
  <sh:pageField fld="66" hier="126"
    name="[Promotion].[Promotions].[All Promotions]" cap="All Promotions"/>
</sh:pageFields>
```

end example]

Parent Elements
pivotTableDefinition (§3.10.1.73)

Child Elements	Subclause
pageField (Page Field)	§3.10.1.62

Attributes	Description
count (Page Item Count)	Specifies the number of items in the page region of the PivotTable. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PageFields">
   <sequence>
      <element name="pageField" maxOccurs="unbounded" type="CT_PageField"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.64 pageItem (Page Item)

Represents an item value for a PivotTable page.

Parent Elements
page (§3.10.1.61)

Attributes	Description
name (Page Item Name)	Specifies the name of this page item. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PageItem">
   <attribute name="name" type="ST_Xstring" use="required"/>
</complexType>
```

3.10.1.65 pages (Page Item Values)

Represents the collection of page item values for each page field.

Parent Elements
consolidation (§3.10.1.20)

Child Elements	Subclause

Child Elements	Subclause
page (Page Items)	§3.10.1.61

Attributes	Description
count (Page Item String Count)	Specifies the number of page item strings in the collection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Pages">
   <sequence>
      <element name="page" type="CT_PCDSCPage" minOccurs="1" maxOccurs="4"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.10.1.66 pivotAreas (Pivot Areas)

Represents the collection of pivot areas that comprise the PivotTable location.

[*Example*:

```
<sh:pivotAreas count="1">
  <sh:pivotArea field="2" dataOnly="0" outline="0"/>
</sh:pivotAreas>
```

end example]

Parent Elements
conditionalFormat (§3.10.1.18)

Child Elements	Subclause
pivotArea (Pivot Area)	§3.3.1.66

Attributes	Description
count (Pivot Area Count)	Specifies the number of PivotAreas for the PivotTable location. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PivotAreas">
   <sequence>
      <element name="pivotArea" minOccurs="0" maxOccurs="unbounded" type="CT_PivotArea"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.67 pivotCacheDefinition (PivotCache Definition)

Represents the pivotCacheDefinition part. This part defines each field in the source data, including the name, the string resources of the instance data (for shared items), and information about the type of data that appears in the field.

[*Example*:

```
<pivotCacheDefinition xmlns="…" xmlns:r="…" r:id="rId1" refreshedBy="AnonUser"
  refreshedDate="2006-05-22T10:07:16" createdVersion="3" refreshedVersion="3"
  minRefreshableVersion="3" recordCount="182">
  <cacheSource type="worksheet">
    <worksheetSource name="Table1"/>
  </cacheSource>
  <cacheFields count="28">
    <cacheField name="Customer Name" numFmtId="0">
    <cacheField name="Postal Code" numFmtId="0">
      <sharedItems/>
    </cacheField>
    <cacheField name="Product Category" numFmtId="0">
      <sharedItems count="1">
        <s v="Bikes"/>
      </sharedItems>
    </cacheField>
    <cacheField name="Year" numFmtId="0">
      <sharedItems count="1">
        <s v="2001"/>
      </sharedItems>
    </cacheField>
    <cacheField name="Quarter" numFmtId="0">
      <sharedItems containsSemiMixedTypes="0" containsString="0"
        containsNumber="1" containsInteger="1" minValue="3" maxValue="3"
        count="1">
        <n v="3"/>
      </sharedItems>
    </cacheField>
  </cacheFields>
</pivotCacheDefinition>
```

end example]

Parent Elements
Root element of SpreadsheetML Pivot Table Cache Definition part

Child Elements	Subclause
cacheFields (PivotCache Fields)	§3.10.1.4
cacheHierarchies (PivotCache Hierarchies)	§3.10.1.5
cacheSource (PivotCache Source Description)	§3.10.1.7
calculatedItems (Calculated Items)	§3.10.1.9
calculatedMembers (Calculated Members)	§3.10.1.11
dimensions (OLAP Dimensions)	§3.10.1.25
extLst (Future Feature Data Storage Area)	§3.2.10
kpis (OLAP KPIs)	§3.10.1.48
maps (OLAP Measure Group)	§3.10.1.52
measureGroups (OLAP Measure Groups)	§3.10.1.54
tupleCache (Tuple Cache)	§3.10.1.94

Attributes	Description
backgroundQuery (Background Query)	Specifies a boolean value that indicates whether the application should query and retrieve records asynchronously from the cache. A value of on, 1, or true indicates the application will retrieve records asynchronously from the cache. A value of off, 0, or false indicates the application will retrieve records synchronously. The possible values for this attribute are defined by the XML Schema boolean datatype.
createdVersion (PivotCache Created Version)	Specifies the version of the application that created the cache. This attribute is application-dependent. The possible values for this attribute are defined by the XML Schema unsignedByte datatype.
enableRefresh (Enable PivotCache Refresh)	Specifies a boolean value that indicates whether the end-user can refresh the cache. This attribute depends on whether the application exposes a method for allowing end-users control over refreshing the cache via the user interface. A value of on, 1, or true indicates the end-user can refresh the cache. A value of off, 0, or false indicates the end-user cannot refresh the cache.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema boolean datatype.
id (Relationship Identifier) Namespace: .../officeDocument/2006/relationships	Specifies the unique identifier that corresponds to the related `pivotCacheRecords` part. See (§3.10.1.68) for more information. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
invalid (Invalid Cache)	Specifies a boolean value that indicates whether the cache is invalid and needs to be refreshed. A value of on, 1, or true indicates the cache is invalid and needs to be refreshed. A value of off, 0, or false indicates the cache does not need to be refreshed. The possible values for this attribute are defined by the XML Schema boolean datatype.
minRefreshableVersion (Minimum Version Required for Refresh)	Specifies the earliest version of the application that is required to refresh the cache. This attribute is application-dependent. The possible values for this attribute are defined by the XML Schema unsignedByte datatype.
missingItemsLimit (Missing Items Limit)	Specifies the number of unused items to allow before discarding unused items. This attribute is application-dependent. The application must specify a threshold for unused items. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
optimizeMemory (Optimize Cache for Memory)	Specifies a boolean value that indicates whether the application will apply optimizations to the cache to reduce memory usage. This attribute is application-dependent. This application must define its own cache optimization methods. The application must also decide whether to expose cache optimization status via the user interface or an object model. A value of on, 1, or true indicates the application will apply optimizations to the cache. A value of off, 0, or false indicates the application will not apply optimizations to the cache. The possible values for this attribute are defined by the XML Schema boolean datatype.
recordCount (PivotCache Record Count)	Specifies the number of records in the cache. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

Attributes	Description
refreshedBy (Last Refreshed By)	Specifies the name of the end-user who last refreshed the cache. This attribute is application-dependent and is specified by applications that track and store the identity of the current user. This attribute also depends on whether the application exposes mechanisms via the user interface whereby the end-user can refresh the cache. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
refreshedDate (PivotCache Last Refreshed Date)	Specifies the date when the cache was last refreshed. This attribute depends on whether the application exposes mechanisms via the user interface whereby the end-user can refresh the cache. The possible values for this attribute are defined by the XML Schema double datatype.
refreshedVersion (PivotCache Last Refreshed Version)	Specifies the version of the application that last refreshed the cache. This attribute depends on whether the application exposes mechanisms via the user interface whereby the end-user can refresh the cache. The possible values for this attribute are defined by the XML Schema unsignedByte datatype.
refreshOnLoad (Refresh On Load)	Specifies a boolean value that indicates whether the application will refresh the cache when the workbook has been opened. A value of on, 1, or true indicates that application will refresh the cache when the workbook is loaded. A value of off, 0, or false indicates the application will not automatically refresh cached data. The end user must trigger refresh of the cache manually via the application user interface. The possible values for this attribute are defined by the XML Schema boolean datatype.
saveData (Save Pivot Records)	Specifies a boolean value that indicates whether the pivot records are saved with the cache. A value of on, 1, or true indicates pivot records are saved in the cache. A value of off, 0, or false indicates are not saved in the cache. The possible values for this attribute are defined by the XML Schema boolean datatype.
supportAdvancedDrill (Supports Attribute Drilldown)	Specifies whether the cache's data source supports attribute drilldown. The possible values for this attribute are defined by the XML Schema boolean datatype.
supportSubquery (Supports Subqueries)	Specifies whether the cache's data source supports subqueries. The possible values for this attribute are defined by the XML Schema boolean datatype.
tupleCache (Stores	Specifies a boolean value that indicates whether the PivotCache is used store information

Attributes	Description
Cache for OLAP Functions)	for OLAP sheet data functions. A value of on, 1, or true indicates information about OLAP sheet data functions are stored in the cache. A value of off, 0, or false indicates the PivotCache does not contain information about OLAP sheet data functions. The possible values for this attribute are defined by the XML Schema boolean datatype.
upgradeOnRefresh (Upgrade PivotCache on Refresh)	Specifies a boolean value that indicates whether the cache is scheduled for version upgrade. This attribute depends on whether the application exposes mechanisms via the user interface whereby the cache may be upgraded. A value of on, 1, or true indicates the cache is scheduled for upgrade. A value of off, 0, or false indicates the cache is not scheduled for upgrade. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PivotCacheDefinition">
   <sequence>
      <element name="cacheSource" type="CT_CacheSource" minOccurs="1" maxOccurs="1"/>
      <element name="cacheFields" type="CT_CacheFields" minOccurs="1" maxOccurs="1"/>
      <element name="cacheHierarchies" minOccurs="0" type="CT_CacheHierarchies"/>
      <element name="kpis" minOccurs="0" type="CT_PCDKPIs"/>
      <element name="tupleCache" minOccurs="0" type="CT_TupleCache"/>
      <element name="calculatedItems" minOccurs="0" type="CT_CalculatedItems"/>
      <element name="calculatedMembers" type="CT_CalculatedMembers" minOccurs="0"/>
      <element name="dimensions" type="CT_Dimensions" minOccurs="0"/>
      <element name="measureGroups" type="CT_MeasureGroups" minOccurs="0"/>
      <element name="maps" type="CT_MeasureDimensionMaps" minOccurs="0"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute ref="r:id" use="optional"/>
   <attribute name="invalid" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="saveData" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="refreshOnLoad" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="optimizeMemory" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="enableRefresh" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="refreshedBy" type="ST_Xstring" use="optional"/>
   <attribute name="refreshedDate" type="xsd:double" use="optional"/>
   <attribute name="backgroundQuery" type="xsd:boolean" default="false"/>
   <attribute name="missingItemsLimit" type="xsd:unsignedInt" use="optional"/>
   <attribute name="createdVersion" type="xsd:unsignedByte" use="optional" default="0"/>
   <attribute name="refreshedVersion" type="xsd:unsignedByte" use="optional" default="0"/>
   <attribute name="minRefreshableVersion" type="xsd:unsignedByte" use="optional" default="0"/>
   <attribute name="recordCount" type="xsd:unsignedInt" use="optional"/>
   <attribute name="upgradeOnRefresh" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="tupleCache" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="supportSubquery" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="supportAdvancedDrill" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.10.1.68 pivotCacheRecords (PivotCache Records)

Represents the collection of records in the PivotCache. This part stores the underlying source data that the PivotTable aggregates.

[*Example*:

```
<pivotCacheRecords xmlns="…" xmlns:r="…" count="2">
  <r>
    <x v="0"/>
    <s v="Pacific"/>
    <x v="0"/>
    <s v="Australia"/>
    <x v="0"/>
```

```
      <x v="0"/>
      <s v="3550"/>
      <x v="0"/>
      <x v="0"/>
      <s v="Road-150 Red, 62"/>
      <s v="This bike is ridden by race winners. Developed with the Adventure
        Works Cycles professional race team, it has a extremely light
        heat-treated aluminum frame, and steering that allows precision
        control."/>
      <s v="No Discount"/>
…
      <n v="89.456800000000001"/>
    </r>
…
```

end example]

Parent Elements
Root element of SpreadsheetML Pivot Table Cache Records part

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
r (PivotCache Record)	§3.10.1.77

Attributes	Description
count (PivotCache Records Count)	Specifies the number of records in the cache. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PivotCacheRecords">
   <sequence>
      <element name="r" minOccurs="0" maxOccurs="unbounded" type="CT_Record"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.69 pivotField (PivotTable Field)

Represents a single field in the PivotTable. This complex type contains information about the field, including the collection of items in the field.

[*Example*:

```
<pivotField axis="axisRow" allDrilled="1" showAll="0" measureFilter="1"
  sortType="descending">
  <items count="8">
    <item s="1" c="1" x="0"/>
    <item s="1" c="1" x="1"/>
    <item c="1" x="2"/>
    <item c="1" x="3"/>
    <item c="1" x="4"/>
    <item c="1" x="5"/>
    <item c="1" x="6"/>
    <item t="default"/>
  </items>
  <autoSortScope>
    <pivotArea dataOnly="0" outline="0" fieldPosition="0">
      <references count="2">
        <reference field="4294967294" count="1" selected="0">
          <x v="0"/>
        </reference>
        <reference field="25" count="1" selected="0">
          <x v="0"/>
        </reference>
      </references>
    </pivotArea>
  </autoSortScope>
</pivotField>
```

end example]

Parent Elements
pivotFields (§3.10.1.70)

Child Elements	Subclause
autoSortScope (AutoSort Scope)	§3.10.1.1
extLst (Future Feature Data Storage Area)	§3.2.10
items (Field Items)	§3.10.1.46

Attributes	Description
allDrilled (All Items Expanded)	Specifies a boolean value that indicates whether all items in the field are expanded. Applies only to OLAP PivotTables.

Attributes	Description
	A value of on, 1, or true indicates all items in the field are expanded. A value of off, 0, or false indicates all items are not expanded. However some items may be expanded. The possible values for this attribute are defined by the XML Schema boolean datatype.
autoShow (Auto Show)	Specifies a boolean value that indicates whether an "AutoShow" filter is applied to this field. This attribute depends on the implementation of filtering in the application. A value of on, 1, or true indicates an "AutoShow" filter is applied to the field. A value of off, 0, or false indicates an "AutoShow" filter is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.
avgSubtotal (Average)	Specifies a boolean value that indicates whether to apply the 'Average' aggregation function in the subtotal of this field. A value of on, 1, or true indicates the subtotal for this field is 'Average.' A value of off, 0, or false indicates a different aggregation function is applied to the subtotal for this field. The possible values for this attribute are defined by the XML Schema boolean datatype.
axis (Axis)	Specifies the region of the PivotTable that this field is displayed. The possible values for this attribute are defined by the ST_Axis simple type (§3.18.1).
compact (Compact)	Specifies a boolean value that indicates whether the application will display fields compactly in the sheet on which this PivotTable resides. A value of on, 1, or true indicates the next field should be displayed in the same column of the sheet. A value of off, 0, or false indicates each pivot field will display in its own column in the sheet. The possible values for this attribute are defined by the XML Schema boolean datatype.
countASubtotal (CountA)	Specifies a boolean value that indicates whether to apply the 'countA' aggregation function in the subtotal of this field. A value of on, 1, or true indicates the subtotal for this field is 'countA.' A value of off, 0, or false indicates a different aggregation function is applied to the subtotal for this field.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema boolean datatype.
countSubtotal (Count)	Specifies a boolean value that indicates whether to apply the 'count' aggregation function in the subtotal of this field. A value of on, 1, or true indicates the subtotal for this field is 'count.' A value of off, 0, or false indicates a different aggregation vfunction is applied to the subtotal for this field. The possible values for this attribute are defined by the XML Schema boolean datatype.
dataField (Data Field)	Specifies a boolean value that indicates whether this field appears in the data region of the PivotTable. A value of on, 1, or true indicates this field appears in the data region of the PivotTable. A value of off, 0, or false indicates this field appears in another region of the PivotTable. The possible values for this attribute are defined by the XML Schema boolean datatype.
dataSourceSort (Data Source Sort)	Specifies a boolean value that indicates whether sort is applied to this field in the data source. A value of on, 1, or true indicates this field is sorted in the data source. A value of off, 0, or false indicates this field is not sorted in the data source. The possible values for this attribute are defined by the XML Schema boolean datatype.
defaultAttributeDrillState (Drill State)	Specifies a boolean value that indicates the drill state of the attribute hierarchy in an OLAP-based PivotTable. A value of on, 1, or true indicates the attribute hierarchy is expanded. A value of off, 0, or false indicates the attribute hierarchy is collapsed. This attribute is designed to allow the application to issue more optimized queries when all items of each field have the same drill state. The possible values for this attribute are defined by the XML Schema boolean datatype.
defaultSubtotal (Show Default Subtotal)	Specifies a boolean value that indicates whether the default subtotal aggregation function is displayed for this field. A value of on, 1, or true indicates the default subtotal aggregation function is displayed for this field.

Attributes	Description
	A value of off, 0, or false indicates the default aggregation function is not displayed. The possible values for this attribute are defined by the XML Schema boolean datatype.
dragOff (Drag Off)	Specifies a boolean value that indicates whether the field can be removed from the PivotTable. A value of on, 1, or true indicates the field can be removed from the PivotTable. A value of off, 0, or false indicates the field cannot be removed from the PivotTable. The possible values for this attribute are defined by the XML Schema boolean datatype.
dragToCol (Drag To Column)	Specifies a boolean value that indicates whether the field can be dragged to the column axis. A value of on, 1, or true indicates the field can be dragged to the column axis. A value of off, 0, or false indicates the field cannot be dragged to the column axis. The possible values for this attribute are defined by the XML Schema boolean datatype.
dragToData (Field Can Drag to Data)	Specifies a boolean value that indicates whether the field can be dragged to the data region. A value of on, 1, or true indicates the field can be dragged to the data region. A value of off, 0, or false indicates the field cannot be dragged to the data region. The possible values for this attribute are defined by the XML Schema boolean datatype.
dragToPage (Drag Field to Page)	Specifies a boolean value that indicates whether the field can be dragged to the page region. A value of on, 1, or true indicates the field can be dragged to the page region. A value of off, 0, or false indicates the field cannot be dragged to the page region. The possible values for this attribute are defined by the XML Schema boolean datatype.
dragToRow (Drag To Row)	Specifies a boolean value that indicates whether the field can be dragged to the row axis. A value of on, 1, or true indicates the field can be dragged to the row axis. A value of off, 0, or false indicates the field cannot be dragged to the row axis. The possible values for this attribute are defined by the XML Schema boolean datatype.
hiddenLevel	Specifies a boolean value that indicates whether there is a hidden level in the PivotTable.

Attributes	Description
(Hidden Level)	This attribute applies to OLAP-based PivotTables only. A value of on, 1, or true indicates the OLAP PivotTable contains a hidden level. A value of off, 0, or false indicates the OLAP PivotTable does not contain any hidden levels. The possible values for this attribute are defined by the XML Schema boolean datatype.
hideNewItems (Hide New Items)	Specifies a boolean value that indicates whether new items that appear after a refresh should be hidden by default. A value of on, 1, or true indicates that items that appear after a refresh should be hidden by default. A value of off, 0, or false indicates that items that appear after a refresh should be shown by default. The possible values for this attribute are defined by the XML Schema boolean datatype.
includeNewItemsInFilter (Inclusive Manual Filter)	Specifies a boolean value that indicates whether manual filter is in inclusive mode. A value of on, 1, or true indicates the manual filter is inclusive. A value of off, 0, or false indicates the manual filter is not inclusive. The possible values for this attribute are defined by the XML Schema boolean datatype.
insertBlankRow (Insert Blank Row)	Specifies a boolean value that indicates whether to insert a blank row after each item. A value of on, 1, or true indicates that a blank row will be inserted after each item. A value of off, 0, or false indicates no additional rows will be inserted after each item. The possible values for this attribute are defined by the XML Schema boolean datatype.
insertPageBreak (Insert Item Page Break)	Specifies a boolean value that indicates whether to insert a page break after each item. A value of on, 1, or true indicates that a page break will be inserted after each item. A value of off, 0, or false indicates no page breaks will be inserted after items. The possible values for this attribute are defined by the XML Schema boolean datatype.
itemPageCount (Items Per Page Count)	Specifies the number of items showed per page in the PivotTable. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
maxSubtotal (Max	Specifies a boolean value that indicates whether to apply the 'max' aggregation function

Attributes	Description
Subtotal)	in the subtotal of this field. A value of on, 1, or true indicates that the 'max' aggregation function is applied in the subtotal for this field. A value of off, 0, or false indicates another aggregation function is applied in the subtotal for this field. The possible values for this attribute are defined by the XML Schema boolean datatype.
measureFilter (Measure Filter)	Specifies a boolean value that indicates whether field has a measure based filter. A value of on, 1, or true indicates the field has a measure-based filter. A value of off, 0, or false indicates does not have a measure-based filter. The possible values for this attribute are defined by the XML Schema boolean datatype.
minSubtotal (Min Subtotal)	Specifies a boolean value that indicates whether to apply the 'min' aggregation function in the subtotal of this field. A value of on, 1, or true indicates that the 'min' aggregation function is applied in the subtotal for this field. A value of off, 0, or false indicates another aggregation function is applied in the subtotal for this field. The possible values for this attribute are defined by the XML Schema boolean datatype.
multipleItemSelectionAllowed (Multiple Field Filters)	Specifies a boolean value that indicates whether the field can have multiple items selected in the page field. A value of on, 1, or true indicates the PivotTable can have multiple items selected in the page field. A value of off, 0, or false indicates the PivotTable cannot have multiple items selected in the page field. This attribute depends on the application support for selecting multiple items in page fields. The possible values for this attribute are defined by the XML Schema boolean datatype.
name (Field Name)	Specifies the name of the field. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
nonAutoSortDefault (Auto Sort)	Specifies a boolean value that indicates whether sort operation that will be applied to field should be AutoSort operation or simple data sort operation.

Attributes	Description
	A value of on, 1, or true indicates that an AutoSort operation will be applied to the field. A value of off, 0, or false indicates a simple data sort operation will be applied to the field. The possible values for this attribute are defined by the XML Schema boolean datatype.
numFmtId (Number Format Id)	Specifies the identifier of the number format to apply to this field. Number formats are written to the styles part. See the Styles section (§3.8) for more information on number formats. Note: Formatting information provided by cell table and by PivotTable need not agree. If the two formats differ, the cell-level formatting takes precedence. If you change the layout the PivotTable, the PivotTable formatting will then take precedence. The possible values for this attribute are defined by the ST_NumFmtId simple type (§3.18.49).
outline (Outline Items)	Specifies a boolean value that indicates whether the items in this field should be shown in Outline form. A value of on, 1, or true indicates the items in this field will be shown in Outline form. A value of off, 0, or false indicates the items in this field will not be shown in Outline form. This attribute depends on the application support for displaying items in Outline form. The possible values for this attribute are defined by the XML Schema boolean datatype.
productSubtotal (Product Subtotal)	Specifies a boolean value that indicates whether to apply 'product' aggregation function in the subtotal of this field. A value of on, 1, or true indicates that the 'product' aggregation function is applied in the subtotal for this field. A value of off, 0, or false indicates another aggregation function is applied in the subtotal for this field. The possible values for this attribute are defined by the XML Schema boolean datatype.
rankBy (Auto Show Rank By)	Specifies the index of the data field by which AutoShow will rank. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
serverField (Server-based Page Field)	Specifies a boolean value that indicates whether this is a server-based page field. A value of on, 1, or true indicates this is a server-based page field.

Attributes	Description
	A value of off, 0, or false indicates this is a local page field. The possible values for this attribute are defined by the XML Schema boolean datatype.
showAll (Show All Items)	Specifies a boolean value that indicates whether to show all items for this field. A value of on, 1, or true indicates that all items be shown. A value of off, 0, or false indicates items be shown according to user specified criteria. The possible values for this attribute are defined by the XML Schema boolean datatype.
showDropDowns (Show PivotField Header Drop Downs)	Specifies a boolean value that indicates whether to hide drop down buttons on PivotField headers. This attribute depends on the application implementation for filtering in the user interface. A value of on, 1, or true indicates the application will display some mechanism for selecting and applying filters--for example, a dropdown menu--in the user interface. A value of off, 0, or false indicates for mechanism for applying a filter will be displayed in the user interface. The possible values for this attribute are defined by the XML Schema boolean datatype.
showPropAsCaption (Show As Caption)	Specifies a boolean value that indicates whether to show the property as a member caption. A value of on, 1, or true indicates the property will be shown as a member caption. A value of off, 0, or false indicates the property will not be shown as a member caption. The possible values for this attribute are defined by the XML Schema boolean datatype.
showPropCell (Show Member Property in Cell)	Specifies a boolean value that indicates whether to show the member property value in a PivotTable cell. A value of on, 1, or true indicates the property value will be shown in a PivotTable cell. A value of off, 0, or false indicates the property value will not be shown in a PivotTable cell. The possible values for this attribute are defined by the XML Schema boolean datatype.
showPropTip (Show Member Property ToolTip)	Specifies a boolean value that indicates whether to show the member property value in a tooltip on the appropriate PivotTable cells. A value of on, 1, or true indicates the property value will be shown in a tooltip in the user interface.

Attributes	Description
	A value of off, 0, or false indicates the property will not be shown in a tooltip. This attribute depends on whether the application employs tooltips or similar mechanism in the user interface. The possible values for this attribute are defined by the XML Schema boolean datatype.
sortType (Auto Sort Type)	Specifies the type of sort that is applied to this field. The possible values for this attribute are defined by the ST_FieldSortType simple type (§3.18.29).
stdDevPSubtotal (StdDevP Subtotal)	Specifies a boolean value that indicates whether to apply the 'stdDevP' aggregation function in the subtotal of this field. A value of on, 1, or true indicates that the 'stdDevP' aggregation function is applied in the subtotal for this field. A value of off, 0, or false indicates another aggregation function is applied in the subtotal for this field. The possible values for this attribute are defined by the XML Schema boolean datatype.
stdDevSubtotal (StdDev Subtotal)	Specifies a boolean value that indicates whether to use 'stdDev' in the subtotal of this field. A value of on, 1, or true indicates that the 'stdDev' aggregation function is applied in the subtotal for this field. A value of off, 0, or false indicates another aggregation function is applied in the subtotal for this field. The possible values for this attribute are defined by the XML Schema boolean datatype.
subtotalCaption (Custom Subtotal Caption)	Specifies the custom text that is displayed for the subtotals label. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
subtotalTop (Subtotals At Top)	Specifies a boolean value that indicates whether to display subtotals at the top of the group. Applies only when `Outline` its true. A value of on, 1, or true indicates a subtotal will be display at the top of the group. A value of off, 0, or false indicates subtotal will not be displayed at the top of the group. The possible values for this attribute are defined by the XML Schema boolean datatype.
sumSubtotal (Sum Subtotal)	Specifies a boolean value that indicates whether apply the 'sum' aggregation function in the subtotal of this field.

Attributes	Description
	A value of on, 1, or true indicates the 'sum' aggregation function will be applied in the subtotal of this field. A value of off, 0, or false indicates another aggregation function will be applied in the subtotal of this field. The possible values for this attribute are defined by the XML Schema boolean datatype.
topAutoShow (Top Auto Show)	Specifies a boolean value that indicates whether an AutoShow filter applied to this field is set to show the top ranked values. A value of on, 1, or true indicates whether an AutoShow filter will show top values for this field. A value of off, 0, or false indicates bottom ranked values will be shown. The possible values for this attribute are defined by the XML Schema boolean datatype.
uniqueMemberProperty (Unique Member Property)	Specifies the unique name of the member property to be used as a caption for the field and field items. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
varPSubtotal (VarP Subtotal)	Specifies a boolean value that indicates whether to apply the 'varP' aggregation function in the subtotal of this field. A value of on, 1, or true indicates the 'varP' aggregation function will be applied in the subtotal of this field. A value of off, 0, or false indicates another aggregation function will be applied in the subtotal of this field. The possible values for this attribute are defined by the XML Schema boolean datatype.
varSubtotal (Variance Subtotal)	Specifies a boolean value that indicates whether to apply the 'variance' aggregation function in the subtotal of this field. A value of on, 1, or true indicates the 'variance' aggregation function will be applied in the subtotal of this field. A value of off, 0, or false indicates another aggregation function will be applied in the subtotal of this field. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PivotField">
   <sequence>
      <element name="items" minOccurs="0" type="CT_Items"/>
      <element name="autoSortScope" minOccurs="0" type="CT_AutoSortScope"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="name" type="ST_Xstring"/>
   <attribute name="axis" use="optional" type="ST_Axis"/>
   <attribute name="dataField" type="xsd:boolean" default="false"/>
   <attribute name="subtotalCaption" type="ST_Xstring"/>
   <attribute name="showDropDowns" type="xsd:boolean" default="true"/>
   <attribute name="hiddenLevel" type="xsd:boolean" default="false"/>
   <attribute name="uniqueMemberProperty" type="ST_Xstring"/>
   <attribute name="compact" type="xsd:boolean" default="true"/>
   <attribute name="allDrilled" type="xsd:boolean" default="false"/>
   <attribute name="numFmtId" type="ST_NumFmtId" use="optional"/>
   <attribute name="outline" type="xsd:boolean" default="true"/>
   <attribute name="subtotalTop" type="xsd:boolean" default="true"/>
   <attribute name="dragToRow" type="xsd:boolean" default="true"/>
   <attribute name="dragToCol" type="xsd:boolean" default="true"/>
   <attribute name="multipleItemSelectionAllowed" type="xsd:boolean" default="false"/>
   <attribute name="dragToPage" type="xsd:boolean" default="true"/>
   <attribute name="dragToData" type="xsd:boolean" default="true"/>
   <attribute name="dragOff" type="xsd:boolean" default="true"/>
   <attribute name="showAll" type="xsd:boolean" default="true"/>
   <attribute name="insertBlankRow" type="xsd:boolean" default="false"/>
   <attribute name="serverField" type="xsd:boolean" default="false"/>
   <attribute name="insertPageBreak" type="xsd:boolean" default="false"/>
   <attribute name="autoShow" type="xsd:boolean" default="false"/>
   <attribute name="topAutoShow" type="xsd:boolean" default="true"/>
   <attribute name="hideNewItems" type="xsd:boolean" default="false"/>
   <attribute name="measureFilter" type="xsd:boolean" default="false"/>
   <attribute name="includeNewItemsInFilter" type="xsd:boolean" default="false"/>
   <attribute name="itemPageCount" type="xsd:unsignedInt" default="10"/>
   <attribute name="sortType" type="ST_FieldSortType" default="manual"/>
   <attribute name="dataSourceSort" type="xsd:boolean" use="optional"/>
   <attribute name="nonAutoSortDefault" type="xsd:boolean" default="false"/>
   <attribute name="rankBy" type="xsd:unsignedInt" use="optional"/>
   <attribute name="defaultSubtotal" type="xsd:boolean" default="true"/>
   <attribute name="sumSubtotal" type="xsd:boolean" default="false"/>
   <attribute name="countASubtotal" type="xsd:boolean" default="false"/>
   <attribute name="avgSubtotal" type="xsd:boolean" default="false"/>
   <attribute name="maxSubtotal" type="xsd:boolean" default="false"/>
   <attribute name="minSubtotal" type="xsd:boolean" default="false"/>
   <attribute name="productSubtotal" type="xsd:boolean" default="false"/>
   <attribute name="countSubtotal" type="xsd:boolean" default="false"/>
   <attribute name="stdDevSubtotal" type="xsd:boolean" default="false"/>
   <attribute name="stdDevPSubtotal" type="xsd:boolean" default="false"/>
   <attribute name="varSubtotal" type="xsd:boolean" default="false"/>
   <attribute name="varPSubtotal" type="xsd:boolean" default="false"/>
   <attribute name="showPropCell" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="showPropTip" type="xsd:boolean" use="optional" default="false"/>
```

```
    <attribute name="showPropAsCaption" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="defaultAttributeDrillState" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.10.1.70 pivotFields (PivotTable Fields)

Represents the collection of fields that appear on the PivotTable.

[*Example*:

```
<pivotFields count="28">
  <pivotField showAll="0" includeNewItemsInFilter="1">
    <items count="8">
      <item x="66"/>
      <item x="133"/>
      <item x="74"/>
      <item x="27"/>
      <item x="118"/>
      <item x="63"/>
      <item x="141"/>
      <item t="default"/>
    </items>
  </pivotField>
  <pivotField showAll="0" includeNewItemsInFilter="1"/>
  <pivotField axis="axisPage" showAll="0" includeNewItemsInFilter="1">
    <items count="2">
      <item x="0"/>
      <item t="default"/>
    </items>
  </pivotField>
<pivotField showAll="0" includeNewItemsInFilter="1"/>
```

end example]

Parent Elements
pivotTableDefinition (§3.10.1.73)

Child Elements	Subclause
pivotField (PivotTable Field)	§3.10.1.69

Attributes	Description
count (Field Count)	Specifies the number of fields in the PivotTable.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PivotFields">
   <sequence>
      <element name="pivotField" maxOccurs="unbounded" type="CT_PivotField"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.71 pivotHierarchies (PivotTable OLAP Hierarchies)

Represents the collection of OLAP hierarchies associated with the PivotTable.

[*Example*:

```
<sh:pivotHierarchies count="3">
  <sh:pivotHierarchy dragToRow="0" dragToCol="0" dragToPage="0" dragToData="1"/>
  <sh:pivotHierarchy dragToRow="0" dragToCol="0" dragToPage="0" dragToData="1"/>
  <sh:pivotHierarchy dragToRow="0" dragToCol="0" dragToPage="0" dragToData="1"/>
  <sh:pivotHierarchy dragToRow="0" dragToCol="0" dragToPage="0" dragToData="1"/>
</sh:pivotHierarchies>
```

end example]

Parent Elements
pivotTableDefinition (§3.10.1.73)

Child Elements	Subclause
pivotHierarchy (OLAP Hierarchy)	§3.10.1.72

Attributes	Description
count (OLAP Hierarchy Count)	Specifies the number of OLAP hierarchies in the collection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PivotHierarchies">
   <sequence>
      <element name="pivotHierarchy" maxOccurs="unbounded" type="CT_PivotHierarchy"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.72 pivotHierarchy (OLAP Hierarchy)

Represents a OLAP hierarchy associated with the PivotTable. A hierarchy is a hierarchical representation of related OLAP dimensions. Hierarchies are defined on the OLAP server and cannot be changed in the PivotTable. For example, hierarchy "A" might be defined as follows:

Level 1 Country/Region

Level 2 State\Provence

Level 3 City

[*Example*:

```
<sh:pivotHierarchy dragToRow="0" dragToCol="0" dragToPage="0" dragToData="1"/>
```

end example]

Parent Elements
pivotHierarchies (§3.10.1.71)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
members (Members)	§3.10.1.56
mps (OLAP Member Properties)	§3.10.1.59

Attributes	Description
caption (Hierarchy Caption)	Specifies the user defined caption of the hierarchy. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
dragOff (Drag Off)	Specifies a boolean value that indicates whether the user is allowed to remove this hierarchy from the PivotTable. A value of on, 1, or true indicates the user can remove this hierarchy from the PivotTable.

Attributes	Description
	A value of off, 0, or false indicates the user cannot remove the hierarchy from the PivotTable. The possible values for this attribute are defined by the XML Schema boolean datatype.
dragToCol (Drag To Column)	Specifies a boolean value that indicates whether the user is allowed to put this hierarchy into the column area of the PivotTable. A value of on, 1, or true indicates the user can put this hierarchy into the column area of the PivotTable. A value of off, 0, or false indicates the user cannot remove this hierarchy. The possible values for this attribute are defined by the XML Schema boolean datatype.
dragToData (Drag To Data)	Specifies a boolean value that indicates whether the user is allowed to put this hierarchy into the data area of the view. A value of on, 1, or true indicates A value of off, 0, or false indicates The possible values for this attribute are defined by the XML Schema boolean datatype.
dragToPage (Drag to Page)	Specifies a boolean value that indicates whether the user is allowed to put this hierarchy into the page area of the PivotTable. A value of on, 1, or true indicates the user can put this hierarchy into the page area of the PivotTable. A value of off, 0, or false indicates cannot put this hierarchy into the page area. The possible values for this attribute are defined by the XML Schema boolean datatype.
dragToRow (Drag To Row)	Specifies a boolean value that indicates whether the user is allowed to put this hierarchy into the row area of the PivotTable. A value of on, 1, or true indicates the user can put this hierarchy into the row area of the PivotTable. A value of off, 0, or false indicates cannot put this hierarchy into the row area. The possible values for this attribute are defined by the XML Schema boolean datatype.
includeNewItemsInFilter (Inclusive Manual Filter)	Specifies a boolean value that indicates whether the application will show only the items the user has selected. A value of on, 1, or true indicates the application will show only items the user has

Attributes	Description
	selected; all other items will be hidden. A value of off, 0, or false indicates the application will show all items. The possible values for this attribute are defined by the XML Schema boolean datatype.
multipleItemSelectionAllowed (Multiple Field Filters)	Specifies a boolean value that indicates whether the user can select multiple members when the hierarchy is in the page field area of the view. A value of on, 1, or true indicates the user can select multiple members. A value of off, 0, or false indicates the user cannot select multiple members. The possible values for this attribute are defined by the XML Schema boolean datatype.
outline (Outline New Levels)	Specifies a boolean value that indicates whether new levels added to the PivotTable are shown in Outline mode. A value of on, 1, or true indicates new levels are shown in Outline mode. A value of off, 0, or false indicates new items are not shown in Outline mode. The possible values for this attribute are defined by the XML Schema boolean datatype.
showInFieldList (Show In Field List)	Specifies a boolean value that indicates whether this hierarchy is omitted from the field list. This attribute depends on how the application exposes a list of fields for PivotTables in the user interface. A value of on, 1, or true indicates this hierarchy is show in the field list or similar mechanism in the user interface. A value of off, 0, or false indicates is not shown in the field list. The possible values for this attribute are defined by the XML Schema boolean datatype.
subtotalTop (New Levels Subtotals At Top)	Specifies a boolean value that indicates whether new levels added to the view will show their subtotals at the top. A value of on, 1, or true indicates new levels added to the view show their subtotals at the top. A value of off, 0, or false indicates new levels added to the view show their subtotals at the bottom. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PivotHierarchy">
   <sequence>
      <element name="mps" minOccurs="0" type="CT_MemberProperties"/>
      <element name="members" minOccurs="0" maxOccurs="unbounded" type="CT_Members"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="outline" type="xsd:boolean" default="false"/>
   <attribute name="multipleItemSelectionAllowed" type="xsd:boolean" default="false"/>
   <attribute name="subtotalTop" type="xsd:boolean" default="false"/>
   <attribute name="showInFieldList" type="xsd:boolean" default="true"/>
   <attribute name="dragToRow" type="xsd:boolean" default="true"/>
   <attribute name="dragToCol" type="xsd:boolean" default="true"/>
   <attribute name="dragToPage" type="xsd:boolean" default="true"/>
   <attribute name="dragToData" type="xsd:boolean" default="false"/>
   <attribute name="dragOff" type="xsd:boolean" default="true"/>
   <attribute name="includeNewItemsInFilter" type="xsd:boolean" default="false"/>
   <attribute name="caption" type="ST_Xstring" use="optional"/>
</complexType>
```

3.10.1.73 pivotTableDefinition (PivotTable Definition)

Represents the PivotTable root element for non-null PivotTables. There exists one pivotTableDefinition for each PivotTableDefinition part. The PivotTable definition encompasses the following information:

Structure

- Top-level attributes
- Location information
- Collection of fields
- Fields on the row axis
- Items on the row axis (specific values)
- Fields on the column axis
- Items on the column axis (specific values)
- Fields on the report filter region
- Fields in the values region
- Style information

Outline of the XML for a pivotTableDefinition

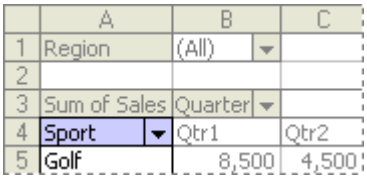
```
<pivotTableDefinition>
  <location/>
  <pivotFields/>
  <rowFields/>
  <rowItems/>
```

```
    <colFields/>
    <colItems/>
    <pageFields/>
    <dataFields/>
    <conditionalFormats/>
    <pivotTableStyleInfo/>
  </pivotTableDefinition>
```

Illustrations

Layout

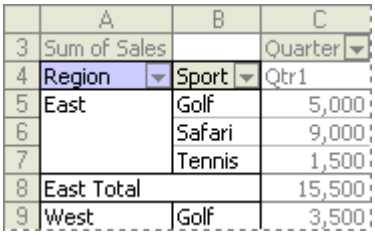
Row field



	A	B	C
1	Region	(All)	
2			
3	Sum of Sales	Quarter	
4	Sport	Qtr1	Qtr2
5	Golf	8,500	4,500

The blue field is a row field.

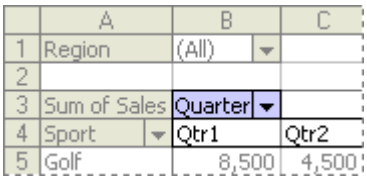
A PivotTable report that has more than one row field has one inner row field (Sport, in the example below), the one closest to the data area. Any other row fields are outer row fields (Region, in the example below). Items in the outermost row field are displayed only once, but items in the rest of the row fields are repeated as needed.



	A	B	C
3	Sum of Sales		Quarter
4	Region	Sport	Qtr1
5	East	Golf	5,000
6		Safari	9,000
7		Tennis	1,500
8	East Total		15,500
9	West	Golf	3,500

Region is an outer row field; Sport is an inner row field.

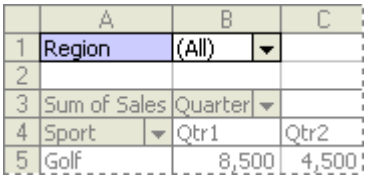
Column field



	A	B	C
1	Region	(All)	
2			
3	Sum of Sales	Quarter	
4	Sport	Qtr1	Qtr2
5	Golf	8,500	4,500

The blue field is a column field.

Page field

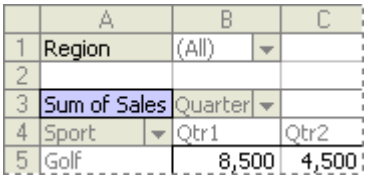


	A	B	C
1	Region	(All)	
2			
3	Sum of Sales	Quarter	
4	Sport	Qtr1	Qtr2
5	Golf	8,500	4,500

The blue field is a page field.

Page fields allow you to filter the entire PivotTable report to display data for a single item or all the items.

Data field



	A	B	C
1	Region	(All)	
2			
3	Sum of Sales	Quarter	
4	Sport	Qtr1	Qtr2
5	Golf	8,500	4,500

The blue field is a data field.

Data fields provide the data values to be summarized. Usually data fields contain numbers, which are combined with the Sum summary function, but data fields can also contain text, in which case the PivotTable report uses the Count summary function.

If a report has more than one data field, a single field button named Data appears in the report for access to all of the data fields.

[*Example*:

```
<?xml …?>
<sh:pivotTableDefinition xmlns:sh="…" name="PivotTable1" cacheId="3"
  applyNumberFormats="0" applyBorderFormats="0" applyFontFormats="0"
  applyPatternFormats="0" applyAlignmentFormats="0" applyWidthHeightFormats="1"
  dataCaption="Values" updatedVersion="3" minRefreshableVersion="3"
  showCalcMbrs="0" useAutoFormatting="1" colGrandTotals="0" itemPrintTitles="1"
  createdVersion="3" indent="0" outline="1" outlineData="1"
  multipleFieldFilters="0" fieldListSortAscending="1">
    <sh:location ref="B5:H49" firstHeaderRow="1" firstDataRow="4"
      firstDataCol="1" rowPageCount="2" colPageCount="1"/>
```

```
<sh:pivotFields count="28">
  <sh:pivotField axis="axisPage" showAll="0" includeNewItemsInFilter="1">
    <sh:items count="5">
      <sh:item x="1"/>
      <sh:item x="2"/>
      <sh:item x="3"/>
      <sh:item x="4"/>
      <sh:item x="5"/>
    </sh:items>
  </sh:pivotField>
  <sh:pivotField compact="0" showAll="0" includeNewItemsInFilter="1"/>
  <sh:pivotField showAll="0" includeNewItemsInFilter="1"/>
  <sh:pivotField axis="axisCol" showAll="0" includeNewItemsInFilter="1">
    <sh:items count="5">
      <sh:item x="0"/>
        <sh:item h="1" x="1"/>
        <sh:item h="1" x="2"/>
        <sh:item h="1" x="3"/>
        <sh:item t="default"/>
      </sh:items>
    </sh:pivotField>
    <sh:pivotField axis="axisCol" showAll="0" includeNewItemsInFilter="1">
      <sh:items count="5">
        <sh:item x="2"/>
        <sh:item x="3"/>
        <sh:item x="0"/>
        <sh:item x="1"/>
        <sh:item t="default"/>
      </sh:items>
    </sh:pivotField>
    <sh:pivotField showAll="0" includeNewItemsInFilter="1"/>
    <sh:pivotField dataField="1" showAll="0" includeNewItemsInFilter="1"/>
    <sh:pivotField showAll="0" includeNewItemsInFilter="1"/>
</sh:pivotFields>
<sh:rowFields count="2">
  <sh:field x="2"/>
  <sh:field x="5"/>
</sh:rowFields>
<sh:rowItems count="3">
  <sh:i r="1">
    <sh:x v="236"/>
  </sh:i>
```

```
    <sh:i r="1">
      <sh:x v="232"/>
     </sh:i>
     <sh:i t="grand">
       <sh:x/>
     </sh:i>
  </sh:rowItems>
  <sh:colFields count="3">
    <sh:field x="-2"/>
    <sh:field x="14"/>
    <sh:field x="15"/>
  </sh:colFields>
  <sh:colItems count="3">
    <sh:i>
     <sh:x/>
     <sh:x/>
     <sh:x v="2"/>
   </sh:i>
   <sh:i r="2">
     <sh:x v="3"/>
   </sh:i>
   <sh:i t="default" r="1">
     <sh:x/>
   </sh:i>
</sh:colItems>
<sh:pageFields count="2">
  <sh:pageField fld="0" hier="0"/>
  <sh:pageField fld="7" hier="0"/>
</sh:pageFields>
<sh:dataFields count="2">
  <sh:dataField name="Sum of Sales Amount" fld="25" baseField="0"
    baseItem="0"/>
  <sh:dataField name="Sum of Tax Amount" fld="26" baseField="0" baseItem="0"/>
</sh:dataFields>
<sh:conditionalFormats count="1">
  <sh:conditionalFormat priority="1">
    <sh:pivotAreas count="1">
      <sh:pivotArea type="data" collapsedLevelsAreSubtotals="1">
        <sh:references count="2">
          <sh:reference field="14" count="1" selected="0">
            <sh:x v="0"/>
          </sh:reference>
```

```
            <sh:reference field="15" count="2" selected="0">
              <sh:x v="2"/>
              <sh:x v="3"/>
            </sh:reference>
          </sh:references>
        </sh:pivotArea>
      </sh:pivotAreas>
    </sh:conditionalFormat>
  </sh:conditionalFormats>
  <sh:pivotTableStyleInfo name="PivotStyleDark8" showRowHeaders="1"
    showColHeaders="1" showRowStripes="0" showColStripes="0"
    showLastColumn="1"/>
</sh:pivotTableDefinition>
```

end example]

Parent Elements
Root element of SpreadsheetML Pivot Table part

Child Elements	Subclause
chartFormats (PivotChart Formats)	§3.10.1.13
colFields (Column Fields)	§3.10.1.14
colHierarchiesUsage (Column OLAP Hierarchy References)	§3.10.1.15
colItems (Column Items)	§3.10.1.17
conditionalFormats (Conditional Formats)	§3.10.1.19
dataFields (Data Fields)	§3.10.1.23
extLst (Future Feature Data Storage Area)	§3.2.10
filters (Filters)	§3.10.1.34
formats (PivotTable Formats)	§3.10.1.36
location (PivotTable Location)	§3.10.1.49
pageFields (Page Field Items)	§3.10.1.63
pivotFields (PivotTable Fields)	§3.10.1.70
pivotHierarchies (PivotTable OLAP Hierarchies)	§3.10.1.71
pivotTableStyleInfo (PivotTable Style)	§3.10.1.74
rowFields (Row Fields)	§3.10.1.81
rowHierarchiesUsage (Row OLAP Hierarchy References)	§3.10.1.82
rowItems (Row Items)	§3.10.1.84

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>applyAlignmentFormats (Apply Alignment Formats)</td><td>If <code>true</code> apply legacy table autoformat alignment properties.

The possible values for this attribute are defined by the XML Schema boolean datatype.</td></tr>
<tr><td>applyBorderFormats (Apply Border Formats)</td><td>If <code>true</code> apply legacy table autoformat border properties.

The possible values for this attribute are defined by the XML Schema boolean datatype.</td></tr>
<tr><td>applyFontFormats (Apply Font Formats)</td><td>If <code>true</code> apply legacy table autoformat font properties.

The possible values for this attribute are defined by the XML Schema boolean datatype.</td></tr>
<tr><td>applyNumberFormats (Apply Number Formats)</td><td>If <code>true</code> apply legacy table autoformat number format properties.

The possible values for this attribute are defined by the XML Schema boolean datatype.</td></tr>
<tr><td>applyPatternFormats (Apply Pattern Formats)</td><td>If <code>true</code> apply legacy table autoformat pattern properties.

The possible values for this attribute are defined by the XML Schema boolean datatype.</td></tr>
<tr><td>applyWidthHeightFormats (Apply Width / Height Formats)</td><td>If <code>true</code> apply legacy table autoformat width/height properties.

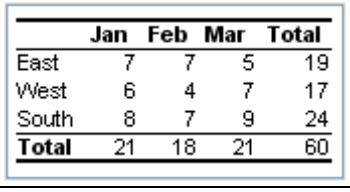
The possible values for this attribute are defined by the XML Schema boolean datatype.</td></tr>
<tr><td>asteriskTotals (Asterisk Totals)</td><td>Specifies a boolean value that indicates whether an asterisks should be displayed in subtotals and totals when visual totals are not used in OLAP -based PivotTables.

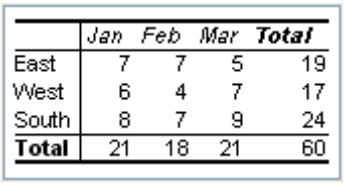
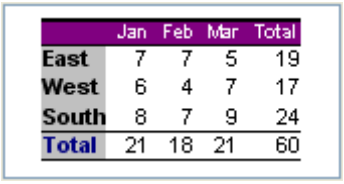
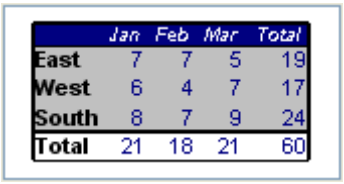
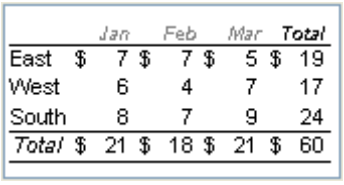
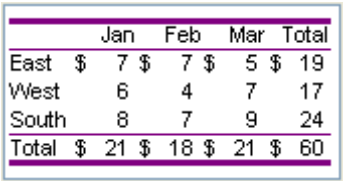
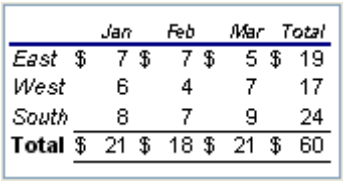
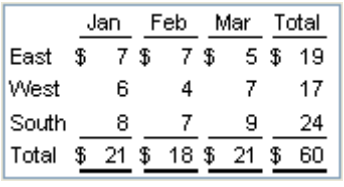
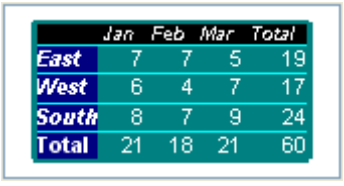
A value of on, 1, or true indicates an asterisks will be displayed in subtotals and totals for OLAP PivotTables when visual tools are not available.

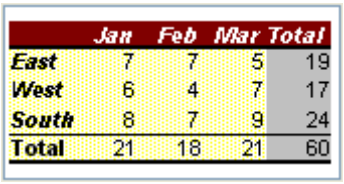
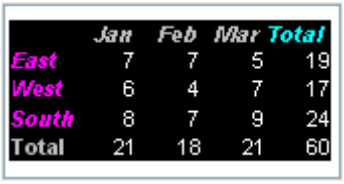
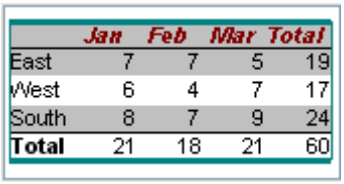
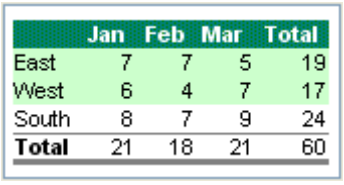
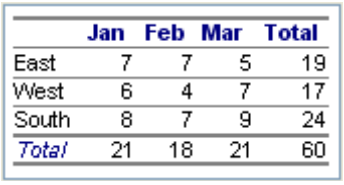
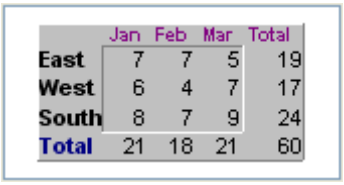
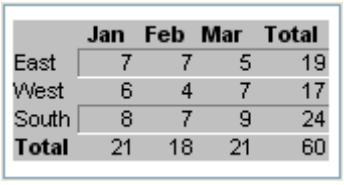
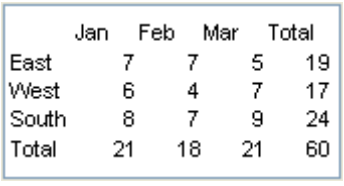
A value of off, 0, or false indicates an asterisk will not be displayed. This attribute depends on the implementation and availability of visual tools in the application user interface.

The possible values for this attribute are defined by the XML Schema boolean datatype.</td></tr>
<tr><td>autoFormatId (Auto Format Id)</td><td>Identifies which legacy table autoformat to apply.

Here are representations of the supported table autoformats:

<table>
<tr><th>autoFormatId</th><th>Description</th></tr>
<tr><td>0</td><td> 
</td></tr>
</table>
</td></tr>
</table>

Attributes	Description
	1 
	2 
	3 
	4 
	5 
	6 
	7 
	8 

Attributes	Description
	9 
	10 
	11 
	12 
	13 
	14 
	15 
	16 

The possible values for this attribute are defined by the XML Schema unsignedInt

Attributes	Description
	datatype.
cacheId (PivotCache Definition Id)	Specifies the identifier of the related PivotCache definition. This Id is listed in the pivotCaches collection in the workbook part. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
chartFormat (Chart Format Id)	Specifies the next chart formatting identifier to use on the PivotTable. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
colGrandTotals (Grand Totals On Columns)	Specifies a boolean value that indicates whether grand totals should be displayed for the PivotTable columns. A value of on, 1, or true indicates grand totals should be displayed. A value of off, 0, or false indicates grand totals should not be displayed for PivotTable columns. The possible values for this attribute are defined by the XML Schema boolean datatype.
colHeaderCaption (Column Header Caption)	Specifies the string to be displayed in column header in compact mode. This attribute depends on whether the application implements a compact mode for displaying PivotTables in the user interface. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
compact (Compact New Fields)	Specifies a boolean value that indicates whether new fields should have their `compact` flag set to true. A value of on, 1, or true indicates new fields should default to `compact` mode equal to true. A value of off, 0, or false indicates new fields should default to `compact` mode equal to false. This attribute depends on whether the application implements a compact mode in the user interface. The possible values for this attribute are defined by the XML Schema boolean datatype.
compactData (Compact Data)	Specifies a boolean value that indicates whether the field next to the data field in the PivotTable should be displayed in the same column of the spreadsheet The possible values for this attribute are defined by the XML Schema boolean datatype.
createdVersion (PivotCache Created Version)	Specifies the version of the application that created the cache. This attribute is application-dependent. The possible values for this attribute are defined by the XML Schema unsignedByte

Attributes	Description
	datatype.
customListSort (Custom List AutoSort)	Specifies a boolean value that indicates whether the "custom lists" option is offered when sorting this PivotTable. A value of on, 1, or true indicates custom lists are offered when sorting this PivotTable. A value of off, 0, or false indicates custom lists are not offered. This attribute depends on the implementation of sorting features in the application. The possible values for this attribute are defined by the XML Schema boolean datatype.
dataCaption (Data Field Header Name)	Specifies the name of the value area field header in the PivotTable. This caption is shown when the PivotTable when two or more fields are in the values area. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
dataOnRows (Data On Rows)	Specifies a boolean value that indicates the default orientation for fields in the data region. A value of on, 1, or true indicates vertical orientation. A value of off, 0, or false indicates horizontal orientation. The possible values for this attribute are defined by the XML Schema boolean datatype.
dataPosition (Default Data Field Position)	Specifies the default position for the data field in the PivotTable. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
disableFieldList (Disable Field List)	Specifies a boolean value that indicates whether to disable the PivotTable field list. A value of on, 1, or true indicates the field list, or similar mechanism for selecting fields in the user interface, is disabled. A value of off, 0, or false indicates the field list is enabled. The possible values for this attribute are defined by the XML Schema boolean datatype.
editData (Allow Edit Data)	Specifies a boolean value that indicates whether the user is allowed to edit the cells in the data area of the PivotTable. A value of on, 1, or true indicates the user can edit values in the data area. A value of off, 0, or false indicates the cells in the data area are not editable. The possible values for this attribute are defined by the XML Schema boolean datatype.
enableDrill (Enable	Specifies a boolean value that indicates whether the user is prevented from drilling down

Attributes	Description
Drill Down)	on a PivotItem or aggregate value. A value of on, 1, or true indicates the user can drill down on a pivot item or aggregate value. A value of off, 0, or false indicates the user is prevented from drilling down pivot item. The possible values for this attribute are defined by the XML Schema boolean datatype.
enableFieldProperties (Enable Field Properties)	Specifies a boolean value that indicates whether the user is prevented from displaying PivotField properties. A value of on, 1, or true indicates the user can display pivot field properties. A value of off, 0, or false indicates the user cannot display pivot field properties. This attribute depends on how pivot field properties are exposed in the application user interface. The possible values for this attribute are defined by the XML Schema boolean datatype.
enableWizard (Enable PivotTable Wizard)	Specifies a boolean value that indicates whether the user is prevented from displaying the PivotTable wizard. A value of on, 1, or true indicates the user may display the PivotTable wizard. A value of off, 0, or false indicates the user may not display the PivotTable wizard. This attribute depends on whether the application exposes a wizard or similar mechanism for creating and working with PivotTables in the user interface. The possible values for this attribute are defined by the XML Schema boolean datatype.
errorCaption (Error Caption)	Specifies the string to be displayed in cells that contain errors. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
fieldListSortAscending (Default Sort Order)	Specifies a boolean value that indicates whether fields in the PivotTable are sorted in non-default order in the field list. A value of on, 1, or true indicates fields for the PivotTable are sorted in the field list. The sort order from the data source is applied for range-based PivotTables. Alphabetical sorting is applied for external data PivotTables. A value of off, 0, or false indicates fields in the field list are not sorted. The possible values for this attribute are defined by the XML Schema boolean datatype.
fieldPrintTitles (Field Print Titles)	Specifies a boolean value that indicates whether the row and column titles from the PivotTable should be printed.

Attributes	Description
	A value of on, 1, or true indicates row and column titles should be printed. A value of off, 0, or false indicates row and column titles should not be printed. The possible values for this attribute are defined by the XML Schema boolean datatype.
grandTotalCaption (Grand Totals Caption)	Specifies the string to be displayed for grand totals. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
gridDropZones (Enable Drop Zones)	Specifies a boolean value that indicates whether the in-grid drop zones should be enabled. A value of on, 1, or true indicates in-grid drop zones should be enabled. A value of off, 0, or false indicates in-grid drop zones should be disabled. This attribute depends on how the application implements drop zones in the user interface. The possible values for this attribute are defined by the XML Schema boolean datatype.
immersive (Stop Immersive UI)	Specifies a boolean value that indicates whether PivotTable immersive experience user interface should be turned off. A value of on, 1, or true indicates the PivotTable immersive experience should be turned off for this PivotTable. A value of off, 0, or false indicates the immersive experience should be left on. This attribute depends on whether the application implements an immersive experience in the user interface. The possible values for this attribute are defined by the XML Schema boolean datatype.
indent (Indentation for Compact Axis)	Specifies the indentation increment for compact axis and can be used to set the Report Layout to Compact Form. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
itemPrintTitles (Item Print Titles)	Specifies a boolean value that indicates whether PivotItem names should be repeated at the top of each printed page. A value of on, 1, or true indicates pivot items names should be repeated at the top of each page. A value of off, 0, or false indicates should not be repeated. The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description
mdxSubqueries (MDX Subqueries Supported)	Specifies a boolean value that indicates whether MDX sub-queries are supported by OLAP data provider for this PivotTable. A value of on, 1, or true indicates MDX sub-queries are supported by the OLAP data provider. A value of off, 0, or false indicates MDX sub-queries are not supported. The possible values for this attribute are defined by the XML Schema boolean datatype.
mergeItem (Merge Titles)	Specifies a boolean value that indicates whether row or column titles that span multiple cells should be merged into a single cell. A value of on, 1, or true indicates that titles that span multiple cells will be merged into a single cell. A value of off, 0, or false indicates titles are not merged. The possible values for this attribute are defined by the XML Schema boolean datatype.
minRefreshableVersion (Minimum Refreshable Version)	Specifies the minimum version of the application required to update this PivotTable view. This attribute is application-dependent. The possible values for this attribute are defined by the XML Schema unsignedByte datatype.
missingCaption (Caption for Missing Values)	Specifies the string to be displayed in cells with no value The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
multipleFieldFilters (Multiple Field Filters)	Specifies a boolean value that indicates whether the fields of a PivotTable can have multiple filters set on them. A value of on, 1, or true indicates the fields of a PivotTable can have multiple filters. A value of off, 0, or false indicates the fields of a PivotTable can only have a simple filter. The possible values for this attribute are defined by the XML Schema boolean datatype.
name (Name)	Specifies the PivotTable name. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
outline (Outline New Fields)	Specifies a boolean value that indicates whether new fields should have their `outline` flag set to true. A value of on, 1, or true indicates new fields will be created with `outline` equal to true.

Attributes	Description
	A value of off, 0, or false indicates new fields will be created with `outline` equal to false. The possible values for this attribute are defined by the XML Schema boolean datatype.
outlineData (Outline Data Fields)	Specifies a boolean value that indicates whether data fields in the PivotTable should be displayed in outline form. A value of on, 1, or true indicates data fields will display in outline form. A value of off, 0, or false indicates data fields will not display in outline form. The possible values for this attribute are defined by the XML Schema boolean datatype.
pageOverThenDown (Page Over Then Down)	Specifies a boolean value that indicates how the page fields are laid out when there are multiple PivotFields in the page area. A value of on, 1, or true indicates the fields will display "Over, then down" A value of off, 0, or false indicates the fields will display "down, then Over" The possible values for this attribute are defined by the XML Schema boolean datatype.
pageStyle (Page Header Style Name)	Specifies the name of the style to apply to each of the field item headers in the page area of the PivotTable. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
pageWrap (Page Wrap)	Specifies the number of page fields to display before starting another row or column. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
pivotTableStyle (Table Style Name)	Specifies the name of the style to apply to the main table area of the PivotTable. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
preserveFormatting (Preserve Formatting)	Specifies a boolean value that indicates whether the formatting applied by the user to the PivotTable cells is discarded on refresh. A value of on, 1, or true indicates the formatting applied by the end user is discarded on refresh. A value of off, 0, or false indicates the end-user formatting is retained on refresh. The possible values for this attribute are defined by the XML Schema boolean datatype.
printDrill (Print Drill Indicators)	Specifies a boolean value that indicates whether drill indicators should be printed. A value of on, 1, or true indicates

Attributes	Description
	A value of off, 0, or false indicates The possible values for this attribute are defined by the XML Schema boolean datatype.
published (Data Fields Published)	Specifies a boolean value that indicates whether data fields in the PivotTable are published. A value of on, 1, or true indicates A value of off, 0, or false indicates The possible values for this attribute are defined by the XML Schema boolean datatype.
rowGrandTotals (Row Grand Totals)	Specifies a boolean value that indicates whether grand totals should be displayed for the PivotTable rows. The default value for this attribute is true. A value of on, 1, or true indicates grand totals will be displayed for the PivotTable rows. A value of off, 0, or false indicates grand totals will not be displayed. The possible values for this attribute are defined by the XML Schema boolean datatype.
rowHeaderCaption (Row Header Caption)	Specifies the string to be displayed in row header in `compact` mode. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
showCalcMbrs (Show Calculated Members)	Specifies a boolean value that indicates whether calculated members should be shown in the PivotTable view. This attribute applies to PivotTables from OLAP-sources only. A value of on, 1, or true indicates that calculated members should be shown. A value of off, 0, or false indicates calculated members should not be shown. The possible values for this attribute are defined by the XML Schema boolean datatype.
showDataDropDown (Show Drop Down)	Specifies a boolean value that indicates whether the drop-down lists for the fields in the PivotTable should be hidden. This attribute depends on whether the application implements drop down lists or similar mechanism in the user interface. A value of on, 1, or true indicates drop down lists will be displayed for fields. A value of off, 0, or false indicates drop down lists will not be displayed. The possible values for this attribute are defined by the XML Schema boolean datatype.
showDataTips (Show ToolTips on Data)	Specifies a boolean value that indicates whether tooltips should be displayed for PivotTable data cells.

Attributes	Description
	A value of on, 1, or true indicates tooltips will be displayed. A value of off, 0, or false indicates tooltips will not be displayed. This attribute depends on whether the application employs tooltips or similar mechanism in the user interface. The possible values for this attribute are defined by the XML Schema boolean datatype.
showDrill (Show Expand Collapse)	Specifies a boolean value that indicates whether drill indicators should be hidden. A value of on, 1, or true indicates drill indicators will be displayed. A value of off, 0, or false indicates drill indicators will not be displayed. The possible values for this attribute are defined by the XML Schema boolean datatype.
showDropZones (Show Drop Zones)	Specifies a boolean value that indicates whether the PivotTable should display large drop zones when there are no fields in the data region. A value of on, 1, or true indicates a large drop zone will be displayed. A value of off, 0, or false indicates a large drop zone will not be displayed. The possible values for this attribute are defined by the XML Schema boolean datatype.
showEmptyCol (Show Empty Column)	Specifies a boolean value that indicates whether to include empty columns in the table. A value of on, 1, or true indicates empty columns will be included in the PivotTable. A value of off, 0, or false indicates empty columns will be excluded. The possible values for this attribute are defined by the XML Schema boolean datatype.
showEmptyRow (Show Empty Row)	Specifies a boolean value that indicates whether to include empty rows in the table. A value of on, 1, or true indicates empty rows will be included in the PivotTable. A value of off, 0, or false indicates empty rows will be excluded. The possible values for this attribute are defined by the XML Schema boolean datatype.
showError (Show Error)	Specifies a boolean value that indicates whether to show error messages in cells. A value of on, 1, or true indicates error messages will be shown in cells. A value of off, 0, or false indicates error messages will be shown through another mechanism the application provides in the user interface. The possible values for this attribute are defined by the XML Schema boolean datatype.
showHeaders	Specifies a boolean value that indicates whether to suppress display of pivot field

Attributes	Description
(Show Field Headers)	headers. A value of on, 1, or true indicates field headers will be shown in the PivotTable. A value of off, 0, or false indicates field headers will be excluded. The possible values for this attribute are defined by the XML Schema boolean datatype.
showItems (Show Item Names)	Specifies a boolean value that indicates whether to display item names when adding a field onto a PivotTable that has no data fields. A value of on, 1, or true indicates item names will be displayed. A value of off, 0, or false indicates item names will not be displayed. The possible values for this attribute are defined by the XML Schema boolean datatype.
showMemberPropertyTips (Show Member Property ToolTips)	Specifies a boolean value that indicates whether member property information should be omitted from PivotTable tooltips. A value of on, 1, or true indicates member property information will be included. A value of off, 0, or false indicates member property information will be excluded. This attribute depends on whether the application employs tooltips or similar mechanism in the user interface. The possible values for this attribute are defined by the XML Schema boolean datatype.
showMissing (Show Missing)	Specifies a boolean value that indicates whether to show a message in cells with no value. A value of on, 1, or true indicates to show a message string in cells without values. A value of off, 0, or false indicates no message string will shown in cells without values. The possible values for this attribute are defined by the XML Schema boolean datatype.
showMultipleLabel (Show Multiple Labels)	Specifies a boolean value that indicates whether a page field with multiple selected items should display "(multiple items)" instead of "All". This attribute applies only to non-OLAP PivotTables. The messages displayed depend on the application implementation. A value of on, 1, or true indicates a different message string will be displayed for a page field with multiple items. A value of off, 0, or false indicates the same message string will be displayed for all page fields. The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description
subtotalHiddenItems (Subtotal Hidden Items)	Specifies a boolean value that indicates whether data for hidden pivotItems for PivotFields in the data area should be included in subtotals. A value of on, 1, or true indicates that data for hidden pivot items in the data area will be included in subtotals. A value of off, 0, or false indicates hidden pivot items will not be included in subtotals. The possible values for this attribute are defined by the XML Schema boolean datatype.
tag (PivotTable Custom String)	Specifies a user-defined string that is associated with this PivotTable. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
updatedVersion (PivotTable Last Updated Version)	Specifies the version of the application that last updated the PivotTable view. This attribute is application-dependent. The possible values for this attribute are defined by the XML Schema unsignedByte datatype.
useAutoFormatting (Auto Formatting)	Specifies a boolean value that indicates whether auto formatting has been applied to the PivotTable view. A value of on, 1, or true indicates A value of off, 0, or false indicates The possible values for this attribute are defined by the XML Schema boolean datatype.
vacatedStyle (Vacated Style)	Specifies the name of the style to apply to the cells left blank when a PivotTable shrinks during a refresh operation The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
visualTotals (Total Visual Data)	Specifies a boolean value that indicates whether totals should be based on visible data only. This attribute applies to OLAP PivotTables only. A value of on, 1, or true indicates subtotals will be computed on visible data only. A value of off, 0, or false indicates subtotals will be computed on all data. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_pivotTableDefinition">
    <sequence>
        <element name="location" type="CT_Location"/>
        <element name="pivotFields" type="CT_PivotFields" minOccurs="0"/>
        <element name="rowFields" type="CT_RowFields" minOccurs="0"/>
        <element name="rowItems" type="CT_rowItems" minOccurs="0"/>
        <element name="colFields" type="CT_ColFields" minOccurs="0"/>
        <element name="colItems" type="CT_colItems" minOccurs="0"/>
        <element name="pageFields" type="CT_PageFields" minOccurs="0"/>
        <element name="dataFields" type="CT_DataFields" minOccurs="0"/>
        <element name="formats" type="CT_Formats" minOccurs="0"/>
        <element name="conditionalFormats" type="CT_ConditionalFormats" minOccurs="0"/>
        <element name="chartFormats" type="CT_ChartFormats" minOccurs="0"/>
        <element name="pivotHierarchies" type="CT_PivotHierarchies" minOccurs="0"/>
        <element name="pivotTableStyleInfo" minOccurs="0" maxOccurs="1" type="CT_PivotTableStyle"/>
        <element name="filters" minOccurs="0" maxOccurs="1" type="CT_PivotFilters"/>
        <element name="rowHierarchiesUsage" type="CT_RowHierarchiesUsage" minOccurs="0"
          maxOccurs="1"/>
        <element name="colHierarchiesUsage" type="CT_ColHierarchiesUsage" minOccurs="0"
          maxOccurs="1"/>
        <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
    </sequence>
    <attribute name="name" use="required" type="ST_Xstring"/>
    <attribute name="cacheId" use="required" type="xsd:unsignedInt"/>
    <attribute name="dataOnRows" type="xsd:boolean" default="false"/>
    <attribute name="dataPosition" type="xsd:unsignedInt" use="optional"/>
    <attributeGroup ref="AG_AutoFormat"/>
    <attribute name="dataCaption" use="required" type="ST_Xstring"/>
    <attribute name="grandTotalCaption" type="ST_Xstring"/>
    <attribute name="errorCaption" type="ST_Xstring"/>
    <attribute name="showError" type="xsd:boolean" default="false"/>
    <attribute name="missingCaption" type="ST_Xstring"/>
    <attribute name="showMissing" type="xsd:boolean" default="true"/>
    <attribute name="pageStyle" type="ST_Xstring"/>
    <attribute name="pivotTableStyle" type="ST_Xstring"/>
    <attribute name="vacatedStyle" type="ST_Xstring"/>
    <attribute name="tag" type="ST_Xstring"/>
    <attribute name="updatedVersion" type="xsd:unsignedByte" default="0"/>
    <attribute name="minRefreshableVersion" type="xsd:unsignedByte" default="0"/>
    <attribute name="asteriskTotals" type="xsd:boolean" default="false"/>
    <attribute name="showItems" type="xsd:boolean" default="true"/>
    <attribute name="editData" type="xsd:boolean" default="false"/>
    <attribute name="disableFieldList" type="xsd:boolean" default="false"/>
    <attribute name="showCalcMbrs" type="xsd:boolean" default="true"/>
    <attribute name="visualTotals" type="xsd:boolean" default="true"/>
    <attribute name="showMultipleLabel" type="xsd:boolean" default="true"/>
    <attribute name="showDataDropDown" type="xsd:boolean" default="true"/>
    <attribute name="showDrill" type="xsd:boolean" default="true"/>
    <attribute name="printDrill" type="xsd:boolean" default="false"/>
    <attribute name="showMemberPropertyTips" type="xsd:boolean" default="true"/>
    <attribute name="showDataTips" type="xsd:boolean" default="true"/>
    <attribute name="enableWizard" type="xsd:boolean" default="true"/>
```

```
    <attribute name="enableDrill" type="xsd:boolean" default="true"/>
    <attribute name="enableFieldProperties" type="xsd:boolean" default="true"/>
    <attribute name="preserveFormatting" type="xsd:boolean" default="true"/>
    <attribute name="useAutoFormatting" type="xsd:boolean" default="false"/>
    <attribute name="pageWrap" type="xsd:unsignedInt" default="0"/>
    <attribute name="pageOverThenDown" type="xsd:boolean" default="false"/>
    <attribute name="subtotalHiddenItems" type="xsd:boolean" default="false"/>
    <attribute name="rowGrandTotals" type="xsd:boolean" default="true"/>
    <attribute name="colGrandTotals" type="xsd:boolean" default="true"/>
    <attribute name="fieldPrintTitles" type="xsd:boolean" default="false"/>
    <attribute name="itemPrintTitles" type="xsd:boolean" default="false"/>
    <attribute name="mergeItem" type="xsd:boolean" default="false"/>
    <attribute name="showDropZones" type="xsd:boolean" default="true"/>
    <attribute name="createdVersion" type="xsd:unsignedByte" default="0"/>
    <attribute name="indent" type="xsd:unsignedInt" default="1"/>
    <attribute name="showEmptyRow" type="xsd:boolean" default="false"/>
    <attribute name="showEmptyCol" type="xsd:boolean" default="false"/>
    <attribute name="showHeaders" type="xsd:boolean" default="true"/>
    <attribute name="compact" type="xsd:boolean" default="true"/>
    <attribute name="outline" type="xsd:boolean" default="false"/>
    <attribute name="outlineData" type="xsd:boolean" default="false"/>
    <attribute name="compactData" type="xsd:boolean" default="true"/>
    <attribute name="published" type="xsd:boolean" default="false"/>
    <attribute name="gridDropZones" type="xsd:boolean" default="false"/>
    <attribute name="immersive" type="xsd:boolean" default="true"/>
    <attribute name="multipleFieldFilters" type="xsd:boolean" default="true"/>
    <attribute name="chartFormat" type="xsd:unsignedInt" default="0"/>
    <attribute name="rowHeaderCaption" type="ST_Xstring"/>
    <attribute name="colHeaderCaption" type="ST_Xstring"/>
    <attribute name="fieldListSortAscending" type="xsd:boolean" default="false"/>
    <attribute name="mdxSubqueries" type="xsd:boolean" default="false"/>
    <attribute name="customListSort" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

3.10.1.74 pivotTableStyleInfo (PivotTable Style)

Represent information on style applied to the PivotTable.

[*Example*:

```
<sh:pivotTableStyleInfo name="PivotStyleLight16" showRowHeaders="1"
  showColHeaders="1" showRowStripes="0" showColStripes="0" showLastColumn="1"/>
```

end example]

Parent Elements
pivotTableDefinition (§3.10.1.73)

Attributes	Description

Attributes	Description
name (Table Style Name)	Specifies the name of the table style to use with this table. The possible values for this attribute are defined by the XML Schema string datatype.
showColHeaders (Show Table Style Column Header Formatting)	Specifies a boolean value that indicates whether to show column headers for the table. A value of on, 1, or true indicates column headers will be shown. A value of off, 0, or false indicates column headers will be omitted. 'True' if table style column header formatting should be displayed. The possible values for this attribute are defined by the XML Schema boolean datatype.
showColStripes (Show Column Stripes)	Specifies a boolean value that indicates whether to show column stripe formatting for the table. A value of on, 1, or true indicates column stripe formatting will be shown. A value of off, 0, or false indicates no column formatting will be shown. True if table style column stripe formatting should be displayed. The possible values for this attribute are defined by the XML Schema boolean datatype.
showLastColumn (Show Last Column)	Specifies a boolean value that indicates whether to show the last column. The possible values for this attribute are defined by the XML Schema boolean datatype.
showRowHeaders (Show Row Header Formatting)	Specifies a boolean value that indicates whether to show row headers for the table. A value of on, 1, or true indicates table style formatting will be displayed. A value of off, 0, or false indicates table style formatting will not be displayed. The possible values for this attribute are defined by the XML Schema boolean datatype.
showRowStripes (Show Row Stripes)	Specifies a boolean value that indicates whether to show row stripe formatting for the table. A value of on, 1, or true indicates row stripe formatting will be displayed. A value of off, 0, or false indicates no row formatting will be shown. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PivotTableStyle">
   <attribute name="name" type="xsd:string"/>
   <attribute name="showRowHeaders" type="xsd:boolean"/>
   <attribute name="showColHeaders" type="xsd:boolean"/>
   <attribute name="showRowStripes" type="xsd:boolean"/>
   <attribute name="showColStripes" type="xsd:boolean"/>
   <attribute name="showLastColumn" type="xsd:boolean" use="optional"/>
</complexType>
```

3.10.1.75 query (Query)

Represents an OLAP sheet data cached query.

Parent Elements
queryCache (§3.10.1.76)

Child Elements	Subclause
tpls (Tuples)	§3.10.1.93

Attributes	Description
mdx (MDX Query String)	Specifies the Multidimensional Expressions (MDX) query string. [*Note*: Data connectivity can use a number of different technologies. One example of potential values stored in this attribute can be found at: http://msdn2.microsoft.com/en-us/library/ms145595.aspx *end note*] See the MDX Language Reference for more information: The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Query">
   <sequence>
      <element name="tpls" minOccurs="0" type="CT_Tuples"/>
   </sequence>
   <attribute name="mdx" use="required" type="ST_Xstring"/>
</complexType>
```

3.10.1.76 queryCache (OLAP Query Cache)

Represents the cache of OLAP sheet data queries.

Parent Elements

Parent Elements
tupleCache (§3.10.1.94)

Child Elements	Subclause
query (Query)	§3.10.1.75

Attributes	Description
count (Cached Query Count)	Specifies the number of cached queries in the collection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_QueryCache">
   <sequence>
      <element name="query" maxOccurs="unbounded" type="CT_Query"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.77 r (PivotCache Record)

Represents a single record of data in the PivotCache.

[*Example*:

```
<r>
  <s v="3550"/>
  <s v="Road-150 Red, 62"/>
  <s v="This bike is ridden by race winners. Developed with the Adventure Works
    Cycles professional race team, it has a extremely light heat-treated
    aluminum frame, and steering that allows precision control."/>
  <s v="No Discount"/>
  <x v="0"/>
  <s v="Australian Dollar"/>
  <n v="1"/>
  <n v="3578.27"/>
  <n v="0"/>
  <n v="2171.2941999999998"/>
  <n v="3578.27"/>
  <n v="89.456800000000001"/>
</r>
```

end example]

Parent Elements
pivotCacheRecords (§3.10.1.68)

Child Elements	Subclause
b (Boolean)	§3.10.1.2
d (Date Time)	§3.10.1.21
e (Error Value)	§3.10.1.27
m (No Value)	§3.10.1.50
n (Numeric)	§3.10.1.60
s (Character Value)	§3.10.1.85
x (Shared Items Index)	§3.10.1.97

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Record">
   <choice maxOccurs="unbounded">
      <element name="m" type="CT_Missing"/>
      <element name="n" type="CT_Number"/>
      <element name="b" type="CT_Boolean"/>
      <element name="e" type="CT_Error"/>
      <element name="s" type="CT_String"/>
      <element name="d" type="CT_DateTime"/>
      <element name="x" type="CT_Index"/>
   </choice>
</complexType>
```

3.10.1.78 rangePr (Range Grouping Properties)

Represents the collection of range grouping properties.

[*Example*:

```
<rangePr groupBy="months" startDate="2002-01-01T00:00:00"
  endDate="2006-05-06T00:00:00"/>
```

end example]

Parent Elements
fieldGroup (§3.10.1.30)

Attributes	Description
autoEnd (Source Data Ending Range)	Specifies a boolean value that indicates whether the application will use the source data to set the ending range value.

Attributes	Description
	A value of on, 1, or true indicates the ending range value will be set from the source data. A value of off, 0, or false indicates ending range values will be set by the value specified in endDate or endNum. The possible values for this attribute are defined by the XML Schema boolean datatype.
autoStart (Source Data Set Beginning Range)	Specifies a boolean value that indicates whether we use source data to set the beginning range value. A value of on, 1, or true indicates the beginning range value will be set from the source data. A value of off, 0, or false indicates the beginning range value will be set from the value specified in startDate or startNum. The possible values for this attribute are defined by the XML Schema boolean datatype.
endDate (Date Grouping End Value)	Specifies the ending value for date grouping if autoEnd is false. The possible values for this attribute are defined by the XML Schema dateTime datatype.
endNum (Numeric Grouping End Value)	Specifies the ending value for numeric grouping if autoEnd is false. The possible values for this attribute are defined by the XML Schema double datatype.
groupBy (Group By)	Specifies the grouping. The possible values for this attribute are defined by the ST_GroupBy simple type (§3.18.39).
groupInterval (Grouping Interval)	Specifies the grouping interval for numeric range grouping. Specifies the number of days to group by in date range grouping. The possible values for this attribute are defined by the XML Schema double datatype.
startDate (Date Grouping Start Value)	Specifies the starting value for date grouping if autoStart is false. The possible values for this attribute are defined by the XML Schema dateTime datatype.
startNum (Numeric Grouping Start Value)	Specifies the starting value for numeric grouping if autoStart is false. The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RangePr">
    <attribute name="autoStart" type="xsd:boolean" default="true"/>
    <attribute name="autoEnd" type="xsd:boolean" default="true"/>
    <attribute name="groupBy" type="ST_GroupBy" default="range"/>
    <attribute name="startNum" type="xsd:double"/>
    <attribute name="endNum" type="xsd:double"/>
    <attribute name="startDate" type="xsd:dateTime"/>
    <attribute name="endDate" type="xsd:dateTime"/>
    <attribute name="groupInterval" type="xsd:double" default="1"/>
</complexType>
```

3.10.1.79 rangeSet (Range Set)

Represents a single range in the rangeSets collection. This complex type is intended to facilitate creating a PivotTable report by consolidating SpreadsheetML ranges that have similar categories of data to be summarized. The simplest layout for the data source is for each rangeSets of data to be in list-like format, with column labels in the first row, row labels in the first column, the rest of the rows having similar items in the same row and column, and no blank rows or columns within the range. A particular rangeSet can consist of a built-in named range that is provided by the application, a user defined named range, a range reference, or a reference to an external workbook.

When multiple ranges are consolidated using this functionality, up to 4 custom report filters (also known as page fields) can be created to help filter the PivotTable report, by specifically enabling one or more of the individual ranges to be selected in the report filter. For each custom page field created, a custom label can be specified and assigned to each range participating in the consolidation range, so that the PivotTable can be filtered by one or more of the ranges being summarized.

[*Example*: Consider a workbook with 6 worksheets. On Sheet1 we have:

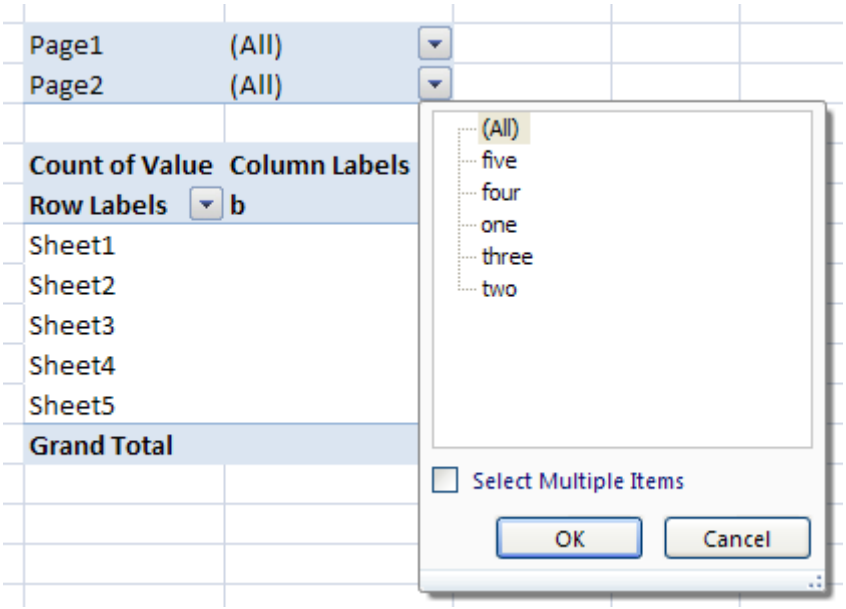
a	b
Sheet1	Sheet1
Sheet1	Sheet1

On Sheet2 we have:

a	b
Sheet2	Sheet2
Sheet2	Sheet2

... and so on up through Sheet5.

On Sheet6, we have the consolidated ranges being summarized by a PivotTable, and two page filters exist for the PivotTable.



Notice that for the second page filter, the items have been assigned a custom label, "one", "two", ..., "five", for each of Sheet1, Sheet2, ..., Sheet5 data sources, respectively. Similarly, the items have been assigned a custom label, "1", "2", ..., "5" for each of Sheet1, Sheet2, ..., Shet5 data sources, respectively.

The XML representing these custom page filters shall be like the following:

```
<cacheSource type="consolidation">
  <consolidation autoPage="0">
    <pages count="2">
      <page count="5">
        <pageItem name="1"/>
        <pageItem name="2"/>
        <pageItem name="3"/>
        <pageItem name="4"/>
        <pageItem name="5"/>
      </page>
      <page count="5">
        <pageItem name="one"/>
        <pageItem name="two"/>
        <pageItem name="three"/>
        <pageItem name="four"/>
        <pageItem name="five"/>
      </page>
    </pages>
```

```
      <rangeSets count="5">
        <rangeSet i1="0" i2="0" ref="A1:B3" sheet="Sheet1"/>
        <rangeSet i1="1" i2="1" ref="A1:B3" sheet="Sheet2"/>
        <rangeSet i1="2" i2="2" ref="A1:B3" sheet="Sheet3"/>
        <rangeSet i1="3" i2="3" ref="A1:B3" sheet="Sheet4"/>
        <rangeSet i1="4" i2="4" ref="A1:B3" sheet="Sheet5"/>
      </rangeSets>
    </consolidation>
  </cacheSource>
```

end example]

[*Note*: Attributes i1, i2, i3, and i4 correspond to custom page fields created in the user interface. Spreadsheet ML only supports 4 custom page fields. *end note*]

Parent Elements
rangeSets (§3.10.1.80)

Attributes	Description
i1 (Field Item Index Page 1)	Specifies the index of a page field item in page filter one. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
i2 (Field Item Index Page 2)	Specifies the index of a page field item in page filter two. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
i3 (Field Item index Page 3)	Specifies the index of a page field item in page filter three. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
i4 (Field Item Index Page 4)	Specifies the index of a page field item in page filter four. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
id (Relationship Id) Namespace: .../officeDocument/2006/relationships	Specifies the unique identifier of the Workbook part where the range set is stored. See Workbook (§) for more information. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
name (Named Range)	Specifies the named range.

Attributes	Description
	The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
ref (Reference)	Specifies the cell range. The possible values for this attribute are defined by the ST_Ref simple type (§3.18.64).
sheet (Sheet Name)	Specifies the sheet name. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RangeSet">
   <attribute name="i1" type="xsd:unsignedInt" use="optional"/>
   <attribute name="i2" type="xsd:unsignedInt" use="optional"/>
   <attribute name="i3" type="xsd:unsignedInt" use="optional"/>
   <attribute name="i4" type="xsd:unsignedInt" use="optional"/>
   <attribute name="ref" type="ST_Ref" use="optional"/>
   <attribute name="name" type="ST_Xstring" use="optional"/>
   <attribute name="sheet" type="ST_Xstring" use="optional"/>
   <attribute ref="r:id" use="optional"/>
</complexType>
```

3.10.1.80 rangeSets (Range Sets)

Represents the collection of reference-page items pairs.

Parent Elements
consolidation (§3.10.1.20)

Child Elements	Subclause
rangeSet (Range Set)	§3.10.1.79

Attributes	Description
count (Reference and Page Item Count)	Specifies the number of reference and page items. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RangeSets">
   <sequence>
      <element name="rangeSet" type="CT_RangeSet" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.10.1.81 rowFields (Row Fields)

Represents the collection of row fields for the PivotTable.

	A	B	C
1	Region	(All)	
2			
3	Sum of Sales	Quarter	
4	Sport	Qtr1	Qtr2
5	Golf	8,500	4,500

In the image above, the blue field is a row field. A PivotTable report that has more than one row field has one inner row field (Sport, in the example below), the one closest to the data area. Any other row fields are outer row fields (Region, in the example below). Items in the outermost row field are displayed only once, but items in the rest of the row fields are repeated as needed.

	A	B	C
3	Sum of Sales		Quarter
4	Region	Sport	Qtr1
5	East	Golf	5,000
6		Safari	9,000
7		Tennis	1,500
8	East Total		15,500
9	West	Golf	3,500

In the image above, Region is an outer row field. Sport is an inner row field.

[*Example*:

```
<rowFields count="2">
  <field x="7"/>
  <field x="8"/>
</rowFields>
```

end example]

Parent Elements
pivotTableDefinition (§3.10.1.73)

Child Elements	Subclause
field (Field)	§3.10.1.29

Attributes	Description
count (Repeated Items Count)	Specifies the number of repeated items in the collection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RowFields">
   <sequence>
      <element name="field" maxOccurs="unbounded" type="CT_Field"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" default="0"/>
</complexType>
```

3.10.1.82 rowHierarchiesUsage (Row OLAP Hierarchy References)

Represents the collection of references to OLAP hierarchies on the row axis of a PivotTable.

[*Example*:

```
<sh:rowHierarchiesUsage count="1">
  <sh:rowHierarchyUsage hierarchyUsage="9"/>
</sh:rowHierarchiesUsage>
```

end example]

Parent Elements
pivotTableDefinition (§3.10.1.73)

Child Elements	Subclause
rowHierarchyUsage (Row OLAP Hierarchies)	§3.10.1.83

Attributes	Description
count (Item Count)	Specifies the number of items in the collection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RowHierarchiesUsage">
   <sequence>
      <element name="rowHierarchyUsage" minOccurs="1" maxOccurs="unbounded"
        type="CT_HierarchyUsage"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.83 rowHierarchyUsage (Row OLAP Hierarchies)

Represents a references to an OLAP Hierarchy on the row axis of a PivotTable.

[*Example*:

```
<sh:rowHierarchyUsage hierarchyUsage="9"/>
```

end example]

Parent Elements
rowHierarchiesUsage (§3.10.1.82)

Attributes	Description
hierarchyUsage (Hierarchy Usage)	Specifies the reference to an OLAP hierarchy in a PivotTable. The possible values for this attribute are defined by the XML Schema int datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HierarchyUsage">
   <attribute name="hierarchyUsage" type="xsd:int" use="required"/>
</complexType>
```

3.10.1.84 rowItems (Row Items)

Represents the collection of items in row axis of the PivotTable.

In the SpreadsheetML example below, the item values are found in cells B10:B13. For example "Bikes" is in B10, and corresponds to the first `<i>` element below.

[*Example*:*L*

```
<rowItems count="4">
  <i>
    <x/>
  </i>
```

```
    <i r="1">
      <x/>
    </i>
    <i r="1">
      <x v="1"/>
    </i>
    <i t="grand">
      <x/>
    </i>
  </rowItems>
```

end example]

Looking at the layout of the PivotTable in this example, "Bikes" is the first (and only) item value in the first row, in cell B10. In the XML defining the PivotTable row item values, the first <i> element corresponds to the first row. There is a single index element <x>. The first (and only) <x> element corresponds to the first field on the row axis, namely "Product Category", and an index value of "0" indicates that the 0th item in the items collection for that pivotField definition is how to obtain the item value. Note that "Bikes" isn't explicitly listed as a value here, but instead the 0th item is an index to this field's shared items collection in the pivotCacheDefinition part.

For the second row there are two item values, one item value (Bikes) from the first field in that row (Product Category) and one item value (Mountain Bikes) from the second field in that row (Product Subcategory). In the PivotTable, the first item value "Bikes" is hidden from view. In the XML for this example, the second <i> element expresses both item values for this row. The first item value "Bikes" is expressed implicitly, because the value of @r on the second <i> element is '1', indicating that the first item value from the previous row will be reused again as the first item value for the current row. The second item value is expressed explicitly via the <x> element under the second <i> element. The index of '0' indicates that the 0th item in the <pivotField> element for that field is how to obtain the item value. Note again that the 0th item is itself an index into this field's shared items collection in the pivotCacheDefinition part.

The item values for the third row can be discovered in a similar way.

Parent Elements
pivotTableDefinition (§3.10.1.73)

Child Elements	Subclause
i (Row Items)	§3.10.1.44

Attributes	Description
count (Items in a	Specifies the number of items in the row axis of the PivotTable.

Attributes	Description
Row Count)	The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_rowItems">
   <sequence>
      <element name="i" maxOccurs="unbounded" type="CT_I"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.85 s (Character Value)

Represents a character value in a PivotTable.

[*Example*:

```
<sharedItems count="2">
  <s v="7527 Brook Way"/>
  <s v="3310 Harvey Way"/>
</sharedItems>
```

end example]

Parent Elements
entries (§3.10.1.28); groupItems (§3.10.1.38); r (§3.10.1.77); sharedItems (§3.10.1.90)

Child Elements	Subclause
tpls (Tuples)	§3.10.1.93
x (Member Property Index)	§3.10.1.96

Attributes	Description
b (Bold)	Specifies a boolean value that indicates whether this value contains bold formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of on, 1, or true indicates this value contains bold formatting on the server. The possible values for this attribute are defined by the XML Schema boolean datatype.
bc (Background Color)	Specifies the background color for this value that was provided by the OLAP server. This attribute applies to OLAP-based PivotTables only. The color is specified as a HEX value in RGB space.

Attributes	Description
	The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§3.18.86).
c (Item Caption)	Specifies the caption for the this item. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
cp (Member Property Count)	Specifies the number of member property values. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
f (Calculated Item)	Specifies a boolean value that indicates whether this is a calculated item value. A value of on, 1, or true indicates this item is a calculated value. A value of off, 0, or false indicates this item is not a calculated value. The possible values for this attribute are defined by the XML Schema boolean datatype.
fc (Foreground Color)	Specifies the foreground color for this value that was provided by the OLAP server. This attribute applies to OLAP-based PivotTables only. The color is specified as a HEX value in RGB space. The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§3.18.86).
i (Italic)	Specifies a boolean value that indicates whether the value contains italic formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of on, 1, or true indicates this value contains italic formatting on the server. The possible values for this attribute are defined by the XML Schema boolean datatype.
in (Format Index)	Specifies the index to the OLAP serverformat element where the format string for this entry is stored. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
st (Strikethrough)	Specifies a boolean value that indicates whether the value contains strikethrough formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of on, 1, or true indicates this value contains strikethrough formatting on the server. The possible values for this attribute are defined by the XML Schema boolean datatype.
u (Unused Item)	Specifies a boolean value that indicates whether this is an unused item. The application

Attributes	Description
	marks an item as unused when an item is deleted from the data source. The item and associated metadata are retained in the cache until the threshold for unused items specified in missingItemsLimit is reached. A value of on, 1, or true indicates this item is unused. A value of off, 0, or false indicates this item is used. The possible values for this attribute are defined by the XML Schema boolean datatype.
un (Underline)	Specifies a boolean value that indicates whether the value contains underline formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of on, 1, or true indicates this value contains underline formatting on the server. The possible values for this attribute are defined by the XML Schema boolean datatype.
v (Value)	Specifies the value of the item. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <sequence>
      <element name="tpls" minOccurs="0" maxOccurs="unbounded" type="CT_Tuples"/>
      <element name="x" minOccurs="0" maxOccurs="unbounded" type="CT_X"/>
   </sequence>
   <attribute name="v" use="required" type="ST_Xstring"/>
   <attribute name="u" type="xsd:boolean"/>
   <attribute name="f" type="xsd:boolean"/>
   <attribute name="c" type="ST_Xstring"/>
   <attribute name="cp" type="xsd:unsignedInt"/>
   <attribute name="in" type="xsd:unsignedInt" use="optional"/>
   <attribute name="bc" type="ST_UnsignedIntHex" use="optional"/>
   <attribute name="fc" type="ST_UnsignedIntHex" use="optional"/>
   <attribute name="i" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="un" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="st" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="b" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.10.1.86 serverFormat (Server Format)

Represents the numeric format specified by the OLAP server for a tuple.

Parent Elements
serverFormats (§3.10.1.87)

Attributes	Description
culture (Culture)	Specifies a language used to determine the currency symbol to display for currency values. For example, if the culture is "en-us", the values in the application will format the values with a dollar sign. If the culture is "fr-fr" the application will format the values with a euro sign. This value conforms to the language tagging conventions of RFC 3066 and later. The pattern <language>-<REGION> is used, e.g., "en-us" or "fr-fr". The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
format (Format)	Specifies the format string to use for all other numeric values. This string is supplied by the OLAP server. Therefore, the syntax for reading the format string depends on the server implementation. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ServerFormat">
   <attribute name="culture" use="optional" type="ST_Xstring"/>
   <attribute name="format" use="optional" type="ST_Xstring"/>
</complexType>
```

3.10.1.87 serverFormats (Server Formats)

Represents the collection of numeric and currency formats specified by the OLAP server for a tuple

Parent Elements
tupleCache (§3.10.1.94)

Child Elements	Subclause
serverFormat (Server Format)	§3.10.1.86

Attributes	Description
count (Format Count)	Specifies the number of formats in the collection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ServerFormats">
   <sequence>
      <element name="serverFormat" type="CT_ServerFormat" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.88 set (OLAP Set)

Represents an OLAP sheet data set or tuple set. The set is defined by a Multidimensional Expressions (MDX) query that specifies criteria for the dimension members that belong to the set.

For example, the following MDX expression defines the set for the 10 salespersons with the lowest sales:

BottomCount([Salesperson].[Salesperson Name].Members,10,[Measures].[Sales])

The MDX expression is specified in the setDefinition attribute.

Parent Elements
sets (§3.10.1.89)

Child Elements	Subclause
sortByTuple (Sort By Tuple)	§3.10.1.91
tpls (Tuples)	§3.10.1.93

Attributes	Description
count (Number of Tuples)	Specifies the number of tuples in the set. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
maxRank (Maximum Rank Requested)	Specifies the largest rank entry the user has requested. The possible values for this attribute are defined by the XML Schema int datatype.
queryFailed (Query Failed)	Specifies a boolean value that indicates whether querying on this set failed. A value of on, 1, or true indicates a query against this set failed. A value of off, 0, or false indicates a query against this set succeeded. The possible values for this attribute are defined by the XML Schema boolean datatype.
setDefinition (MDX	Specifies the Multidimensional Expressions (MDX) set definition.

Attributes	Description
Set Definition)	[*Note*: Data connectivity can use a number of different technologies. One example of potential values stored in this attribute can be found at: http://msdn2.microsoft.com/en-us/library/ms145595.aspx *end note*] The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
sortType (Set Sort Order)	Specifies the sort order of the set. The possible values for this attribute are defined by the ST_SortType simple type (§3.18.76).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Set">
   <sequence>
      <element name="tpls" minOccurs="0" maxOccurs="unbounded" type="CT_Tuples"/>
      <element name="sortByTuple" minOccurs="0" type="CT_Tuples"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
   <attribute name="maxRank" use="required" type="xsd:int"/>
   <attribute name="setDefinition" use="required" type="ST_Xstring"/>
   <attribute name="sortType" type="ST_SortType" default="none"/>
   <attribute name="queryFailed" type="xsd:boolean" default="false"/>
</complexType>
```

3.10.1.89 sets (Sets)

Represents the collection of OLAP sheet data entries or tuple sets.

Parent Elements
tupleCache (§3.10.1.94)

Child Elements	Subclause
set (OLAP Set)	§3.10.1.88

Attributes	Description
count (Tuple Set Count)	Specifies the number of tuple sets. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Sets">
   <sequence>
      <element name="set" maxOccurs="unbounded" type="CT_Set"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.10.1.90 sharedItems (Shared Items)

Represents the collection of unique items for a field in the PivotCacheDefinition. The sharedItems complex type stores data type and formatting information about the data in a field. Items in the PivotCacheDefinition can be shared in order to reduce the redundancy of those values that are referenced in multiple places across all the PivotTable parts. For example, a value might be part of a filter, it might appear on a row or column axis, and will appear in the pivotCacheRecords definition as well. However, because of the performance cost of creating the optimized shared items, items are only shared if they are actually in use in the PivotTable. Therefore, depending on user actions on the PivotTable layout, the pivotCacheDefinition and underlying PivotCacheRecords part may be updated.

If there are no shared items, then field values are stored directly in the pivotCacheRecords part.

[*Example*:

```
<sharedItems count="1">
  <s v="[Customer].[Customer Geography].[Country].&amp;[United States]"
    c="United States"/>
</sharedItems>
```

end example]

The following attributes are not required or validated if there are no items in sharedItems.

- containsBlank
- containsSemiMixedTypes
- containsMixedTypes
- longText

The following attributes not validated unless there is more than one item in sharedItems or the one and only item is not a blank item. If the first item is a blank item the data type the field cannot be verified.

- containsNumber
- containsDates
- containsString
- containsInteger

The following attributes are not validated and can be omitted without loss of functionality.

- containsNonDate
- count

The following attributes are not required and can be omitted. However, refreshing the PivotTable could produce different groupings than before.

- maxDate
- minDate
- maxValue
- minValue

Note: validation is performed to ensure that “date” attributes are not mixed with “value” attributes.

Parent Elements
cacheField (§3.10.1.3)

Child Elements	Subclause
b (Boolean)	§3.10.1.2
d (Date Time)	§3.10.1.21
e (Error Value)	§3.10.1.27
m (No Value)	§3.10.1.50
n (Numeric)	§3.10.1.60
s (Character Value)	§3.10.1.85

Attributes	Description
containsBlank (Contains Blank)	Specifies a boolean value that indicates whether this field contains a blank value. A value of on, 1, or true indicates this field contains one or more blank values. A value of off, 0, or false indicates this field does not contain blank values. The possible values for this attribute are defined by the XML Schema boolean datatype.
containsDate (Contains Date)	Specifies a boolean value that indicates that the field contains at least one date. A value of on, 1, or true indicates the field contains at least one date value. A value of off, 0, or false indicates the field does not contain any date values. The possible values for this attribute are defined by the XML Schema boolean datatype.
containsInteger (Contains Integer)	Specifies a boolean value that indicates whether this field contains integer values.

Attributes	Description
	A value of on, 1, or true indicates this field contains integer values. A value of off, 0, or false indicates non-integer or mixed values. The possible values for this attribute are defined by the XML Schema boolean datatype.
containsMixedTypes (Contains Mixed Data Types)	Specifies a boolean value that indicates whether this field contains more than one data type. A value of on, 1, or true indicates this field contains more than one data type. A value of off, 0, or false indicates contains only one data type. The field may still contain blank values. The possible values for this attribute are defined by the XML Schema boolean datatype.
containsNonDate (Contains Non Date)	Specifies a boolean value that indicates that the field contains at least one value that is not a date. A value of on, 1, or true indicates the field contains at least one non-date values. A value of off, 0, or false indicates this field contains no date fields. The possible values for this attribute are defined by the XML Schema boolean datatype.
containsNumber (Contains Numbers)	Specifies a boolean value that indicates whether this field contains numeric values. A value of on, 1, or true indicates this field contains at least one numeric value. A value of off, 0, or false indicates this field contains no numeric values. The possible values for this attribute are defined by the XML Schema boolean datatype.
containsSemiMixedTypes (Contains Semi Mixed Data Types)	Specifies a boolean value that indicates that this field contains text values. The field may also contain a mix of other data type and blank values. A value of on, 1, or true indicates at least one text value, and may also contain a mix of other data types and blank values. A value of off, 0, or false indicates the field does not have a mix of text and other values. The possible values for this attribute are defined by the XML Schema boolean datatype.
containsString (Contains String)	Specifies a boolean value that indicates whether this field contains a text value. A value of on, 1, or true indicates this field contains at least one text value. A value of off, 0, or false indicates this field does not contain any text values.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema boolean datatype.
count (Shared Items Count)	Specifies the number of shared items to load for this field. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
longText (Long Text)	Specifies a boolean value that indicates whether this field contains a long text value. A string is considered long if it is over 255 characters. A value of on, 1, or true indicates the value contains more than 255 characters of text. A value of off, 0, or false indicates the value contains less than 255 characters. [*Note*: This is used as many legacy spreadsheet application support a limit of 255 characters for text values. *end note*] The possible values for this attribute are defined by the XML Schema boolean datatype.
maxDate (Maximum Date Time Value)	Specifies the maximum date/time value found in a date field. The possible values for this attribute are defined by the XML Schema dateTime datatype.
maxValue (Maximum Numeric Value)	Specifies the maximum numeric value found in a numeric field. The possible values for this attribute are defined by the XML Schema double datatype.
minDate (Minimum Date Time)	Specifies the minimum date/time value found in a date field. The possible values for this attribute are defined by the XML Schema dateTime datatype.
minValue (Minimum Numeric Value)	Specifies the minimum numeric value found in a numeric field. The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SharedItems">
   <choice minOccurs="0" maxOccurs="unbounded">
      <element name="m" type="CT_Missing" minOccurs="1" maxOccurs="1"/>
      <element name="n" type="CT_Number" minOccurs="1" maxOccurs="1"/>
      <element name="b" type="CT_Boolean" minOccurs="1" maxOccurs="1"/>
      <element name="e" type="CT_Error" minOccurs="1" maxOccurs="1"/>
      <element name="s" type="CT_String" minOccurs="1" maxOccurs="1"/>
      <element name="d" type="CT_DateTime" minOccurs="1" maxOccurs="1"/>
   </choice>
   <attribute name="containsSemiMixedTypes" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="containsNonDate" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="containsDate" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="containsString" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="containsBlank" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="containsMixedTypes" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="containsNumber" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="containsInteger" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="minValue" type="xsd:double" use="optional"/>
   <attribute name="maxValue" type="xsd:double" use="optional"/>
   <attribute name="minDate" type="xsd:dateTime" use="optional"/>
   <attribute name="maxDate" type="xsd:dateTime" use="optional"/>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
   <attribute name="longText" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.10.1.91 sortByTuple (Sort By Tuple)

Represents the sort applied to a tuple.

Parent Elements
set (§3.10.1.88)

Child Elements	Subclause
tpl (Tuple)	§3.10.1.92

Attributes	Description
c (Member Name Count)	Specifies the number of member names. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Tuples">
   <sequence>
      <element name="tpl" type="CT_Tuple" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="c" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.10.1.92 tpl (Tuple)

Represents an OLAP sheet data entry member.

Parent Elements
sortByTuple (§3.10.1.91); tpls (§3.10.1.93)

Attributes	Description
fld (Field Index)	Specifies the index of the field to which the member belongs. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
hier (Hierarchy Index)	Specified the index of the hierarchy to which the member belongs. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
item (Item Index)	Specifies the index of the item in the field that represents this item. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Tuple">
   <attribute name="fld" type="xsd:unsignedInt"/>
   <attribute name="hier" type="xsd:unsignedInt"/>
   <attribute name="item" type="xsd:unsignedInt" use="required"/>
</complexType>
```

3.10.1.93 tpls (Tuples)

Represents members for the OLAP sheet data entry, also known as a tuple.

Parent Elements
e (§3.10.1.27); m (§3.10.1.50); n (§3.10.1.60); query (§3.10.1.75); s (§3.10.1.85); set (§3.10.1.88)

Child Elements	Subclause

Child Elements	Subclause
tpl (Tuple)	§3.10.1.92

Attributes	Description
c (Member Name Count)	Specifies the number of member names. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Tuples">
   <sequence>
      <element name="tpl" type="CT_Tuple" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="c" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.10.1.94 tupleCache (Tuple Cache)

Represents the cache of OLAP sheet data members, or tuples.

Parent Elements
pivotCacheDefinition (§3.10.1.67)

Child Elements	Subclause
entries (Entries)	§3.10.1.28
extLst (Future Feature Data Storage Area)	§3.2.10
queryCache (OLAP Query Cache)	§3.10.1.76
serverFormats (Server Formats)	§3.10.1.87
sets (Sets)	§3.10.1.89

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TupleCache">
   <sequence>
      <element name="entries" minOccurs="0" type="CT_PCDSDTCEntries"/>
      <element name="sets" minOccurs="0" type="CT_Sets"/>
      <element name="queryCache" minOccurs="0" type="CT_QueryCache"/>
      <element name="serverFormats" minOccurs="0" maxOccurs="1" type="CT_ServerFormats"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
</complexType>
```

3.10.1.95 worksheetSource (Worksheet PivotCache Source)

Represents the location of the source of the data that is stored in the cache.

[*Example*:

```
<cacheSource type="worksheet">
  <worksheetSource name="Table1" r:id="rId2"/>
</cacheSource>
```

end example]

Parent Elements
cacheSource (§3.10.1.7)

Attributes	Description
id (Relationship Id) Namespace: .../officeDocument/2006/relationships	Specifies the identifier to the Sheet part whose data is stored in the cache. See the Sheet section (§3.2) for more information. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
name (Named Range)	Specifies the named range that is the source of the data. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
ref (Reference)	Specifies the reference that defines a cell range that is the source of the data. This attribute depends on how the application implements cell references. The possible values for this attribute are defined by the ST_Ref simple type (§3.18.64).
sheet (Sheet Name)	Specifies the name of the sheet that is the source for the cached data. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_WorksheetSource">
   <attribute name="ref" type="ST_Ref" use="optional"/>
   <attribute name="name" type="ST_Xstring" use="optional"/>
   <attribute name="sheet" type="ST_Xstring" use="optional"/>
   <attribute ref="r:id" use="optional"/>
</complexType>
```

3.10.1.96 x (Member Property Index)

Represents an array of indexes to cached member property values.

Parent Elements
b (§3.10.1.2); d (§3.10.1.21); e (§3.10.1.27); i (§3.10.1.44); m (§3.10.1.50); n (§3.10.1.60); s (§3.10.1.85)

Attributes	Description
v (Shared Items Index)	Specifies the index into the shared items table in the PivotCache that identifies this item. The possible values for this attribute are defined by the XML Schema int datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_X">
   <attribute name="v" type="xsd:int" default="0"/>
</complexType>
```

3.10.1.97 x (Shared Items Index)

This element represents an array of indexes to cached shared item values

Parent Elements
discretePr (§3.10.1.26); r (§3.10.1.77); reference (§3.10.2.1); tables (§3.13.9)

Attributes	Description
v (Shared Items Index)	Specifies the index into the shared items table in the PivotCache that identifies this item. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Index">
   <attribute name="v" use="required" type="xsd:unsignedInt"/>
</complexType>
```

3.10.2 Shared Pivot Table Data

This section defines the part where shared PivotTable data is stored.

3.10.2.1 reference (Reference)

Represents a set of selected fields and selected items within those fields.

[*Example*:

```
<sh:reference field="4294967294" count="1" selected="0">
  <sh:x v="0"/>
</sh:reference>
```

end example]

Parent Elements
references (§3.10.2.2)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
x (Shared Items Index)	§3.10.1.97

Attributes	Description
avgSubtotal (Include Average Filter)	Specifies a boolean value that indicates whether the 'average' aggregate function is included in the filter. A value of on, 1, or true indicates the average aggregation function will is included in the filter. A value of off, 0, or false indicates another aggregation function is included in the filter. The possible values for this attribute are defined by the XML Schema boolean datatype.
byPosition (Positional Reference)	Specifies a boolean value that indicates whether the item is referred to by position rather than item index. A value of on, 1, or true indicates the item is referred to by position. A value of off, 0, or false indicates the item is referred to by index. The possible values for this attribute are defined by the XML Schema boolean datatype.
count (Item Index Count)	Specifies the number of item indexes in the collection of indexes (x tags). The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
countASubtotal (Include CountA Filter)	Specifies a boolean value that indicates whether the 'countA' subtotal is included in the filter. A value of on, 1, or true indicates the count aggregation function will is included in the filter. A value of off, 0, or false indicates another aggregation function is included in the filter. The possible values for this attribute are defined by the XML Schema boolean datatype.
countSubtotal (Include Count	Specifies a boolean value that indicates whether the count aggregate function is included in the filter.

Attributes	Description
Subtotal)	A value of on, 1, or true indicates the count aggregation function will is included in the filter. A value of off, 0, or false indicates another aggregation function is included in the filter. The possible values for this attribute are defined by the XML Schema boolean datatype.
defaultSubtotal (Include Default Filter)	Specifies a boolean value that indicates whether the default subtotal is included in the filter. A value of on, 1, or true indicates the default subtotal will is included in the filter. The default is to display the total or the grand total. A value of off, 0, or false indicates another subtotal or aggregation function is included in the filter. The possible values for this attribute are defined by the XML Schema boolean datatype.
field (Field Index)	Specifies the index of the field to which this filter refers. A value of -2 indicates the 'data' field. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
maxSubtotal (Include Maximum Filter)	Specifies a boolean value that indicates whether the 'maximum' aggregate function is included in the filter. A value of on, 1, or true indicates the maximum aggregation function will is included in the filter. A value of off, 0, or false indicates another aggregation function is included in the filter. The possible values for this attribute are defined by the XML Schema boolean datatype.
minSubtotal (Include Minimum Filter)	Specifies a boolean value that indicates whether the 'minimum' aggregate function is included in the filter. A value of on, 1, or true indicates the minimum aggregation function will is included in the filter. A value of off, 0, or false indicates another aggregation function is included in the filter. The possible values for this attribute are defined by the XML Schema boolean datatype.
productSubtotal (Include Product Filter)	Specifies a boolean value that indicates whether the 'product' aggregate function is included in the filter. A value of on, 1, or true indicates the product aggregation function will is included in the

Attributes	Description
	filter. A value of off, 0, or false indicates another aggregation function is included in the filter. The possible values for this attribute are defined by the XML Schema boolean datatype.
relative (Relative Reference)	Specifies a boolean value that indicates whether the item is referred to by a relative reference rather than an absolute reference. This attribute is used if posRef is set to true. A value of on, 1, or true indicates the item is referred to by a relative reference. A value of off, 0, or false indicates the item is referred to by an absolute reference. The possible values for this attribute are defined by the XML Schema boolean datatype.
selected (Selected)	Specifies a boolean value that indicates whether this field has selection. This attribute is used when the PivotTable is in Outline view. It is also used when both header and data cells have selection. A value of on, 1, or true indicates the field has selection. A value of off, 0, or false indicates the field does not have selection. The possible values for this attribute are defined by the XML Schema boolean datatype.
stdDevPSubtotal (Include StdDevP Filter)	Specifies a boolean value that indicates whether the population standard deviation aggregate function is included in the filter. A value of on, 1, or true indicates the population standard deviation aggregation function will is included in the filter. A value of off, 0, or false indicates another aggregation function is included in the filter. The possible values for this attribute are defined by the XML Schema boolean datatype.
stdDevSubtotal (Include StdDev Filter)	Specifies a boolean value that indicates whether the standard deviation aggregate function is included in the filter. A value of on, 1, or true indicates the standard deviation aggregation function will is included in the filter. A value of off, 0, or false indicates another aggregation function is included in the filter. The possible values for this attribute are defined by the XML Schema boolean datatype.
sumSubtotal (Include Sum Filter)	Specifies a boolean value that indicates whether the sum aggregate function is included in the filter.

Attributes	Description
	A value of on, 1, or true indicates the sum aggregation function will is included in the filter. A value of off, 0, or false indicates another aggregation function is included in the filter. The possible values for this attribute are defined by the XML Schema boolean datatype.
varPSubtotal (Include VarP Filter)	Specifies a boolean value that indicates whether the population variance aggregate function is included in the filter. A value of on, 1, or true indicates the population variance aggregation function will is included in the filter. A value of off, 0, or false indicates another aggregation function is included in the filter. The possible values for this attribute are defined by the XML Schema boolean datatype.
varSubtotal (Include Var Filter)	Specifies a boolean value that indicates whether the variance aggregate function is included in the filter. A value of on, 1, or true indicates the variance aggregation function will is included in the filter. A value of off, 0, or false indicates another aggregation function is included in the filter. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PivotAreaReference">
   <sequence>
      <element name="x" minOccurs="0" maxOccurs="unbounded" type="CT_Index"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="field" use="optional" type="xsd:unsignedInt"/>
   <attribute name="count" type="xsd:unsignedInt"/>
   <attribute name="selected" type="xsd:boolean" default="true"/>
   <attribute name="byPosition" type="xsd:boolean" default="false"/>
   <attribute name="relative" type="xsd:boolean" default="false"/>
   <attribute name="defaultSubtotal" type="xsd:boolean" default="false"/>
   <attribute name="sumSubtotal" type="xsd:boolean" default="false"/>
   <attribute name="countASubtotal" type="xsd:boolean" default="false"/>
   <attribute name="avgSubtotal" type="xsd:boolean" default="false"/>
   <attribute name="maxSubtotal" type="xsd:boolean" default="false"/>
   <attribute name="minSubtotal" type="xsd:boolean" default="false"/>
   <attribute name="productSubtotal" type="xsd:boolean" default="false"/>
   <attribute name="countSubtotal" type="xsd:boolean" default="false"/>
   <attribute name="stdDevSubtotal" type="xsd:boolean" default="false"/>
   <attribute name="stdDevPSubtotal" type="xsd:boolean" default="false"/>
   <attribute name="varSubtotal" type="xsd:boolean" default="false"/>
   <attribute name="varPSubtotal" type="xsd:boolean" default="false"/>
</complexType>
```

3.10.2.2 references (References)

Represents the set of selected fields and the selected items within those fields.

[*Example*:

```
<sh:references count="5">
  <sh:reference field="4294967294" count="1" selected="0">
    <sh:x v="0"/>
  </sh:reference>
  <sh:reference field="2" count="1" selected="0">
    <sh:x v="0"/>
  </sh:reference>
  <sh:reference field="14" count="1" selected="0">
    <sh:x v="0"/>
  </sh:reference>
  <sh:reference field="15" count="2" selected="0">
    <sh:x v="2"/>
    <sh:x v="3"/>
  </sh:reference>
</sh:references>
```

end example]

Parent Elements
pivotArea (§3.3.1.66)

Child Elements	Subclause
reference (Reference)	§3.10.2.1

Attributes	Description
count (Pivot Filter Count)	Specifies the number of filtered records available in the PivotTable. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PivotAreaReferences">
   <sequence>
      <element name="reference" maxOccurs="unbounded" type="CT_PivotAreaReference"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt"/>
</complexType>
```

3.11 Shared Workbook Data

The Shared Workbooks architecture enables a spreadsheet application to record revisions made to a workbook (e.g., track changes), and is designed to enable either single, or multiple users editing the same workbook at the same time. Therefore the application needs to support the ability to read changes made by another user and update it's own state of the same workbook with those changes, even when those changes are made concurrently with other changes made by other users. Inevitably there will be conflicts, and therefore merge conflict resolution should be supported the runtime application. The file format only contains enough information so that the spreadsheet application can deal with conflicts, and can undo/redo changes from the change history at run time.

3.11.1 Shared Workbook Data

Within a shared workbook, the changes made to the spreadsheet at runtime are persisted as sets of different revisions collectively forming a revision history. These are persisted to the file on disk during a save event, and are saved in different xml parts known as revision logs. There is a headers table xml part that summarizes when changes were made, who made them, and it lists the relationship from each header to the individual revision log that records the specific changes.

[*Example*: This example shows the header and revision log for two simple events: adding text to a cell, and inserting a new sheet.

First, take a look at the header table, and revision log:

```
<headers xmlns="…" xmlns:r="…" guid="{A84A6777-8908-4CB9-9EB6-625CEFF419D3}">
  <header guid="{A84A6777-8908-4CB9-9EB6-625CEFF419D3}"
    dateTime="2006-07-14T13:42:54" maxSheetId="4" userName="UserName"
    r:id="rId1">
    <sheetIdMap count="3">
      <sheetId val="1"/>
      <sheetId val="2"/>
      <sheetId val="3"/>
    </sheetIdMap>
  </header>
</headers>
```

And the revision log is essentially empty:

```
<revisions xmlns="…" xmlns:r="…"/>
```

Now, after inserting the text "foo" into cell A1, and saving, the header looks like this:

```
<headers xmlns="…" xmlns:r="…" guid="{CFEA9B63-728B-4274-A346-0440E1573AB4}"
  diskRevisions="1" revisionId="1" version="2">
  <header guid="{A84A6777-8908-4CB9-9EB6-625CEFF419D3}"
    dateTime="2006-07-14T13:42:54" maxSheetId="4" userName="UserName"
    r:id="rId1">
    <sheetIdMap count="3">
      <sheetId val="1"/>
      <sheetId val="2"/>
      <sheetId val="3"/>
    </sheetIdMap>
  </header>
  <header guid="{CFEA9B63-728B-4274-A346-0440E1573AB4}"
    dateTime="2006-07-14T13:44:40" maxSheetId="4" userName="UserName"
    r:id="rId2" minRId="1">
    <sheetIdMap count="3">
      <sheetId val="1"/>
      <sheetId val="2"/>
      <sheetId val="3"/>
    </sheetIdMap>
  </header>
</headers>
```

A new header entry is added, with a GUID and a revision ID (rId2) that specifies which log to look into to see the details about the revision.

The old log is saved, and the newly created log (corresponding to rId2) now looks like this:

```
<revisions xmlns="…" xmlns:r="…">
  <rcc rId="1" sId="1">
    <nc r="A1" t="inlineStr">
      <is>
        <t>foo</t>
        <phoneticPr fontId="0"/>
      </is>
    </nc>
  </rcc>
</revisions>
```

The log shows that the contents of a cell were revised, and the new cell contents is text containg "foo" as the string.

After inserting a new sheet, the header looks like this:

```
<headers xmlns="…" xmlns:r="…" guid="{7E1DAFA8-EF95-4865-8FE8-CC17B28635CF}"
  diskRevisions="1" revisionId="2" version="3">
  <header guid="{A84A6777-8908-4CB9-9EB6-625CEFF419D3}"
    dateTime="2006-07-14T13:42:54" maxSheetId="4"
    userName="UserName" r:id="rId1">
    <sheetIdMap count="3">
      <sheetId val="1"/>
      <sheetId val="2"/>
      <sheetId val="3"/>
    </sheetIdMap>
  </header>
  <header guid="{CFEA9B63-728B-4274-A346-0440E1573AB4}"
    dateTime="2006-07-14T13:44:40" maxSheetId="4" userName="UserName"
    r:id="rId2" minRId="1">
    <sheetIdMap count="3">
      <sheetId val="1"/>
      <sheetId val="2"/>
      <sheetId val="3"/>
    </sheetIdMap>
  </header>
  <header guid="{7E1DAFA8-EF95-4865-8FE8-CC17B28635CF}"
    dateTime="2006-07-14T13:48:56" maxSheetId="5" userName="UserName"
    r:id="rId3" minRId="2">
```

```
      <sheetIdMap count="4">
        <sheetId val="1"/>
        <sheetId val="2"/>
        <sheetId val="3"/>
        <sheetId val="4"/>
      </sheetIdMap>
    </header>
  </headers>
```

You can see that the last, most recent, header entry shows an entry for the new sheet. The most recent log looks like this:

```
  <revisions xmlns="…" xmlns:r="…">
    <ris rId="2" sheetId="4" name="[shared example.xlsx]Sheet4"
      sheetPosition="3"/>
    <rcv guid="{841DBE00-ECD0-478E-893B-30CE5DABBEF5}" action="delete"/>
    <rcv guid="{841DBE00-ECD0-478E-893B-30CE5DABBEF5}" action="add"/>
  </revisions>
```

This shows the new sheet, sheetId 4, is added to the workbook. The custom view (rcv) for the user is updated as a new sheet was added.

end example]

3.11.1.1 header (Header)

This element is essentially a table that contains metadata about a list of specific changes that have taken place for this workbook. It lists when the changes were made, who made them, and the relationship IDs so that the log detailing the specific change can be found. If tracking changes, or sharing workbooks, are enabled, then changes are persisted on the Save event, or at a specified time interval. A header is created for each set of changes.

Parent Elements
Root element of SpreadsheetML Shared Workbook Revision Headers partheaders (§3.11.1.2)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
reviewedList (Reviewed List)	§3.11.1.15
sheetIdMap (Sheet Id Map)	§3.11.1.24

Attributes	Description
dateTime (Date	The date and time when this set of revisions was saved.

Attributes	Description
Time)	Note: This can happen when the user explicitly saves, or the save can occur due to a time interval, specified in the spreadsheet application, elapsing. The possible values for this attribute are defined by the XML Schema dateTime datatype.
guid (GUID)	A globally unique identifier for this set of revisions. The possible values for this attribute are defined by the ST_Guid simple type (§3.18.41).
id (Relationship ID) Namespace: .../officeDocument/2006/relationships	This is the ID that is used to find the corresponding log record of the changes made for this header. Use the corresponding relationship expressed in the revisionHeaders part to locate the log record that lists the specific changes. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
maxRId (Max Revision Id)	The highest revision Id that belongs to this header. Note: this can be used when, given a revision ID, the spreadsheet application needs to determine which revision log to access. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
maxSheetId (Last Sheet Id)	Internal identifier of the next available sheet in this workbook. Note: the numbering here should be the index of the next available sheet in the workbook in a 1-based index system. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
minRId (Minimum Revision Id)	The lowest revision id that belongs to this header. Note: this can be used when, given a revision ID, the spreadsheet application needs to determine which revision log to access. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
userName (User Name)	A string representing the name of the user making the revision.. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RevisionHeader">
   <sequence>
      <element name="sheetIdMap" minOccurs="1" maxOccurs="1" type="CT_SheetIdMap"/>
      <element name="reviewedList" minOccurs="0" maxOccurs="1" type="CT_ReviewedRevisions"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="guid" type="ST_Guid" use="required"/>
   <attribute name="dateTime" type="xsd:dateTime" use="required"/>
   <attribute name="maxSheetId" type="xsd:unsignedInt" use="required"/>
   <attribute name="userName" type="ST_Xstring" use="required"/>
   <attribute ref="r:id" use="required"/>
   <attribute name="minRId" type="xsd:unsignedInt" use="optional"/>
   <attribute name="maxRId" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.11.1.2 headers (Revision Headers)

This element represents the list of revision headers.

This section contains many references to history, versions, and revisions, and it is helpful to clarify the relationships here. In general, a series of changes (revisions) can be made to a spreadsheet. When a batch of those revisions is saved to disk, the version number of the spreadsheet is incremented. The batch of changes is saved to the revision history, which is persisted on disk with the file in the form of different log files and headers.

There are some attributes that deal with history which may seem redundant (such as `diskRevisions`, and `history`, among others) - these are there for backwards compatibility with older versions of spreadsheet applications and do not need to be used for creating new files.

Child Elements	Subclause
header (Header)	§3.11.1.1

Attributes	Description
diskRevisions (Disk Revisions)	A Boolean value indicating that this shared workbook file contains revisions. `True` when the workbook does have revisions, `false` otherwise. [*Note*: this attribute is used for backwards compatibility. *end note*] The possible values for this attribute are defined by the XML Schema boolean datatype.
exclusive (Exclusive Mode)	A Boolean value indicating that this shared workbook is in exclusive mode. A workbook is in exclusive mode when a user has a lock on it for appending revisions to the file. [*Note*: This is used for backwards compatibility with older spreadsheet applications. *end*

Attributes	Description
	note] The possible values for this attribute are defined by the XML Schema boolean datatype.
guid (Last Revision GUID)	The globally unique identifier of the last set of revisions. This shall match the GUID for the most recent header. The possible values for this attribute are defined by the ST_Guid simple type (§3.18.41).
history (History)	A Boolean value indicating that this shared workbook maintains a revision history. True if a history is maintained, false otherwise. [*Note*: This is used for backwards compatibility with older spreadsheet applications. *end note*] The possible values for this attribute are defined by the XML Schema boolean datatype.
keepChangeHistory (Keep Change History)	A Boolean value indicating whether the revision history should be kept for this shared workbook. True if the history should be kept, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.
lastGuid (Last GUID)	Unique identifier of the last set of revisions that was saved into the file. Note: The spreadsheet application may have certain modes, such as a timed save mode, where the application doesn't do a full save, but instead just appends the most recent revision records. In cases like this, for a new user that opens such a file while it is being edited, the file that was loaded from disk will only have the changes that were saved during a full save. To get the current state of the file which includes edits by other users, the spreadsheet application would need to apply all the revisions from `lastGuid` to `guid`. The possible values for this attribute are defined by the ST_Guid simple type (§3.18.41).
preserveHistory (Preserve History)	An integer representing the number of days the spreadsheet application will keep the change history for this workbook. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
protected (Protected)	A Boolean value indicating whether the change tracking in this shared workbook can be removed. True if the tracking can be removed, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.
revisionId (Revision Id)	The current revision number of this shared workbook. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
shared (Shared Workbook)	A Boolean value indicating that this workbook is shared. True when the workbook is shared, false otherwise.

Attributes	Description
	[*Note*: This is used for backwards compatibility with older spreadsheet applications. *end note*] The possible values for this attribute are defined by the XML Schema boolean datatype.
trackRevisions (Track Revisions)	A Boolean value indicating that revisions are tracked in this shared workbook. True when revisions are tracked, false otherwise. [*Note*: This is used for backwards compatibility with older spreadsheet applications. *end note*] The possible values for this attribute are defined by the XML Schema boolean datatype.
version (Version)	An integer representing the current version of this shared workbook. The integer should begin counting from 1 for the first version. The possible values for this attribute are defined by the XML Schema int datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RevisionHeaders">
    <sequence>
        <element name="header" type="CT_RevisionHeader" minOccurs="1" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="guid" type="ST_Guid" use="required"/>
    <attribute name="lastGuid" type="ST_Guid" use="optional"/>
    <attribute name="shared" type="xsd:boolean" default="true"/>
    <attribute name="diskRevisions" type="xsd:boolean" default="false"/>
    <attribute name="history" type="xsd:boolean" default="true"/>
    <attribute name="trackRevisions" type="xsd:boolean" default="true"/>
    <attribute name="exclusive" type="xsd:boolean" default="false"/>
    <attribute name="revisionId" type="xsd:unsignedInt" default="0"/>
    <attribute name="version" type="xsd:int" default="1"/>
    <attribute name="keepChangeHistory" type="xsd:boolean" use="optional" default="true"/>
    <attribute name="protected" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="preserveHistory" type="xsd:unsignedInt" default="30"/>
</complexType>
```

3.11.1.3 nc (New Cell Data)

This element represents new cell data that was added to the worksheet.

For most spreadsheet application purposes, only the data type and reference will need to be used for revision tracking purposes. The rest of the cell properties can be written out, but are not necessarily needed as they can be recorded in other areas of the spreadsheet. For instance the <rfmt> element can be used to record style information instead of the `S (style index)` attribute.

Parent Elements

Parent Elements
rcc (§3.11.1.9)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
f (Formula)	§3.3.1.37
is (Rich Text Inline)	§3.3.1.50
v (Cell Value)	§3.3.1.93

Attributes	Description
cm (Cell Metadata Index)	The zero-based index of the cell metadata record associated with this cell. Metadata information is found in the Metadata Part. Cell metadata is extra information stored at the cell level, and is attached to the cell (travels through moves, copy / paste, clear, etc). Cell metadata is not accessible via formula reference. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
ph (Show Phonetic)	A Boolean value indicating if the spreadsheet application should show phonetic information. Phonetic information is displayed in the same cell across the top of the cell and serves as a 'hint' which indicates how the text should be pronounced. This should only be used for East Asian languages. The possible values for this attribute are defined by the XML Schema boolean datatype.
r (Reference)	An A1 style reference to the location of this cell The possible values for this attribute are defined by the ST_CellRef simple type (§3.18.8).
s (Style Index)	The index of this cell's style. Style records are stored in the Styles Part. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
t (Cell Data Type)	An enumeration representing the cell's data type. The possible values for this attribute are defined by the ST_CellType simple type (§3.18.12).
vm (Value Metadata Index)	The zero-based index of the value metadata record associated with this cell's value. Metadata records are stored in the Metadata Part. Value metadata is extra information stored at the cell level, but associated with the value rather than the cell itself. Value metadata is accessible via formula reference. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Cell">
   <sequence>
      <element name="f" type="CT_CellFormula" minOccurs="0" maxOccurs="1"/>
      <element name="v" type="ST_Xstring" minOccurs="0" maxOccurs="1"/>
      <element name="is" type="CT_Rst" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="r" type="ST_CellRef" use="optional"/>
   <attribute name="s" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="t" type="ST_CellType" use="optional" default="n"/>
   <attribute name="cm" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="vm" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="ph" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.11.1.4 ndxf (New Formatting Information)

This element represents new differential formatting information for this cell. This formatting is applied to the existing formatting of the cell.

Parent Elements
rcc (§3.11.1.9)

Child Elements	Subclause
alignment (Alignment)	§3.8.1
border (Border)	§3.8.4
extLst (Future Feature Data Storage Area)	§3.2.10
fill (Fill)	§3.8.19
font (Font)	§3.8.21
numFmt (Number Format)	§3.8.30
protection (Protection Properties)	§3.8.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Dxf">
   <sequence>
      <element name="font" type="CT_Font" minOccurs="0" maxOccurs="1"/>
      <element name="numFmt" type="CT_NumFmt" minOccurs="0" maxOccurs="1"/>
      <element name="fill" type="CT_Fill" minOccurs="0" maxOccurs="1"/>
      <element name="alignment" type="CT_CellAlignment" minOccurs="0" maxOccurs="1"/>
      <element name="border" type="CT_Border" minOccurs="0" maxOccurs="1"/>
      <element name="protection" type="CT_CellProtection" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

3.11.1.5 oc (Old Cell Data)

This element represents old cell data. Old cell data is data that was previously stored in the cell.

For most spreadsheet application purposes, only the data type and reference will need to be used for revision tracking purposes. The rest of the cell properties can be written out, but are not necessarily needed as they can be recorded in other areas of the spreadsheet. For instance the <rfmt> element can be used to record style information instead of the `S (style index)` attribute.

Parent Elements
rcc (§3.11.1.9)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
f (Formula)	§3.3.1.37
is (Rich Text Inline)	§3.3.1.50
v (Cell Value)	§3.3.1.93

Attributes	Description
cm (Cell Metadata Index)	The zero-based index of the cell metadata record associated with this cell. Metadata information is found in the Metadata Part. Cell metadata is extra information stored at the cell level, and is attached to the cell (travels through moves, copy / paste, clear, etc). Cell metadata is not accessible via formula reference. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
ph (Show Phonetic)	A Boolean value indicating if the spreadsheet application should show phonetic information. Phonetic information is displayed in the same cell across the top of the cell and serves as a 'hint' which indicates how the text should be pronounced. This should only be used for East Asian languages. The possible values for this attribute are defined by the XML Schema boolean datatype.
r (Reference)	An A1 style reference to the location of this cell The possible values for this attribute are defined by the ST_CellRef simple type (§3.18.8).
s (Style Index)	The index of this cell's style. Style records are stored in the Styles Part. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
t (Cell Data Type)	An enumeration representing the cell's data type.

Attributes	Description
	The possible values for this attribute are defined by the ST_CellType simple type (§3.18.12).
vm (Value Metadata Index)	The zero-based index of the value metadata record associated with this cell's value. Metadata records are stored in the Metadata Part. Value metadata is extra information stored at the cell level, but associated with the value rather than the cell itself. Value metadata is accessible via formula reference. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Cell">
   <sequence>
      <element name="f" type="CT_CellFormula" minOccurs="0" maxOccurs="1"/>
      <element name="v" type="ST_Xstring" minOccurs="0" maxOccurs="1"/>
      <element name="is" type="CT_Rst" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="r" type="ST_CellRef" use="optional"/>
   <attribute name="s" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="t" type="ST_CellType" use="optional" default="n"/>
   <attribute name="cm" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="vm" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="ph" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.11.1.6 odxf (Old Formatting Information)

This element represents the old differential formatting information for this cell. Old differential formatting is differential formatting that was previously applied to the cell.

Parent Elements
rcc (§3.11.1.9)

Child Elements	Subclause
alignment (Alignment)	§3.8.1
border (Border)	§3.8.4
extLst (Future Feature Data Storage Area)	§3.2.10
fill (Fill)	§3.8.19
font (Font)	§3.8.21
numFmt (Number Format)	§3.8.30
protection (Protection Properties)	§3.8.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Dxf">
   <sequence>
      <element name="font" type="CT_Font" minOccurs="0" maxOccurs="1"/>
      <element name="numFmt" type="CT_NumFmt" minOccurs="0" maxOccurs="1"/>
      <element name="fill" type="CT_Fill" minOccurs="0" maxOccurs="1"/>
      <element name="alignment" type="CT_CellAlignment" minOccurs="0" maxOccurs="1"/>
      <element name="border" type="CT_Border" minOccurs="0" maxOccurs="1"/>
      <element name="protection" type="CT_CellProtection" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

3.11.1.7 oldFormula (Old Formula)

This element represents the old formula for a defined name in this cell. This is only used for named cells. Formulas that are entered in a cell with no name are represented by the formula element `<f>`.

The possible values for this element are defined by the ST_Formula simple type (§3.18.36).

Parent Elements
rdn (§3.11.1.13)

3.11.1.8 raf (Revision AutoFormat)

This element represents a revision record of auto formatting change information for a table.

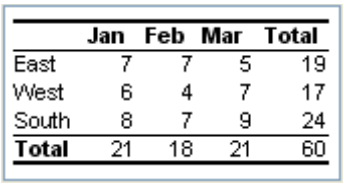
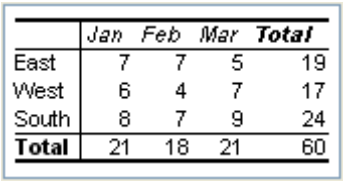
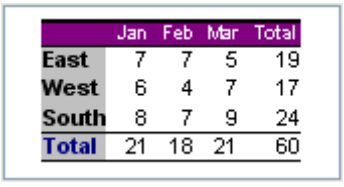
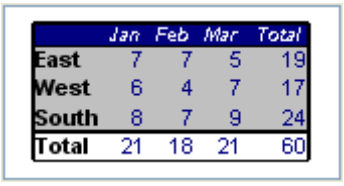
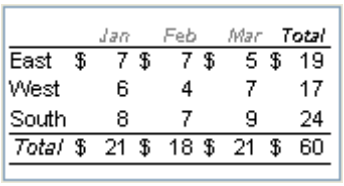
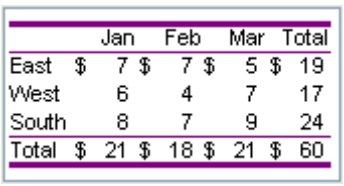
Parent Elements
revisions (§3.11.1.16)

Attributes	Description
applyAlignmentFormats (Apply Alignment Formats)	If `true` apply legacy table autoformat alignment properties. The possible values for this attribute are defined by the XML Schema boolean datatype.
applyBorderFormats (Apply Border Formats)	If `true` apply legacy table autoformat border properties. The possible values for this attribute are defined by the XML Schema boolean datatype.
applyFontFormats (Apply Font Formats)	If `true` apply legacy table autoformat font properties. The possible values for this attribute are defined by the XML Schema boolean datatype.
applyNumberFormats (Apply Number Formats)	If `true` apply legacy table autoformat number format properties. The possible values for this attribute are defined by the XML Schema boolean datatype.
applyPatternFormats (Apply Pattern Formats)	If `true` apply legacy table autoformat pattern properties. The possible values for this attribute are defined by the XML Schema boolean datatype.

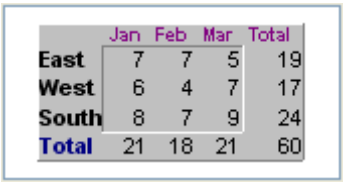
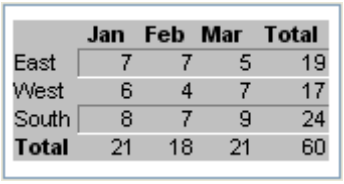
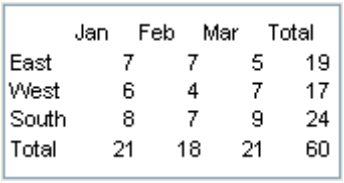
<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>applyWidthHeightFormats (Apply Width / Height Formats)</td><td>If <code>true</code> apply legacy table autoformat width/height properties.

The possible values for this attribute are defined by the XML Schema boolean datatype.</td></tr>
<tr><td>autoFormatId (Auto Format Id)</td><td>Identifies which legacy table autoformat to apply.

Here are representations of the supported table autoformats:

<table>
<tr><th>autoFormatId</th><th>Description</th></tr>
<tr><td>0</td><td> 
</td></tr>
<tr><td>1</td><td> 
</td></tr>
<tr><td>2</td><td> 
</td></tr>
<tr><td>3</td><td> 
</td></tr>
<tr><td>4</td><td> 
</td></tr>
<tr><td>5</td><td> 
</td></tr>
</table></td></tr>
</table>

Attributes	Description
	6
	7
	8
	9
	10
	11
	12
	13

Attributes	Description
	14 
	15 
	16 
	The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
ref (Reference)	A-1 style reference to the location where the formatting was applied The possible values for this attribute are defined by the ST_Ref simple type (§3.18.64).
sheetId (Sheet Id)	An integer representing the internal id of the sheet on which the revision occurred. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RevisionAutoFormatting">
   <attribute name="sheetId" type="xsd:unsignedInt" use="required"/>
   <attributeGroup ref="AG_AutoFormat"/>
   <attribute name="ref" type="ST_Ref" use="required"/>
</complexType>
```

3.11.1.9 rcc (Revision Cell Change)

This element stores information about the contents of the cell that was replaced.

Parent Elements
revisions (§3.11.1.16); rm (§3.11.1.19); rrc (§3.11.1.21)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10

Child Elements	Subclause
nc (New Cell Data)	§3.11.1.3
ndxf (New Formatting Information)	§3.11.1.4
oc (Old Cell Data)	§3.11.1.5
odxf (Old Formatting Information)	§3.11.1.6

Attributes	Description
dxf (Formatting)	A Boolean flag indicating that there was a differential formatting change for this cell - true if there was a formatting change, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.
endOfListFormulaUpdate (End of List Formula Update)	A Boolean flag indicating that indicates that the formula used at the end of a list has been updated. True if the formula was updated, false otherwise. Note: List in this context does not mean table, rather it refers to the feature where the spreadsheet application automatically creates an internal structure for making data input more consistent on adjacent rows or columns. For instance, if 3 cells in a row are entered with the same format, then when entering data into the 4th adjacent cell, the spreadsheet application might automatically apply that same format. In this case, those cells are treated as a list. The possible values for this attribute are defined by the XML Schema boolean datatype.
numFmtId (Number Format Id)	Zero-based index of the number format (Fmt) record used by this cell format (XF). The possible values for this attribute are defined by the ST_NumFmtId simple type (§3.18.49).
odxf (Old Formatting)	Flag indicating that there is old formatting information available for this cell. The possible values for this attribute are defined by the XML Schema boolean datatype.
oldPh (Old Phonetic Text)	A Boolean flag indicating whether there is old phonetic text information available. True when there is old phonetic text information available, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.
oldQuotePrefix (Old Quote Prefix)	A Boolean value indicating if a single quote prefix is was used on this cell previously. Single quote prefixes are used to cause a formula to be evaluated as a string. True if a single quote prefix was used previously, false otherwise The possible values for this attribute are defined by the XML Schema boolean datatype.
ph (Phonetic Text)	A Boolean flag indicating whether this cell contains phonetic text or not. True when the cell contains phonetic text, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description
quotePrefix (Quote Prefix)	A Boolean value indicating if a single quote prefix is used. Single quote prefixes are used to cause a formula to be evaluated as a string. True if a single quote prefix is used, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.
ra (Revision Undo Rejected)	A Boolean flag which indicates that this revision was due to a previous undo (ua) revision being rejected. The possible values for this attribute are defined by the XML Schema boolean datatype.
rId (Revision Id)	An integer representing the number of this revision. This id shall apply to reviewable revision types only. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
s (Style Revision)	Flag indicating that formatting change for this cell affected the cell's style. (Only applicable for Undo operations) The possible values for this attribute are defined by the XML Schema boolean datatype.
sId (Sheet Id)	Internal identifier of the sheet on which the revision occurred. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
ua (Revision From Rejection)	A Boolean flag indicating that this revision occurred because another revision was rejected and therefore undone. The possible values for this attribute are defined by the XML Schema boolean datatype.
xfDxf (Row Column Formatting Change)	Flag indicating that the formatting change had an effect on the formatting of the entire row or column that this cell belongs to. (Only applicable for Undo operations). The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RevisionCellChange">
   <sequence>
      <element name="oc" type="CT_Cell" minOccurs="0" maxOccurs="1"/>
      <element name="nc" type="CT_Cell" minOccurs="1" maxOccurs="1"/>
      <element name="odxf" type="CT_Dxf" minOccurs="0" maxOccurs="1"/>
      <element name="ndxf" type="CT_Dxf" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attributeGroup ref="AG_RevData"/>
   <attribute name="sId" type="xsd:unsignedInt" use="required"/>
   <attribute name="odxf" type="xsd:boolean" default="false"/>
   <attribute name="xfDxf" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="s" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="dxf" type="xsd:boolean" default="false"/>
   <attribute name="numFmtId" type="ST_NumFmtId" use="optional"/>
   <attribute name="quotePrefix" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="oldQuotePrefix" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="ph" type="xsd:boolean" default="false"/>
   <attribute name="oldPh" type="xsd:boolean" default="false"/>
   <attribute name="endOfListFormulaUpdate" type="xsd:boolean" default="false"/>
</complexType>
```

3.11.1.10 rcft (Revision Merge Conflict)

This element represents a revision record which indicates that there was a merge conflict.

Parent Elements
revisions (§3.11.1.16)

Attributes	Description
ra (Revision Undo Rejected)	A Boolean flag which indicates that this revision was due to a previous undo (ua) revision being rejected. The possible values for this attribute are defined by the XML Schema boolean datatype.
rId (Revision Id)	An integer representing the number of this revision. This id shall apply to reviewable revision types only. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
sheetId (Sheet Id)	An integer representing the internal id of the sheet on which the revision occurred. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
ua (Revision From Rejection)	A Boolean flag indicating that this revision occurred because another revision was rejected and therefore undone.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RevisionConflict">
   <attributeGroup ref="AG_RevData"/>
   <attribute name="sheetId" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.11.1.11 rcmt (Revision Cell Comment)

This element represents a revision record of a cell comment change.

Parent Elements
revisions (§3.11.1.16)

Attributes	Description
action (User Action)	An enumeration identifying what kind of an operation the user performed on the comment. The possible values for this attribute are defined by the ST_RevisionAction simple type (§3.18.67).
alwaysShow (Always Show Comment)	A Boolean value indicating that the user has set this comment to always be visible. True if the comment is set to always be visible, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.
author (Author)	A string representing the name of the author who changed this comment. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
cell (Cell)	An A-1 style reference to the cell where the comment was changed. The possible values for this attribute are defined by the ST_CellRef simple type (§3.18.8).
guid (GUID)	A globally unique identifier of this comment. The possible values for this attribute are defined by the ST_Guid simple type (§3.18.41).
hiddenColumn (Hidden Column)	A Boolean value indicating that the comment belongs to a cell in a hidden column. True if the comment is in a hidden column, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.
hiddenRow (Comment In Hidden Row)	A Boolean value indicating that the comment belongs to a cell in a hidden row. True if the comment is in a hidden row, false otherwise.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema boolean datatype.
newLength (New Comment Length)	Length of the comment text added in this revision. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
old (Old Comment)	An ignorable Boolean value used for backwards compatibility that indicates that the original comment was created by a legacy spreadsheet application. The possible values for this attribute are defined by the XML Schema boolean datatype.
oldLength (Original Comment Length)	Length of the comment before this revision was made. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
sheetId (Sheet Id)	An integer representing the internal id of the sheet on which the revision occurred. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RevisionComment">
   <attribute name="sheetId" type="xsd:unsignedInt" use="required"/>
   <attribute name="cell" type="ST_CellRef" use="required"/>
   <attribute name="guid" type="ST_Guid" use="required"/>
   <attribute name="action" type="ST_RevisionAction" default="add"/>
   <attribute name="alwaysShow" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="old" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="hiddenRow" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="hiddenColumn" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="author" type="ST_Xstring" use="required"/>
   <attribute name="oldLength" type="xsd:unsignedInt" default="0"/>
   <attribute name="newLength" type="xsd:unsignedInt" default="0"/>
</complexType>
```

3.11.1.12 rcv (Revision Custom View)

This element represents a revision record of adding or removing a custom view to the workbook

Parent Elements
revisions (§3.11.1.16)

Attributes	Description
action (User Action)	An enumeration representing the type of action that the user performed. The possible values for this attribute are defined by the ST_RevisionAction simple type

Attributes	Description
	(§3.18.67).
guid (GUID)	A globally unique identifier of the custom view. The possible values for this attribute are defined by the ST_Guid simple type (§3.18.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RevisionCustomView">
   <attribute name="guid" type="ST_Guid" use="required"/>
   <attribute name="action" type="ST_RevisionAction" use="required"/>
</complexType>
```

3.11.1.13 rdn (Revision Defined Name)

This element represents a revision record of a defined name change.

Parent Elements
revisions (§3.11.1.16)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
formula (Formula)	§3.3.1.40
oldFormula (Old Formula)	§3.11.1.7

Attributes	Description
comment (Name Comment)	A string representing a comment about the defined name. Note: This comment can be shown by the spreadsheet application in a names management UI so that users have more information about what the defined name is used for. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
customMenu (New Custom Menu)	A string representing the new custom menu text The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
customView (Custom View)	A Boolean flag indicating that this named range belongs to a custom view The possible values for this attribute are defined by the XML Schema boolean datatype.
description (Description)	A string representing the new description text for the defined name.

Attributes	Description
	The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
function (Function)	A Boolean value indicating that the defined name refers to a function. True if the defined name is a function, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.
functionGroupId (Function Group Id)	Represents the new function group id. Function group ids are used to help classify functions. For instance, functions in the same group can be searched or selected easily from the spreadsheet applications UI. For instance, filtering the list of all functions to allow the user to choose from functions used for financial data. The following group ids should be used: ID Function group 1 Financial 2 Date and Time 3 Math and Trig 4 Statistical 5 Lookup & Reference 6 Database 7 Text 8 Logical 9 Information 10 Commands 11 Customizing 12 Macro Control 13 DDE/External 14 User Defined 15 Engineering 14 Cube The possible values for this attribute are defined by the XML Schema unsignedByte datatype.
help (New Help Topic)	A string representing the new help topic text. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
hidden (Named Range Hidden)	A Boolean value indicating whether the named range is now hidden. Note: hidden refers to whether the defined name is of a 'hidden' type. This applies to things like a custom filter on a cell, it has a name, but is hidden and so is not visible in any name management UI. The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description
localSheetId (Local Name Sheet Id)	An integer representing the id of the sheet to which this defined name belongs. This shall be used local defined names only. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
name (Name)	A string representing the name for this defined name. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
oldComment (Old Name Comment)	A string representing the old comment about the defined name. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
oldCustomMenu (Old Custom Menu Text)	A string representing the old custom menu text The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
oldDescription (Old Description)	A string representing the old description text The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
oldFunction (Old Function)	A Boolean flag indicating that the old name was a function The possible values for this attribute are defined by the XML Schema boolean datatype.
oldFunctionGroupId (Old Function Group Id)	Old function group ID. The possible values for this attribute are defined by the XML Schema unsignedByte datatype.
oldHelp (Old Help Topic)	A string representing the old help topic text The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
oldHidden (Old Hidden)	A Boolean flag indicating whether the named range was hidden Note: hidden refers to whether the defined name is of a 'hidden' type. This applies to things like a custom filter on a cell, it has a name, but is hidden and so is not visible in any name management UI. The possible values for this attribute are defined by the XML Schema boolean datatype.
oldShortcutKey (Old Short Cut Key)	Old keyboard shortcut. The possible values for this attribute are defined by the XML Schema unsignedByte datatype.

Attributes	Description
oldStatusBar (Old Status Bar)	A string representing the old status bar text The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
ra (Revision Undo Rejected)	A Boolean flag which indicates that this revision was due to a previous undo (ua) revision being rejected. The possible values for this attribute are defined by the XML Schema boolean datatype.
rId (Revision Id)	An integer representing the number of this revision. This id shall apply to reviewable revision types only. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
shortcutKey (Shortcut Key)	Represents the new keyboard shortcut. The possible values for this attribute are defined by the XML Schema unsignedByte datatype.
statusBar (Status Bar)	A string representing the new status bar text. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
ua (Revision From Rejection)	A Boolean flag indicating that this revision occurred because another revision was rejected and therefore undone. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RevisionDefinedName">
   <sequence>
      <element name="formula" type="ST_Formula" minOccurs="0" maxOccurs="1"/>
      <element name="oldFormula" type="ST_Formula" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attributeGroup ref="AG_RevData"/>
   <attribute name="localSheetId" type="xsd:unsignedInt" use="optional"/>
   <attribute name="customView" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="name" type="ST_Xstring" use="required"/>
   <attribute name="function" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="oldFunction" type="xsd:boolean" default="false"/>
   <attribute name="functionGroupId" type="xsd:unsignedByte" use="optional"/>
   <attribute name="oldFunctionGroupId" type="xsd:unsignedByte" use="optional"/>
   <attribute name="shortcutKey" type="xsd:unsignedByte" use="optional"/>
   <attribute name="oldShortcutKey" type="xsd:unsignedByte" use="optional"/>
   <attribute name="hidden" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="oldHidden" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="customMenu" type="ST_Xstring" use="optional"/>
   <attribute name="oldCustomMenu" type="ST_Xstring" use="optional"/>
   <attribute name="description" type="ST_Xstring" use="optional"/>
   <attribute name="oldDescription" type="ST_Xstring" use="optional"/>
   <attribute name="help" type="ST_Xstring" use="optional"/>
   <attribute name="oldHelp" type="ST_Xstring" use="optional"/>
   <attribute name="statusBar" type="ST_Xstring" use="optional"/>
   <attribute name="oldStatusBar" type="ST_Xstring" use="optional"/>
   <attribute name="comment" type="ST_Xstring" use="optional"/>
   <attribute name="oldComment" type="ST_Xstring" use="optional"/>
</complexType>
```

3.11.1.14 reviewed (Reviewed)

This element represents an identifier of a single reviewed revision. A reviewed revision, is a revision that has been reviewed via the spreadsheet application's track changes feature, has been accepted, and has been saved.

Parent Elements
reviewedList (§3.11.1.15)

Attributes	Description
rId (revision Id)	ID of a reviewed revision. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Reviewed">
   <attribute name="rId" type="xsd:unsignedInt" use="required"/>
</complexType>
```

3.11.1.15 reviewedList (Reviewed List)

This element maintains a list of reviewed revisions.

Parent Elements
header (§3.11.1.1)

Child Elements	Subclause
reviewed (Reviewed)	§3.11.1.14

Attributes	Description
count (Reviewed Revisions Count)	Number of reviewed revisions. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ReviewedRevisions">
   <sequence>
      <element name="reviewed" type="CT_Reviewed" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.11.1.16 revisions (Revisions)

This element represents the root node of a list of revisions made in this shared workbook. This root node shows up at the beginning of every log file that contains specific revisions made to the workbook.

When multiple users are sharing, and editing, a workbook at the same time, there may be conflicting changes. The spreadsheet application should have logic to resolve such conflicts, and the file format should only contain enough information so that the spreadsheet application can restore the workbook to the correct state after conflict resolution. Revisions can also be tracked by the spreadsheet application for review by the user at a later time (as opposed to only dealing with conflicts on a save event.) Some edits to workbooks will be made as a result of this conflict resolution. So, there will be cases where a revision is effectively undone by another user, and as a result that undoing is itself a revision that adds or changes data in the file. These types of operations are tracked by the ua and ra attributes of many different elements.

[*Example*:

Step 1:

User 1 inserts Column A. So the XML in the revision log would look like this:

```
<revisions xmlns="…" xmlns:r="…">
  <rrc rId="1" sId="1" ref="A1:A1048576" action="insertCol"/>
</revisions>
```

Step 2:

User 2 synchronizes the file to pick up that change, but then activates the Track Changes feature, and rejects that change. This effectively performs an undo on User 1's insertion. This is denoted in the file with the ua attribute meaning that this change happened as the result of an undo. The XML for the revision log would look like this:

```
<revisions xmlns="…" xmlns:r="…">
  <rrc rId="2" ua="1" sId="1" ref="A1:A1048576" action="deleteCol"/>
  <rcft rId="1" ua="1" sheetId="1"/>
</revisions>
```

Step 3:

User 1 types "foo" in A1, and saves the file. A conflict resolution dialog is shown since User 2's version of the file removed the inserted Column A. User 1 chooses to accept their own changes. This undoes the change that User 2 made. So, in effect, it performed an undo on a previous undo operation. This is denoted in the file format by the ra attribute meaning that a the change occurred because a previous undo was undone. So the resulting XML for the newest log file looks like this:

```
<revisions xmlns="…" xmlns:r="…">
  <rrc rId="3" ua="1" ra="1" sId="1" ref="A1:A1048576" action="insertCol"/>
  <rcft rId="2" ua="1" sheetId="1"/>
  <rcc rId="4" sId="1">
    <nc r="A1" t="inlineStr">
      <is>
        <t>foo</t>
      </is>
    </nc>
  </rcc>
  <rcft rId="2" sheetId="1"/>
</revisions>
```

end example]

Parent Elements
Root element of SpreadsheetML Shared Workbook Revision Log part

Child Elements	Subclause
raf (Revision AutoFormat)	§3.11.1.8

Child Elements	Subclause
rcc (Revision Cell Change)	§3.11.1.9
rcft (Revision Merge Conflict)	§3.11.1.10
rcmt (Revision Cell Comment)	§3.11.1.11
rcv (Revision Custom View)	§3.11.1.12
rdn (Revision Defined Name)	§3.11.1.13
rfmt (Revision Format)	§3.11.1.17
ris (Revision Insert Sheet)	§3.11.1.18
rm (Revision Cell Move)	§3.11.1.19
rqt (Revision Query Table)	§3.11.1.20
rrc (Revision Row Column Insert Delete)	§3.11.1.21
rsnm (Revision Sheet Name)	§3.11.1.22

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Revisions">
   <choice maxOccurs="unbounded">
      <element name="rrc" type="CT_RevisionRowColumn" minOccurs="0" maxOccurs="unbounded"/>
      <element name="rm" type="CT_RevisionMove" minOccurs="0" maxOccurs="unbounded"/>
      <element name="rcv" type="CT_RevisionCustomView" minOccurs="0" maxOccurs="unbounded"/>
      <element name="rsnm" type="CT_RevisionSheetRename" minOccurs="0" maxOccurs="unbounded"/>
      <element name="ris" type="CT_RevisionInsertSheet" minOccurs="0" maxOccurs="unbounded"/>
      <element name="rcc" type="CT_RevisionCellChange" minOccurs="0" maxOccurs="unbounded"/>
      <element name="rfmt" type="CT_RevisionFormatting" minOccurs="0" maxOccurs="unbounded"/>
      <element name="raf" type="CT_RevisionAutoFormatting" minOccurs="0" maxOccurs="unbounded"/>
      <element name="rdn" type="CT_RevisionDefinedName" minOccurs="0" maxOccurs="unbounded"/>
      <element name="rcmt" type="CT_RevisionComment" minOccurs="0" maxOccurs="unbounded"/>
      <element name="rqt" type="CT_RevisionQueryTableField" minOccurs="0" maxOccurs="unbounded"/>
      <element name="rcft" type="CT_RevisionConflict" minOccurs="0" maxOccurs="unbounded"/>
   </choice>
</complexType>
```

3.11.1.17 rfmt (Revision Format)

This element represents a revision record of information about a formatting change.

Parent Elements
revisions (§3.11.1.16); rm (§3.11.1.19); rrc (§3.11.1.21)

Child Elements	Subclause
dxf (Formatting)	§3.8.14
extLst (Future Feature Data Storage Area)	§3.2.10

Attributes	Description
length (Length)	The number of characters that were affected by a string change, counting from `start`. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
s (Style)	Flag indicating that this formatting change affected a cell's style. (Only applicable for Undo operations). The possible values for this attribute are defined by the XML Schema boolean datatype.
sheetId (Sheet Id)	An integer representing the internal id of the sheet on which the revision occurred. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
sqref (Sequence Of References)	A worksheet range to which this formatting was applied. [*Note*: For applications supporting the default grid size (see §3.17.5), full column and row references shall explicitly state the row and column components, e.g., "A1:A1048576" For column "A", and A1:XFD1 for row "1". Applications with larger grid sizes shall interpret these to mean "column A" and "row 1" respectively, for their larger grid size. *end note*] The possible values for this attribute are defined by the ST_Sqref simple type (§3.18.78).
start (Start index)	An integer representing an index showing which character a string change starts at within the string in the cell. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
xfDxf (Row or Column Formatting Change)	A Boolean flag indicating that this formatting change had an affect on the formatting of an entire row or column that an affected cell(s) belongs to. (Only applicable for Undo operations) The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RevisionFormatting">
   <sequence>
      <element name="dxf" type="CT_Dxf" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="sheetId" type="xsd:unsignedInt" use="required"/>
   <attribute name="xfDxf" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="s" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="sqref" type="ST_Sqref" use="required"/>
   <attribute name="start" type="xsd:unsignedInt" use="optional"/>
   <attribute name="length" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.11.1.18 ris (Revision Insert Sheet)

This element represents a revision record of a sheet that was inserted.

Parent Elements
revisions (§3.11.1.16)

Attributes	Description
name (Sheet Name)	The name of the new sheet. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
ra (Revision Undo Rejected)	A Boolean flag which indicates that this revision was due to a previous undo (ua) revision being rejected. The possible values for this attribute are defined by the XML Schema boolean datatype.
rId (Revision Id)	An integer representing the number of this revision. This id shall apply to reviewable revision types only. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
sheetId (Sheet Id)	An integer representing the internal id of the sheet on which the revision occurred The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
sheetPosition (Sheet Position)	An integer representing the zero based position of the new sheet in the sheet tab bar. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
ua (Revision From Rejection)	A Boolean flag indicating that this revision occurred because another revision was rejected and therefore undone. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RevisionInsertSheet">
   <attributeGroup ref="AG_RevData"/>
   <attribute name="sheetId" type="xsd:unsignedInt" use="required"/>
   <attribute name="name" type="ST_Xstring" use="required"/>
   <attribute name="sheetPosition" type="xsd:unsignedInt" use="required"/>
</complexType>
```

3.11.1.19 rm (Revision Cell Move)

This element represents a revision record on a cell(s) that moved.

Parent Elements
revisions (§3.11.1.16)

Child Elements	Subclause
rcc (Revision Cell Change)	§3.11.1.9
rfmt (Revision Format)	§3.11.1.17
undo (Undo)	§3.11.1.25

Attributes	Description
destination (Destination)	New A1 style location of the cell(s) that were moved The possible values for this attribute are defined by the ST_Ref simple type (§3.18.64).
ra (Revision Undo Rejected)	A Boolean flag which indicates that this revision was due to a previous undo (ua) revision being rejected. The possible values for this attribute are defined by the XML Schema boolean datatype.
rId (Revision Id)	An integer representing the number of this revision. This id shall apply to reviewable revision types only. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
sheetId (Sheet Id)	An integer representing the internal id of the sheet on which the revision occurred. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
source (Source)	The original A1 style location of the cell(s) that were moved The possible values for this attribute are defined by the ST_Ref simple type (§3.18.64).
sourceSheetId (Source Sheet Id)	An integer representing the internal id of the sheet where the cell(s) originally resided. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
ua (Revision From Rejection)	A Boolean flag indicating that this revision occurred because another revision was rejected and therefore undone. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RevisionMove">
    <choice minOccurs="0" maxOccurs="unbounded">
        <element name="undo" type="CT_UndoInfo" minOccurs="0" maxOccurs="unbounded"/>
        <element name="rcc" type="CT_RevisionCellChange" minOccurs="0" maxOccurs="unbounded"/>
        <element name="rfmt" type="CT_RevisionFormatting" minOccurs="0" maxOccurs="unbounded"/>
    </choice>
    <attributeGroup ref="AG_RevData"/>
    <attribute name="sheetId" type="xsd:unsignedInt" use="required"/>
    <attribute name="source" type="ST_Ref" use="required"/>
    <attribute name="destination" type="ST_Ref" use="required"/>
    <attribute name="sourceSheetId" type="xsd:unsignedInt" use="optional" default="0"/>
</complexType>
```

3.11.1.20 rqt (Revision Query Table)

This element represents a revision record of a query table field change.

Parent Elements
revisions (§3.11.1.16)

Attributes	Description
fieldId (Field Id)	ID of the specific query table field that was removed. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
ref (QueryTable Reference)	Location of the affected query table. The possible values for this attribute are defined by the ST_Ref simple type (§3.18.64).
sheetId (Sheet Id)	An integer representing the internal id of the sheet on which the revision occurred. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RevisionQueryTableField">
    <attribute name="sheetId" type="xsd:unsignedInt" use="required"/>
    <attribute name="ref" type="ST_Ref" use="required"/>
    <attribute name="fieldId" type="xsd:unsignedInt" use="required"/>
</complexType>
```

3.11.1.21 rrc (Revision Row Column Insert Delete)

This element represents a revision record of a row/column insert/delete action.

Parent Elements
revisions (§3.11.1.16)

Child Elements	Subclause
rcc (Revision Cell Change)	§3.11.1.9
rfmt (Revision Format)	§3.11.1.17
undo (Undo)	§3.11.1.25

Attributes	Description
action (User Action)	Indicates what type of action the user just performed on the row or column. The possible values for this attribute are defined by the ST_rwColActionType simple type (§3.18.68).
edge (Edge Deleted)	A Boolean flag indicating that a row or column is being deleted at the edge of a sorted range (only applicable to a Delete Row/Column revision types). The possible values for this attribute are defined by the XML Schema boolean datatype.
eol (End Of List)	A Boolean flag indicating that a row or a column is being inserted at the end of a list of data. Note: List in this context does not mean table, rather it refers to the feature where the spreadsheet application automatically creates an internal structure for making data input more consistent on adjacent rows or columns. For instance, if 3 cells in a row are entered with the same format, then when entering data into the 4th adjacent cell, the spreadsheet application might automatically apply that same format. In this case, those cells are treated as a list. The possible values for this attribute are defined by the XML Schema boolean datatype.
ra (Revision Undo Rejected)	A Boolean flag which indicates that this revision was due to a previous undo (ua) revision being rejected. The possible values for this attribute are defined by the XML Schema boolean datatype.
ref (Reference)	A reference to the location of the rows/columns that were inserted or deleted. [*Note*: A reference to a whole column or row shall include both the column and row components. For example, column A is referenced by "A1:A1048576", and row 1 is referenced by "A1:XFD1". However, because this attribute value is occurring in the context of an entire row or column insert, the column component of a row reference can be ignored, and the row component of a column reference can be ignored. *end note*] The possible values for this attribute are defined by the ST_Ref simple type (§3.18.64).
rId (Revision Id)	An integer representing the number of this revision. This id shall apply to reviewable revision types only.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
sId (Sheet Id)	An integer representing the internal id of the sheet on which the revision occurred. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
ua (Revision From Rejection)	A Boolean flag indicating that this revision occurred because another revision was rejected and therefore undone. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RevisionRowColumn">
    <choice minOccurs="0" maxOccurs="unbounded">
        <element name="undo" type="CT_UndoInfo" minOccurs="0" maxOccurs="unbounded"/>
        <element name="rcc" type="CT_RevisionCellChange" minOccurs="0" maxOccurs="unbounded"/>
        <element name="rfmt" type="CT_RevisionFormatting" minOccurs="0" maxOccurs="unbounded"/>
    </choice>
    <attributeGroup ref="AG_RevData"/>
    <attribute name="sId" type="xsd:unsignedInt" use="required"/>
    <attribute name="eol" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="ref" type="ST_Ref" use="required"/>
    <attribute name="action" type="ST_rwColActionType" use="required"/>
    <attribute name="edge" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.11.1.22 rsnm (Revision Sheet Name)

This element represents a revision record tracking the renaming a sheet.

Parent Elements
revisions (§3.11.1.16)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10

Attributes	Description
newName (New Sheet Name)	A string representing the new sheet name The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
oldName (Old Sheet Name)	A string representing the old sheet name

Attributes	Description
	The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
ra (Revision Undo Rejected)	A Boolean flag which indicates that this revision was due to a previous undo (ua) revision being rejected. The possible values for this attribute are defined by the XML Schema boolean datatype.
rId (Revision Id)	An integer representing the number of this revision. This id shall apply to reviewable revision types only. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
sheetId (Sheet Id)	An integer representing the internal id of the sheet on which the revision occurred. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
ua (Revision From Rejection)	A Boolean flag indicating that this revision occurred because another revision was rejected and therefore undone. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RevisionSheetRename">
   <sequence>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
   <attributeGroup ref="AG_RevData"/>
   <attribute name="sheetId" type="xsd:unsignedInt" use="required"/>
   <attribute name="oldName" type="ST_Xstring" use="required"/>
   <attribute name="newName" type="ST_Xstring" use="required"/>
</complexType>
```

3.11.1.23 sheetId (Sheet Id)

This element represents a sheet that revision may take place on. Each sheet in the workbook should be represented by one of these elements, and each sheet has an id associated with it. Sheet ids are used to refer to sheets internally by the spreadsheet application.

Parent Elements
sheetIdMap (§3.11.1.24)

Attributes	Description
val (Sheet Id)	An integer serving as a number by which to reference the sheet internally.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SheetId">
   <attribute name="val" type="xsd:unsignedInt" use="required"/>
</complexType>
```

3.11.1.24 sheetIdMap (Sheet Id Map)

This element represents a list of sheets and corresponding ids that are used for tracking revision records.

Parent Elements
header (§3.11.1.1)

Child Elements	Subclause
sheetId (Sheet Id)	§3.11.1.23

Attributes	Description
count (Sheet Count)	Number of sheets. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SheetIdMap">
   <sequence>
      <element name="sheetId" type="CT_SheetId" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.11.1.25 undo (Undo)

This element represents undo information for row/column deletion when there are functions in the spreadsheet that reference the deleted rows/columns. This element is not applicable for insert revisions.

Parent Elements
rm (§3.11.1.19); rrc (§3.11.1.21)

Attributes	Description

Attributes	Description
array (Array Entered)	Flag indicating that the affected formula was array-entered. The possible values for this attribute are defined by the XML Schema boolean datatype.
cs (Cross Sheet Move)	A Boolean flag indicating this was a cross-sheet move. True if it was a cross sheet move, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.
dn (Defined Name)	Identifies the named range that referenced the deleted cell range. Mutually exclusive with the cell reference attribute. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
dr (Range)	The range which was deleted that is referenced by the affected formula. The possible values for this attribute are defined by the ST_RefA simple type (§3.18.65).
exp (Expression)	Identifies the type of the expression that should be adjusted in the corresponding formula. The possible values for this attribute are defined by the ST_FormulaExpression simple type (§3.18.37).
index (Index)	Index of the expression within the corresponding formula that was affected by this change. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
nf (Defined Name Formula)	A Boolean flag indicating that the corresponding formula is part of a defined name. True if this formula is part of a defined name, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.
r (Cell Reference)	Location of the cell whose formula referenced the deleted cell range. Mutually exclusive with the defined name attribute The possible values for this attribute are defined by the ST_CellRef simple type (§3.18.8).
ref3D (Reference 3D)	A Boolean flag indicating that the expression contained the sheet name in addition to the cell reference. True if it contained the sheet name, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.
sId (Sheet Id)	Internal Id of the worksheet that contained the formula that referenced the deleted cell range. Mutually exclusive with the defined name attribute. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

Attributes	Description
v (Value Needed)	A Boolean flag indicating the formula needs the actual value of the cell(s) it's referencing. True if the formula requires the value of the cell it references, false otherwise. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_UndoInfo">
    <attribute name="index" type="xsd:unsignedInt" use="required"/>
    <attribute name="exp" type="ST_FormulaExpression" use="required"/>
    <attribute name="ref3D" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="array" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="v" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="nf" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="cs" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="dr" type="ST_RefA" use="required"/>
    <attribute name="dn" type="ST_Xstring" use="optional"/>
    <attribute name="r" type="ST_CellRef" use="optional"/>
    <attribute name="sId" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.11.2 Shared Workbook User Data

This subclause specifies information about the users of a shared workbook.

3.11.2.1 userInfo (User Information)

This element represents a user, and it stores information about a specific user as it relates to revisions.

Parent Elements
users (§3.11.2.2)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10

Attributes	Description
dateTime (Date Time)	Date and time when this user opened the shared workbook. The possible values for this attribute are defined by the XML Schema dateTime datatype.
guid (User Revisions GUID)	A globally unique identifier identifying the last set of revisions that this uses is synchronized to. This attribute can be used by the spreadsheet application to ensure that revisions this user depends on aren't deleted.

Attributes	Description
	The possible values for this attribute are defined by the ST_Guid simple type (§3.18.41).
id (User Id)	An integer representing an internal user id for this user. Note: this number is allowed to be negative. The possible values for this attribute are defined by the XML Schema int datatype.
name (User Name)	Display name for this user [*Note*: User name strings should not be longer than 54 characters. *end note*] The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SharedUser">
    <sequence>
        <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
    </sequence>
    <attribute name="guid" type="ST_Guid" use="required"/>
    <attribute name="name" type="ST_Xstring" use="required"/>
    <attribute name="id" type="xsd:int" use="required"/>
    <attribute name="dateTime" type="xsd:dateTime" use="required"/>
</complexType>
```

3.11.2.2 users (User List)

This element represents a list of users who currently have this shared workbook open. This list does not include any users who have the workbook open in Read-Only mode.

Parent Elements
Root element of SpreadsheetML Shared Workbook User Data part

Child Elements	Subclause
userInfo (User Information)	§3.11.2.1

Attributes	Description
count (Active User Count)	Number of users who currently have this shared workbook open. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Users">
   <sequence>
      <element name="userInfo" minOccurs="0" maxOccurs="256" type="CT_SharedUser"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.12 QueryTable Data

Query tables are 2 dimensional tables of data bound to an external query of some kind. A query table could for example show specific data from a text file, from a web query, or from a database query.

[*Example*:

Data connectivity can use a number of different technologies. The following spreadsheetML fragment is one an example of a query table connected to a database:

```
<queryTable xmlns="…" name="Northwind Orders" rowNumbers="1"
  growShrinkType="overwriteClear" connectionId="1" autoFormatId="16"
  applyNumberFormats="0" applyBorderFormats="0" applyFontFormats="0"
  applyPatternFormats="0" applyAlignmentFormats="0" applyWidthHeightFormats="0">
  <queryTableRefresh nextId="15">
    <queryTableFields count="12">
      <queryTableField id="1" name="OrderID" tableColumnId="1"/>
      <queryTableField id="2" name="CustomerID" tableColumnId="2"/>
      <queryTableField id="3" name="EmployeeID" tableColumnId="3"/>
      <queryTableField id="4" name="OrderDate" tableColumnId="4"/>
      <queryTableField id="5" name="RequiredDate" tableColumnId="5"/>
      <queryTableField id="6" name="ShippedDate" tableColumnId="6"/>
      <queryTableField id="7" name="ShipName" tableColumnId="7"/>
      <queryTableField id="8" name="ShipAddress" tableColumnId="8"/>
      <queryTableField id="9" name="ShipCity" tableColumnId="9"/>
      <queryTableField id="10" name="ShipRegion" tableColumnId="10"/>
      <queryTableField id="11" name="ShipPostalCode" tableColumnId="11"/>
      <queryTableField id="12" name="ShipCountry" tableColumnId="12"/>
    </queryTableFields>
  </queryTableRefresh>
</queryTable>
```

end example]

And here's an example of the spreadsheetML fragment defining a query table connected to a text import:

[*Example*:

```
<queryTable xmlns="…" name="data in text" connectionId="1" autoFormatId="16"
  applyNumberFormats="0" applyBorderFormats="0" applyFontFormats="1"
  applyPatternFormats="1" applyAlignmentFormats="0"
  applyWidthHeightFormats="0"/>
```

end example]

Elsewhere in the spreadsheetML file, a connection element is defined with the name "Northwind Orders" that describes how to connect to the appropriate database to refresh data for the query table.

3.12.1 deletedField (Deleted Field)

This element specifies a field that has been deleted from the query table.

[*Example*:

```
<queryTableDeletedFields count="2">
  <deletedField name="ShipVia"/>
  <deletedField name="Freight"/>
</queryTableDeletedFields>
```

end example]

Parent Elements
queryTableDeletedFields (§3.12.3)

Attributes	Description
name (Deleted Fields Name)	Specifies the name of the deleted field. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

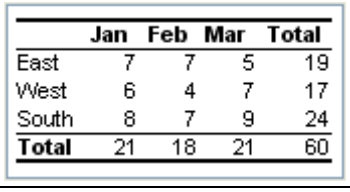
```
<complexType name="CT_DeletedField">
   <attribute name="name" type="ST_Xstring" use="required"/>
</complexType>
```

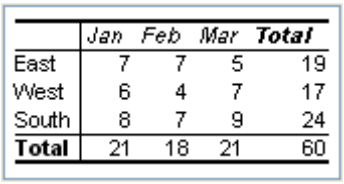
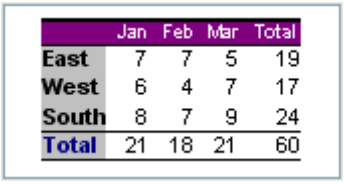
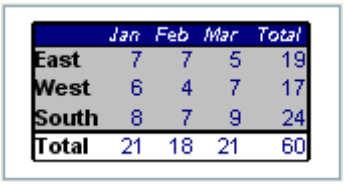
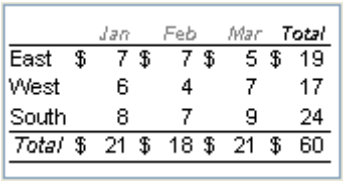
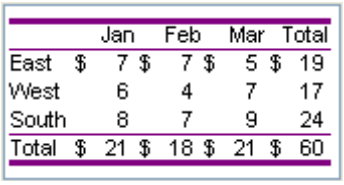
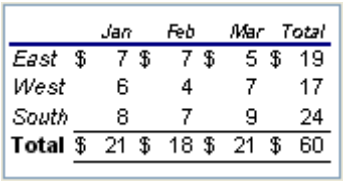
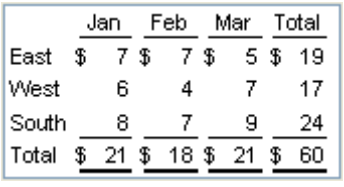
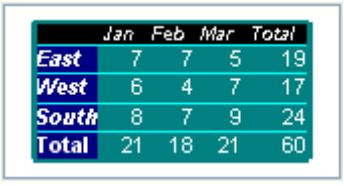
3.12.2 queryTable (Query Table)

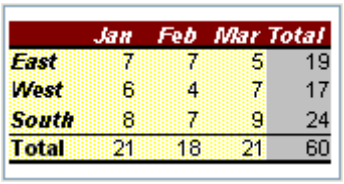
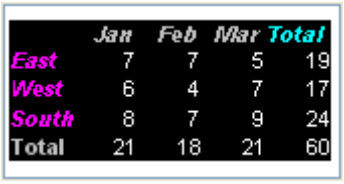
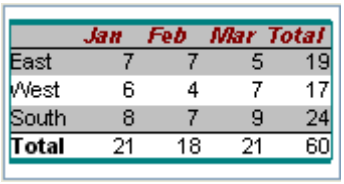
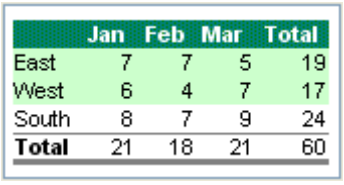
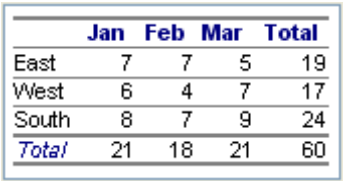
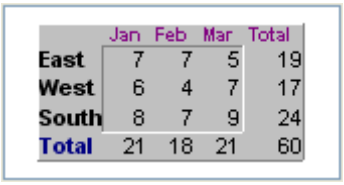
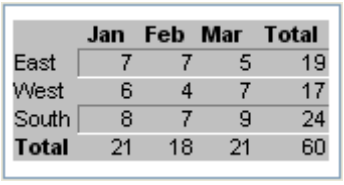
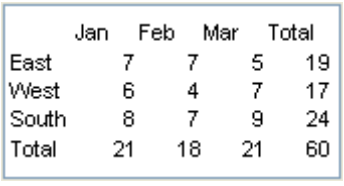
This element specifies all the relevant properties for a query table, one query table element is stored for each query table object in the spreadsheetML document.

Parent Elements
Root element of SpreadsheetML Query Table part

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
queryTableRefresh (QueryTable Refresh Information)	§3.12.6

Attributes	Description
adjustColumnWidth (Adjust Column Width On Refresh)	Specifies whether to automatically adjust column widths on refresh to fit the data retrieved. `true` if column widths should be adjusted. The possible values for this attribute are defined by the XML Schema boolean datatype.
applyAlignmentFormats (Apply Alignment Formats)	If `true` apply legacy table autoformat alignment properties. The possible values for this attribute are defined by the XML Schema boolean datatype.
applyBorderFormats (Apply Border Formats)	If `true` apply legacy table autoformat border properties. The possible values for this attribute are defined by the XML Schema boolean datatype.
applyFontFormats (Apply Font Formats)	If `true` apply legacy table autoformat font properties. The possible values for this attribute are defined by the XML Schema boolean datatype.
applyNumberFormats (Apply Number Formats)	If `true` apply legacy table autoformat number format properties. The possible values for this attribute are defined by the XML Schema boolean datatype.
applyPatternFormats (Apply Pattern Formats)	If `true` apply legacy table autoformat pattern properties. The possible values for this attribute are defined by the XML Schema boolean datatype.
applyWidthHeightFormats (Apply Width / Height Formats)	If `true` apply legacy table autoformat width/height properties. The possible values for this attribute are defined by the XML Schema boolean datatype.
autoFormatId (Auto Format Id)	Identifies which legacy table autoformat to apply. Here are representations of the supported table autoformats: autoFormatId: 0 Description: 

Attributes	Description
	1 
	2 
	3 
	4 
	5 
	6 
	7 
	8 

Attributes	Description
	9 
	10 
	11 
	12 
	13 
	14 
	15 
	16 

The possible values for this attribute are defined by the XML Schema unsignedInt

Attributes	Description
	datatype.
backgroundRefresh (Background Refresh)	Specifies whether or not the query table shall try to refresh data in the background. The possible values for this attribute are defined by the XML Schema boolean datatype.
connectionId (Connection Id)	Specifies the ID number of the external data connection to use to refresh data in the query table. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
disableEdit (Disable Edit)	Specifies whether the connection element used with this query table shall be editable. If `true`, then the connection is not editable. The possible values for this attribute are defined by the XML Schema boolean datatype.
disableRefresh (Disable Refresh)	Specifies whether the query table shall be refreshable. If `true`, then then query table is not refreshable. The possible values for this attribute are defined by the XML Schema boolean datatype.
fillFormulas (Fill Adjacent Formulas)	Specifies whether or not formulas in columns adjacent to the query table should be filled down whenever the query table is refreshed. This is helpful since the number of rows returned by a query table refresh operation can vary. The possible values for this attribute are defined by the XML Schema boolean datatype.
firstBackgroundRefresh (First Background Refresh)	Specifies whether or not data has ever been refreshed for this query table. If the very first background data refresh had not completed at the time the file was saved, this attribute will be set to `true`. The possible values for this attribute are defined by the XML Schema boolean datatype.
growShrinkType (Grow Shrink Type)	Specifies the type of behavior expected for dealing with a variable number of rows of data in the query table between refresh operations. The meaning of the possible values of this attribute {`insertClear`, `insertDelete`, `overwriteClear`} are explained in detail in the definition of the simple type. The possible values for this attribute are defined by the ST_GrowShrinkType simple type (§3.18.40).
headers (First Row Column Titles)	Specifies whether or not the query table has first row with column titles. The possible values for this attribute are defined by the XML Schema boolean datatype.
intermediate (Intermediate)	Specifies whether this query table is in an intermediate state, having been defined but not fully formed and populated with data. In this state, fields and ranges of the query table may be unknown.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema boolean datatype.
name (QueryTable Name)	Specifies the name of the query table. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
preserveFormatting (Preserve Formatting On Refresh)	Specifies whether the application should try to preserve formatting in the query table and copy this formatting to any new rows of data. The possible values for this attribute are defined by the XML Schema boolean datatype.
refreshOnLoad (Refresh On Load)	Specifies whether the query table shall refresh its data automatically when the spreadsheetML document is loaded or opened. The possible values for this attribute are defined by the XML Schema boolean datatype.
removeDataOnSave (Remove Data On Save)	Specifies whether the query table shall remove all data from the worksheet before the spreadsheetML document is saved. Note: this is very helpful for situations where people who have different permissions to view data want to share the same spreadsheetML document. All data from the last user will be removed, and new users re-query the external data sources with their own credentials. The possible values for this attribute are defined by the XML Schema boolean datatype.
rowNumbers (Row Numbers)	Specifies whether the query table shall include a first column of row numbers. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_QueryTable">
   <sequence>
      <element name="queryTableRefresh" type="CT_QueryTableRefresh" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="name" type="ST_Xstring" use="required"/>
   <attribute name="headers" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="rowNumbers" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="disableRefresh" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="backgroundRefresh" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="firstBackgroundRefresh" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="refreshOnLoad" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="growShrinkType" type="ST_GrowShrinkType" use="optional" default="insertDelete"/>
   <attribute name="fillFormulas" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="removeDataOnSave" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="disableEdit" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="preserveFormatting" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="adjustColumnWidth" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="intermediate" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="connectionId" type="xsd:unsignedInt" use="required"/>
   <attributeGroup ref="AG_AutoFormat"/>
</complexType>
```

3.12.3 queryTableDeletedFields (Deleted Fields)

This element is the collection for deletedField (§3.12.1) elements, each of which represents a column or field that has been deleted from the query table.

[*Example*:

```
<queryTableDeletedFields count="2">
  <deletedField name="ShipVia"/>
  <deletedField name="Freight"/>
</queryTableDeletedFields>
```

end example]

Parent Elements
queryTableRefresh (§3.12.6)

Child Elements	Subclause
deletedField (Deleted Field)	§3.12.1

Attributes	Description
count (Deleted	Specifies how many deleted fields there are.

Attributes	Description
Fields Count)	The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_QueryTableDeletedFields">
   <sequence>
      <element name="deletedField" type="CT_DeletedField" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.12.4 queryTableField (QueryTable Field)

This element holds the properties related to a specific field or column in a query table.

Parent Elements
queryTableFields (§3.12.5)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10

Attributes	Description
clipped (Clipped Column)	Specifies whether this field/column is currently clipped and thus not visible in the worksheet. Note: this state might occur for example when a query table is defined near the edge of a worksheet or other object in the spreadsheet that can't be overwritten with external data. In this case some of the fields are displayed, but not all of them. The possible values for this attribute are defined by the XML Schema boolean datatype.
dataBound (Data Bound Column)	Specifies whether this column is a user-defined column or comes from the external data query. User defined columns shall be preserved during data refresh operations. User-defined columns are only supported on query tables that are attached to table objects. The possible values for this attribute are defined by the XML Schema boolean datatype.
fillFormulas (Fill This Formula On Refresh)	Specifies whether the formula in this field/column should be filled down on data refresh. The possible values for this attribute are defined by the XML Schema boolean datatype.
id (Field Id)	Specifies the unique identifier of the query table field.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
name (Name)	Specifies the unique name of the query table field. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
rowNumbers (Row Numbers)	true if this column contains the row numbers for the records returned. The possible values for this attribute are defined by the XML Schema boolean datatype.
tableColumnId (Table Column Id)	Specifies the unique identifier for the table column if the query table is attached to a table object rather than just a range in the sheet. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_QueryTableField">
   <sequence minOccurs="0">
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="id" type="xsd:unsignedInt" use="required"/>
   <attribute name="name" type="ST_Xstring" use="optional"/>
   <attribute name="dataBound" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="rowNumbers" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="fillFormulas" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="clipped" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="tableColumnId" type="xsd:unsignedInt" default="0"/>
</complexType>
```

3.12.5 queryTableFields (Query table fields)

This element is the collection for queryTableField elements.

Parent Elements
queryTableRefresh (§3.12.6)

Child Elements	Subclause
queryTableField (QueryTable Field)	§3.12.4

Attributes	Description
count (Column Count)	Specifies the number of columns there are in this query table. Includes both query-defined and user-defined columns, but not deleted columns.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_QueryTableFields">
   <sequence>
      <element name="queryTableField" type="CT_QueryTableField" minOccurs="0"
        maxOccurs="unbounded"/>
   </sequence>
   <attribute name="count" type="xsd:unsignedInt" use="optional" default="0"/>
</complexType>
```

3.12.6 queryTableRefresh (QueryTable Refresh Information)

This element contains information related to refreshing the query table.

Parent Elements
queryTable (§3.12.2)

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
queryTableDeletedFields (Deleted Fields)	§3.12.3
queryTableFields (Query table fields)	§3.12.5
sortState (Sort State)	§3.3.1.89

Attributes	Description
fieldIdWrapped (Next Field Id Wrapped)	Whether or not the idFieldNext value wrapped around. The possible values for this attribute are defined by the XML Schema boolean datatype.
headersInLastRefresh (Headers In Last Refresh)	Whether or not the Query Table had titles last refresh. The possible values for this attribute are defined by the XML Schema boolean datatype.
minimumVersion (Minimum Refresh Version)	For backward compatibility with legacy versions of spreadsheet applications, this attribute specifies the minimum version of the application that is expected to correctly refresh the data in the query table without any problems. Note: if this attribute is specified, an earlier version of a spreadsheet application should alert the user to the potential incompatibilities when a refresh is attempted. The possible values for this attribute are defined by the XML Schema unsignedByte datatype.

Attributes	Description
nextId (Next field id)	Specifies the next unique queryTableField (§3.12.4) id number available for assignment. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
preserveSortFilterLayout (Preserve Sort & Filter Layout)	Specifies whether sorting, autofilter, layout, and table block formatting should be preserved for this query table across data refresh operations. Note: if this attribute is set to `false`, the query table might be more or less recreated from scratch when data is refreshed. In this case, all user deleted or rearranged columns, user inserted columns that aren't bound to external data, and table column formatting are discarded. The possible values for this attribute are defined by the XML Schema boolean datatype.
unboundColumnsLeft (Columns Left)	Specifies the number of extra columns included at the left end of the field array that aren't bound to external data. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
unboundColumnsRight (Columns Right)	Specifies the number of extra columns included at the right end of the Table that aren't bound to external data. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_QueryTableRefresh">
   <sequence>
      <element name="queryTableFields" type="CT_QueryTableFields" minOccurs="1" maxOccurs="1"/>
      <element name="queryTableDeletedFields" type="CT_QueryTableDeletedFields" minOccurs="0"
        maxOccurs="1"/>
      <element name="sortState" minOccurs="0" maxOccurs="1" type="CT_SortState"/>
      <element name="extLst" minOccurs="0" maxOccurs="1" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="preserveSortFilterLayout" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="fieldIdWrapped" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="headersInLastRefresh" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="minimumVersion" type="xsd:unsignedByte" use="optional" default="0"/>
   <attribute name="nextId" type="xsd:unsignedInt" use="optional" default="1"/>
   <attribute name="unboundColumnsLeft" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="unboundColumnsRight" type="xsd:unsignedInt" use="optional" default="0"/>
</complexType>
```

3.13 External Data Connections

SpreadsheetML allows for the definition of top level data connection objects that describe how to retrieve data from external sources. These connection objects are independent of the constructs in the spreadsheet application that display data such as tables, PivotTables, etc.

Some information about a connection is considered part of the connection's definition. Other information is not inherently part of the connection, but it describes the way the connection is to be used by the containing workbook. Note that in many cases, the spreadsheet application does not need knowledge of the command syntax for the external data source (e.g., database query language), and simply stores a command string that was created by a data provider API (e.g., an ODBC driver).

A connection's definition can be established in a standalone connection file for easier sharing and reuse, but this reference documentation deals with the XML representation for external data connections that is directly embedded within a SpreadsheetML document. This embedded representation is expected whenever external data is used, and ensures portability of the document and continued operation of the external query in the most cases.

3.13.1 connection (Connection)

This element contains both the definition of how to get at an external data source as well as information describing how the connection is used within the workbook. Specific constructs in a worksheet, such as OLAP formulas, QueryTables, or PivotTables make use of information in the connection to retrieve or refresh data based on default events or the user's explicit request.

Parent Elements
connections (§3.13.2)

Child Elements	Subclause
dbPr (Database Properties)	§3.13.3
extLst (Future Feature Data Storage Area)	§3.2.10
olapPr (OLAP Properties)	§3.13.5
parameters (Query Parameters)	§3.13.7
textPr (Text Import Settings)	§3.13.12
webPr (Web Query Properties)	§3.13.13

Attributes	Description
background (Background Refresh)	Indicates whether the connection can be refreshed in the background (asynchronously). `true` if preferred usage of the connection is to refresh asynchronously in the background; `false` if preferred usage of the connection is to refresh synchronously in the foreground.

Attributes	Description
	This flag should be intentionally ignored in specific cases. [*Example*: An example of when the flag would be ignored is in the case of a connection to OLAP data on Microsoft SQL Server Analysis Services, where the connection is used by both a PivotTable and also by CUBE functions within the workbook. That connection will always be refreshed synchronously by the PivotTable and will always be refreshed asynchronously by the CUBE functions. *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
credentials (Reconnection Method)	Specifies the authentication method to be used when establishing (or re-establishing) the connection. The possible values for this attribute are defined by the ST_CredMethod simple type (§3.18.17).
deleted (Deleted Connection)	Indicates whether the associated workbook connection has been deleted. `true` if the connection has been deleted; otherwise, `false`. Deleted connections contain only the attributes name and deleted=`true`, all other information is removed from the SpreadsheetML file. If a new connection is created with the same name as a deleted connection, then the deleted connection is overwritten by the new connection. The possible values for this attribute are defined by the XML Schema boolean datatype.
description (Connection Description)	Specifies the user description for this connection. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
id (Connection Id)	Specifies The unique identifier of this connection. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
interval (Automatic Refresh Interval)	Specifies the number of minutes between automatic refreshes of the connection. When this attribute is not present, the connection is not automatically refreshed. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
keepAlive (Keep Connection Open)	`true` when the spreadsheet application should make efforts to keep the connection open. When `false`, the application should close the connection after retrieving the information. This corresponds to the MaintainConnection property of a PivotCache object. The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description
minRefreshableVersion (Minimum Version Required for Refresh)	For compatibility with legacy spreadsheet applications. This represents the minimum version # that is required to be able to correctly refresh the data connection. This attribute applies to connections that are used by a QueryTable. The possible values for this attribute are defined by the XML Schema unsignedByte datatype.
name (Connection Name)	Specifies the name of the connection. Each connection must have a unique name. When a connection has been marked as deleted and then a new connection is added with the same name, the deleted connection is replaced with the new connection. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
new (New Connection)	`true` if the connection has not been refreshed for the first time; otherwise, `false`. This state can happen when the user saves the file before a query has finished returning. The possible values for this attribute are defined by the XML Schema boolean datatype.
odcFile (Connection File)	Specifies the full path to external connection file from which this connection was created. If a connection fails during an attempt to refresh data, and reconnectionMethod=1, then the spreadsheet application will try again using information from the external connection file instead of the connection object embedded within the workbook. This is a benefit for data source and spreadsheetML document manageability. If the definition in the external connection file is changed (e.g., because of a database server name change), then the workbooks that made use of that connection will fail to connect with their internal connection information, and reload the new connection information from this file. This attribute is cleared by the spreadsheet application when the user manually edits the connection definition within the workbook. May be expressed in URI or system-specific file path notation. [*Note*: Applications can decide what forms of URI they support, and whether system-specific file path notations will be supported. *end note*] The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
onlyUseConnectionFile (Only Use Connection File)	Indicates whether the spreadsheet application should always and only use the connection information in the external connection file indicated by the odcFile attribute when the connection is refreshed. If `false`, then the spreadsheet application should follow the procedure indicated by the reconnectionMethod attribute described below.

Attributes	Description
	Applies only to OLE DB and ODBC connections, this attribute is ignored for other types of connections. The possible values for this attribute are defined by the XML Schema boolean datatype.
reconnectionMethod (Reconnection Method)	Specifies what the spreadsheet application should do when a connection fails. The values are as follows: `1 = As required`: On refresh use the existing connection information and if it ends up being invalid then get updated connection information, if available from the external connection file. `2 = Always`: On every refresh get updated connection information from the external connection file, if available, and use that instead of the existing connection information. In this case the data refresh will fail if the external connection file is unavailable. `3 = Never`: Never get updated connection information from the external connection file even if it is available and even if the existing connection information is invalid. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
refreshedVersion (Last Refresh Version)	For backward compatibility purposes, this attribute indicates the version of the spreadsheet application that last refreshed the connection. This attribute applies to connections that are used by a query table. The possible values for this attribute are defined by the XML Schema unsignedByte datatype.
refreshOnLoad (Refresh on Open)	`true` if this connection should be refreshed when opening the file; otherwise, `false`. The possible values for this attribute are defined by the XML Schema boolean datatype.
saveData (Save Data)	`true` if the external data fetched over the connection to populate a table is to be saved with the workbook; otherwise, `false`. This exists for data security purposes - if no external data is saved in (or "cached") in the workbook, then current user credentials can be required every time to retrieve the relevant data, and people won't see the data the workbook author had last been using before saving the file. The possible values for this attribute are defined by the XML Schema boolean datatype.
savePassword (Save Password)	`true` if the password is to be saved as part of the connection string; otherwise, False. The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description
singleSignOnId (SSO Id)	Identifier for Single Sign On (SSO) used for authentication between an intermediate spreadsheetML server and the external data source. [*Note*: Data connectivity can use a number of different technologies. One example of potential values stored in this attribute can be found at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/spptsdk/html/cSSOReturnCodes_SV01001109.asp *end note*] The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
sourceFile (Source Database File)	Used when the external data source is file-based. When a connection to such a data source fails, the spreadsheet application attempts to connect directly to this file. May be expressed in URI or system-specific file path notation. [*Note*: Applications can decide what forms of URI they support, and whether system-specific file path notations will be supported. *end note*] The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
type (Database Source Type)	Specifies the data source type. Values are as follows: 1. ODBC-based source 2. DAO-based source 3. File based database source 4. Web query 5. OLE DB-based source 6. Text-based source 7. ADO record set 8. DSP The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Connection">
   <sequence>
      <element name="dbPr" minOccurs="0" maxOccurs="1" type="CT_DbPr"/>
      <element name="olapPr" minOccurs="0" maxOccurs="1" type="CT_OlapPr"/>
      <element name="webPr" minOccurs="0" maxOccurs="1" type="CT_WebPr"/>
      <element name="textPr" minOccurs="0" maxOccurs="1" type="CT_TextPr"/>
      <element name="parameters" minOccurs="0" maxOccurs="1" type="CT_Parameters"/>
      <element name="extLst" minOccurs="0" maxOccurs="1" type="CT_ExtensionList"/>
   </sequence>
   <attribute name="id" use="required" type="xsd:unsignedInt"/>
   <attribute name="sourceFile" use="optional" type="ST_Xstring"/>
   <attribute name="odcFile" use="optional" type="ST_Xstring"/>
   <attribute name="keepAlive" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="interval" use="optional" type="xsd:unsignedInt" default="0"/>
   <attribute name="name" use="optional" type="ST_Xstring"/>
   <attribute name="description" use="optional" type="ST_Xstring"/>
   <attribute name="type" use="optional" type="xsd:unsignedInt"/>
   <attribute name="reconnectionMethod" use="optional" type="xsd:unsignedInt" default="1"/>
   <attribute name="refreshedVersion" use="required" type="xsd:unsignedByte"/>
   <attribute name="minRefreshableVersion" use="optional" type="xsd:unsignedByte" default="0"/>
   <attribute name="savePassword" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="new" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="deleted" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="onlyUseConnectionFile" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="background" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="refreshOnLoad" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="saveData" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="credentials" use="optional" type="ST_CredMethod" default="integrated"/>
   <attribute name="singleSignOnId" use="optional" type="ST_Xstring"/>
</complexType>
```

3.13.2 connections (Connections)

This element exists when there are one or more connections in the workbook. It is a container for the individual connection objects.

Parent Elements
Root element of SpreadsheetML Connections part

Child Elements	Subclause
connection (Connection)	§3.13.1

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Connections">
   <sequence>
      <element name="connection" minOccurs="1" maxOccurs="unbounded" type="CT_Connection"/>
   </sequence>
</complexType>
```

3.13.3 dbPr (Database Properties)

This element stores all properties associated with an ODBC or OLE DB external data connection.

[*Example*:

Data connectivity can use a number of different technologies. The following is one example XML fragment defining an OLE DB connection and the associated dbPr element:

```
<connection id="2"
  odcFile="C:\My Documents\My Data Sources\Northwind Orders.odc" keepAlive="1"
  name="Northwind Orders" description="northwind" type="5" refreshedVersion="3">
  <dbPr connection="Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist
    Security Info=True;Initial Catalog=Northwind;Data Source=dataserver1;Use
    Procedure for Prepare=1;Auto Translate=True;Packet Size=4096;Workstation
    ID=LOCAL_MACHINE_NAME;Use Encryption for Data=False;Tag with column
    collation when possible=False"
    command="&amp;quot;Northwind&amp;quot;.&amp;quot;dbo&amp;quot;.&amp;quot;Orders&amp;quot;"
    commandType="3"/>
</connection>
```

end example]

Parent Elements
connection (§3.13.1)

Attributes	Description
command (Command Text)	The string containing the database command to pass to the data provider API that will interact with the external source in order to retrieve data. These strings can be constructed in a variety of ways (from simple UIs built into the spreadsheet application for browsing and choosing tables and fields, to external applications providing user interface to build up complex queries, to advanced users editing text queries). The spreadsheetML application need not understand the command syntax; it can simply pass the command string to the data provider API in order to retrieve the latest external data. [*Example*: Data connectivity can use a number of different technologies. The following is one example of an ODBC command string of commandType=2 (for a Microsoft SQL Server database): `command="SELECT Orders.OrderID, Orders.OrderDate, Orders.ShipName, Orders.ShipAddress, Orders.ShipCity, Orders.ShipRegion, Orders.ShipPostalCode, Orders.ShipCountry_x000d__x000a_FROM Northwind.dbo.Orders Orders_x000d__x000a_WHERE (Orders.ShipCountry=?)"`

Attributes	Description
	Some characters in this string have been escaped - for more information on the escaping scheme, please refer to the ST_Xstring type definition. *end example*] [*Note*: the "?" syntax in the string is something that the ODBC data provider is aware of and may replace with a parameter before execution. *end note*] [*Example*: Data connectivity can use a number of different technologies. The following is one example of an OLE DB command string of commandType=3 (for an Oracle database): `command="&quot;TESTDB&quot;.&quot;ShippersTable&quot;"` *end example*] [*Note*: Data connectivity can use a number of different technologies. A few examples of potential values stored in this attribute can be found at: • http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odbc/htm/odbcsql_statements.asp • http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odbc/htm/odbcsql_minimum_grammar.asp • http://msdn.microsoft.com/library/default.asp?url=/library/en-us/oledb/htm/oledbusing_commands.asp *end note*] The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
commandType (OLE DB Command Type)	Specifies the OLE DB command type. Supported values are as follows: 1. Query specifies a cube name 2. Query specifies a SQL statement 3. Query specifies a table name 4. Query specifies that default information has been given, and it is up to the provider how to interpret. 5. Query is against a web based List Data Provider. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
connection (Connection String)	The connection string is used to make contact with an OLE DB or ODBC data source. These can be constructed in a variety of ways (from UI wizards built into the data provider code, to external query applications, to advanced users editing text files). The spreadsheetML application need not understand the connection syntax at all; it can simply pass the command string to the data provider API in order to re-establish a connection with the external data source.

Attributes	Description

	[*Example*: ODBC connection string to a database:
	
	`connection="DRIVER=SQL Server;SERVER=example_server;UID=example_useralias;APP=Microsoft Office 2007;WSID=user_alias;Trusted_Connection=Yes"`
	
	end example]
	
	[*Example*: of an OLE DB connection string to an Oracle database:
	
	`connection="Provider=OraOLEDB.Oracle.1;Password=example_password;Persist Security Info=True;User ID=example_useralias;Data Source=example_server;Extended Properties=&quot;&quot;"`
	
	end example]
	
	[*Note*: Data connectivity can use a number of different technologies. A few examples of potential values stored in this attribute can be found at:
	• http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odbc/htm/dasdkodbcoverview.asp
	• http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odbcsql/od_odbc_d_4x4k.asp
	• http://msdn.microsoft.com/library/default.asp?url=/library/en-us/ado270/htm/mdreforacleprovspec.asp
	end note]
	
	Connection strings syntaxes are specific to individual ODBC or OLE DB data provider drivers.
	
	The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

serverCommand (Command Text)	Specifies a second command text string that is persisted when PivotTable server-based page fields are in use. For ODBC connections, serverCommand is usually a broader query than command (no WHERE clause is present in the former). Based on these 2 commands, parameter UI can be populated and parameterized queries can be constructed. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DbPr">
    <attribute name="connection" use="required" type="ST_Xstring"/>
    <attribute name="command" use="optional" type="ST_Xstring"/>
    <attribute name="serverCommand" use="optional" type="ST_Xstring"/>
    <attribute name="commandType" use="optional" type="xsd:unsignedInt" default="2"/>
</complexType>
```

3.13.4 m (No Value)

This element is present when tables in a web query are missing.

Parent Elements
tables (§3.13.9)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TableMissing"/>
```

3.13.5 olapPr (OLAP Properties)

This element contains all the properties needed for an OLAP data connection. OLE DB for OLAP is the data provider, and OLAP connections contain both the dbPr and olapPr child elements.

[*Example*:

Data connectivity can use a number of different technologies. The following is an example of a connection to an SAP BW OLAP data source:

```
<connection id="1" odcFile="C:\My Documents\My Data Sources\$INFOCUBE.odc"
  keepAlive="1" name="SAP demo cube" description="SAP DemoCube" type="5"
  refreshedVersion="3" background="1">
  <dbPr connection="Provider=MDrmSap.2;Data Source=BI2;User
    ID=TESTUSER;Location=TESTSERVERNAME;Cache Authentication=False;Encrypt
    Password=False;Integrated Security=&amp;quot;&amp;quot;;Mask Password=False;Persist
    Encrypted=False;Persist Security Info=True;Impersonation
    Level=Anonymous;Mode=Read;Protection Level=None;Extended
    Properties=&amp;quot;SFC_CLIENT=800;&amp;quot;;Initial Catalog=$INFOCUBE"
    command="$0D_DECU" commandType="1"/>
  <olapPr sendLocale="1" rowDrillCount="1000" serverFill="0"
    serverNumberFormat="0" serverFont="0" serverFontColor="0"/>
</connection>
```

end example]

[*Example*:

Data connectivity can use a number of different technologies. The following is an example of a connection to a Microsoft SQL Server Analysis Services OLAP data source:

```
<connection id="1"
  odcFile="C:\My Documents\My Data Sources\Adventure Works DW.odc" keepAlive="1"
  name="Adventure Works DW" type="5" refreshedVersion="3" background="1">
  <dbPr connection="Provider=MSOLAP.3;Cache Authentication=False;Integrated
    Security=SSPI;Persist Security Info=True;Initial Catalog=Adventure Works
    DW;Data Source=DATASERVER1;Impersonation
    Level=Impersonate;Mode=ReadWrite;Protection Level=Pkt Privacy;Auto Synch
    Period=20000;Default Isolation Mode=0;Default MDX Visual Mode=0;MDX
    Compatibility=1;MDX Unique Name Style=0;Non Empty
    Threshold=0;SQLQueryMode=Calculated;Safety Options=2;Secured Cell
    Value=0;SOURCE_DSN_SUFFIX=&quot;Prompt=CompleteRequired;Window
    Handle=0x6A903CC;&quot;;SQL Compatibility=0;Compression Level=0;Real Time
    Olap=False;Packet Size=4096" command="Adventure Works" commandType="1"/>
  <olapPr sendLocale="1" rowDrillCount="1000"/>
</connection>
```

end example]

[*Note*: Data connectivity can use a number of different technologies. One example of potential values stored in this attribute can be found at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/oledb/htm/dasdkoledboverview.asp *end note*]

Parent Elements
connection (§3.13.1)

Attributes	Description
local (Local Cube)	Flag indicating whether we should get data from the local cube on refresh versus the original data source. `true` if a local cube has been created for OLAP data, and it should be used instead of the server. The possible values for this attribute are defined by the XML Schema boolean datatype.
localConnection (Local Cube Connection)	Specifies a connection string to use when a local cube is available. This is used when local is set to `true`. [*Example*: *<olapPr local="true" localConnection="OLEDB;Provider=MSOLAP;Data Source=C:\Data\DataCube.cub" >* *… end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
localRefresh (Local Refresh)	Flag indicating whether we should refresh the local cube from the original data source. When `true`, the original OLAP data source is queried each time the user explicitly refreshes the data in the application, and a new local cube is constructed from this query. The possible values for this attribute are defined by the XML Schema boolean datatype.
rowDrillCount (Drill Through Count)	Maximum number of drill-through rows to return when the user drills through an aggregate value in a PivotTable. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
sendLocale (Send Locale to OLAP)	When `true`, the spreadsheetML app should send the user interface locale ID to the OLAP provider to retrieve localized member names and properties, etc. When `false`, no locale ID is expected. The possible values for this attribute are defined by the XML Schema boolean datatype.
serverFill (OLAP Fill Formatting)	When `true` a PivotTable based on an OLAP source should format the data and aggregate cells in the PivotTable view using the background color from the OLAP source if this information is available. When `false`, OLAP server background fill colors are ignored, and standard formatting rules within the worksheet are followed. The possible values for this attribute are defined by the XML Schema boolean datatype.
serverFont (OLAP Server Font)	When `true`, a PivotTable based on OLAP source should format the data and aggregate cells in the PivotTable view using the font from the OLAP source (e.g., Arial or Tahoma). When `false`, OLAP server fonts are ignored, and standard formatting rules within the worksheet are followed. The possible values for this attribute are defined by the XML Schema boolean datatype.
serverFontColor (OLAP Font Formatting)	When `true` a PivotTable based on OLAP source should format the data and aggregate cells in the PivotTable view using the font color from the OLAP source. When `false`, OLAP server font colors are ignored, and standard formatting rules within the worksheet are followed. The possible values for this attribute are defined by the XML Schema boolean datatype.
serverNumberFormat (OLAP Number Format)	When `true`, a PivotTable based on OLAP source should format the data and aggregate cells in the PivotTable view using the number format from the OLAP source. When `false`, OLAP server number formats are ignored, and standard formatting rules within the worksheet are followed. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OlapPr">
   <attribute name="local" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="localConnection" use="optional" type="ST_Xstring"/>
   <attribute name="localRefresh" use="optional" type="xsd:boolean" default="true"/>
   <attribute name="sendLocale" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="rowDrillCount" use="optional" type="xsd:unsignedInt"/>
   <attribute name="serverFill" use="optional" type="xsd:boolean" default="true"/>
   <attribute name="serverNumberFormat" use="optional" type="xsd:boolean" default="true"/>
   <attribute name="serverFont" use="optional" type="xsd:boolean" default="true"/>
   <attribute name="serverFontColor" use="optional" type="xsd:boolean" default="true"/>
</complexType>
```

3.13.6 parameter (Parameter Properties)

This element stores properties about any parameters used with external data connections. Parameters are used to change the query executed externally and cause different data to be retrieved into the workbook. The type of parameter used – see ST_parameterType (§3.18.56) – determines whether the user will be prompted for a value before data is refreshed, or the value will be pulled from a cell in the workbook, or whether the same value should be used until explicitly changed in the data connection. Parameters are valid for ODBC and web queries.

[*Example*:

Data connectivity can use a number of different technologies. The following is an example of XML defining a connection to a Microsoft Access database, with a parameter based on the value in cell C1 on the first sheet.

```
<connection id="1" name="Connection" type="1" refreshedVersion="2"
  background="1" saveData="1">
  <dbPr connection="DSN=MS Access
    Database;DBQ=C:\Desktop\db1.mdb;DefaultDir=C:\Desktop;DriverId=25;FIL=MS
    Access;MaxBufferSize=2048;PageTimeout=5;" command="SELECT Table1.Field1,
    Table1.Field2_x000d__x000a_FROM `C:\Desktop\db1`.Table1
    Table1_x000d__x000a_WHERE (Table1.Field2=?)"/>
  <parameters count="1">
    <parameter name="user specified value" sqlType="4" parameterType="cell"
      cell="Sheet1!$C$1"/>
  </parameters>
</connection>
```

end example]

Note that the command string in the dbPr element contains a "?" character. This character serves as a parameter marker.

[*Note*: Data connectivity can use a number of different technologies. One example of potential values stored in this attribute can be found at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odbc/htm/odbcstatement_parameters.asp *end note*]

Parent Elements
parameters (§3.13.7)

Attributes	Description
boolean (Boolean)	Boolean value to use as the query parameter. Used only when parameterType = value. The possible values for this attribute are defined by the XML Schema boolean datatype.
cell (Cell Reference)	Cell reference indicating which cell's value to use for the query parameter. Used only when parameterType = cell. [*Example*: `<Parameter parameterType="cell" cell="Sheet1!$C$1">` *end example*] The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
double (Double)	Non-integer numeric value to use as the query parameter. Used only when parameterType = value. The possible values for this attribute are defined by the XML Schema double datatype.
integer (Integer)	Integer value to use as the query parameter. Used when parameterType = value. The possible values for this attribute are defined by the XML Schema int datatype.
name (Parameter Name)	The name of the parameter. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
parameterType (Parameter Type)	Type of parameter used. If the parameterType=value, then the value from boolean, double, integer, or string will be used. In this case, it is expected that only one of {boolean, double, integer, or string} will be specified. The possible values for this attribute are defined by the ST_ParameterType simple type (§3.18.56).
prompt (Parameter Prompt String)	Prompt string for the parameter. Presented to the spreadsheet user along with input UI to collect the parameter value before refreshing the external data. Used only when parameterType = prompt.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).</td></tr>
<tr><td>refreshOnChange (Refresh on Change)</td><td>Flag indicating whether the query should automatically refresh when the contents of a cell that provides the parameter value changes. If <code>true</code>, then external data is refreshed using the new parameter value every time there's a change. If <code>false</code>, then external data is only refreshed when requested by the user, or some other event triggers refresh (e.g., workbook opened).

The possible values for this attribute are defined by the XML Schema boolean datatype.</td></tr>
<tr><td>sqlType (SQL Data Type)</td><td>SQL data type of the parameter. Only valid for ODBC sources.

Supported values include:
<table>
<tr><th>-22</th><th>SQL_UNSIGNED_OFFSET</th></tr>
<tr><td>-20</td><td>SQL_SIGNED_OFFSET</td></tr>
<tr><td>-11</td><td>SQL_GUID</td></tr>
<tr><td>-10</td><td>SQL_WLONGVARCHAR</td></tr>
<tr><td>-9</td><td>SQL_WVARCHAR</td></tr>
<tr><td>-8</td><td>SQL_WCHAR</td></tr>
<tr><td>-7</td><td>SQL_BIT</td></tr>
<tr><td>-6</td><td>SQL_TINYINT</td></tr>
<tr><td>-5</td><td>SQL_BIGINT</td></tr>
<tr><td>-4</td><td>SQL_LONGVARBINARY</td></tr>
<tr><td>-3</td><td>SQL_VARBINARY</td></tr>
<tr><td>-2</td><td>SQL_BINARY</td></tr>
<tr><td>-1</td><td>SQL_LONGVARCHAR</td></tr>
<tr><td>0</td><td>SQL_UNKNOWN_TYPE</td></tr>
<tr><td>1</td><td>SQL_CHAR</td></tr>
<tr><td>2</td><td>SQL_NUMERIC</td></tr>
<tr><td>3</td><td>SQL_DECIMAL</td></tr>
<tr><td>4</td><td>SQL_INTEGER</td></tr>
<tr><td>5</td><td>SQL_SMALLINT</td></tr>
<tr><td>6</td><td>SQL_FLOAT</td></tr>
<tr><td>7</td><td>SQL_REAL</td></tr>
<tr><td>8</td><td>SQL_DOUBLE</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>9</td><td>SQL_TYPE_DATE or SQL_DATE</td></tr>
<tr><td>10</td><td>SQL_TYPE_TIME or SQL_TIME</td></tr>
<tr><td>11</td><td>SQL_TYPE_TIMESTAMP or SQL_TIMESTAMP</td></tr>
<tr><td>12</td><td>SQL_VARCHAR</td></tr>
<tr><td>101</td><td>SQL_INTERVAL_YEAR</td></tr>
<tr><td>102</td><td>SQL_INTERVAL_MONTH</td></tr>
<tr><td>103</td><td>SQL_INTERVAL_DAY</td></tr>
<tr><td>104</td><td>SQL_INTERVAL_HOUR</td></tr>
<tr><td>105</td><td>SQL_INTERVAL_MINUTE</td></tr>
<tr><td>106</td><td>SQL_INTERVAL_SECOND</td></tr>
<tr><td>107</td><td>SQL_INTERVAL_YEAR_TO_MONTH</td></tr>
<tr><td>108</td><td>SQL_INTERVAL_DAY_TO_HOUR</td></tr>
<tr><td>109</td><td>SQL_INTERVAL_DAY_TO_MINUTE</td></tr>
<tr><td>110</td><td>SQL_INTERVAL_DAY_TO_SECOND</td></tr>
<tr><td>111</td><td>SQL_INTERVAL_HOUR_TO_MINUTE</td></tr>
<tr><td>112</td><td>SQL_INTERVAL_HOUR_TO_SECOND</td></tr>
<tr><td>113</td><td>SQL_INTERVAL_MINUTE_TO_SECOND</td></tr>
</table>

The possible values for this attribute are defined by the XML Schema int datatype.
</td></tr>
<tr><td>string (String)</td><td>String value to use as the query parameter. Used only when parameterType = value.

The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).</td></tr>
</table>

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Parameter">
   <attribute name="name" use="optional" type="ST_Xstring"/>
   <attribute name="sqlType" use="optional" type="xsd:int" default="0"/>
   <attribute name="parameterType" use="optional" type="ST_ParameterType" default="prompt"/>
   <attribute name="refreshOnChange" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="prompt" use="optional" type="ST_Xstring"/>
   <attribute name="boolean" use="optional" type="xsd:boolean"/>
   <attribute name="double" use="optional" type="xsd:double"/>
   <attribute name="integer" use="optional" type="xsd:int"/>
   <attribute name="string" use="optional" type="ST_Xstring"/>
   <attribute name="cell" use="optional" type="ST_Xstring"/>
</complexType>
```

3.13.7 parameters (Query Parameters)

This element serves as a collection of parameters for an ODBC or web query.

Parent Elements
connection (§3.13.1)

Child Elements	Subclause
parameter (Parameter Properties)	§3.13.6

Attributes	Description
count (Parameter Count)	The number of parameters used. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Parameters">
   <sequence>
      <element name="parameter" minOccurs="1" maxOccurs="unbounded" type="CT_Parameter"/>
   </sequence>
   <attribute name="count" use="optional" type="xsd:unsignedInt"/>
</complexType>
```

3.13.8 s (Character Value)

This element is used to specify an HTML table to import by name. If the tables are not named, they'll be specified with the <x v="[index]">syntax instead.

Parent Elements
metadataStrings (§3.9.9); tables (§3.13.9)

Attributes	Description
v (Value)	The name of the table to retrieve when the web query is refreshed. This corresponds to the string used for the id attribute of the HTML <table> tag. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_XStringElement">
   <attribute name="v" type="ST_Xstring" use="required"/>
</complexType>
```

3.13.9 tables (Tables)

This element serves as the collection of tables to be returned via a web query data connection. Tables are then most commonly referenced by <x> via their indices (in order of the <Table> tags in the HTML page).

Parent Elements
webPr (§3.13.13)

Child Elements	Subclause
m (No Value)	§3.13.4
s (Character Value)	§3.13.8
x (Shared Items Index)	§3.10.1.97

Attributes	Description
count (Count of Tables)	Number of tables to pull data from when refreshing from a web query. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Tables">
   <choice minOccurs="1" maxOccurs="unbounded">
      <element name="m" type="CT_TableMissing"/>
      <element name="s" type="CT_XStringElement"/>
      <element name="x" type="CT_Index"/>
   </choice>
   <attribute name="count" use="optional" type="xsd:unsignedInt"/>
</complexType>
```

3.13.10 textField (Text Import Field Settings)

This element specifies field settings for text import.

Parent Elements
textFields (§3.13.11)

Attributes	Description

Attributes	Description
position (Position)	The character position the field starts at for fixed-length fields. The index is 0-based. If this attribute does not exist, position=0 is assumed. Subsequent textField elements or carriage returns in the text stream serve to denote endpoints for text fields. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
type (Field Type)	Specifies the field Type. When text is imported into cells in the worksheet, the data in the cells are converted to the type defined here. Types can be specified by the user, or determined algorithmically via heuristics and text analysis. The possible values for this attribute are defined by the ST_ExternalConnectionType simple type (§3.18.28).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextField">
   <attribute name="type" use="optional" type="ST_ExternalConnectionType" default="general"/>
   <attribute name="position" use="optional" type="xsd:unsignedInt" default="0"/>
</complexType>
```

3.13.11 textFields (Fields)

This element that denotes a set of fields to retrieve from a text file. Contains 1 or more textField elements.

Parent Elements
textPr (§3.13.12)

Child Elements	Subclause
textField (Text Import Field Settings)	§3.13.10

Attributes	Description
count (Count of Fields)	Number of distinct fields to retrieve. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextFields">
   <sequence>
      <element name="textField" minOccurs="1" maxOccurs="unbounded" type="CT_TextField"/>
   </sequence>
   <attribute name="count" use="optional" type="xsd:unsignedInt" default="1"/>
</complexType>
```

3.13.12 textPr (Text Import Settings)

This element contains all of the text import settings.

Here's an example of the XML for a text connection:

[*Example*:

```
<connection id="1" name="text data" type="6" refreshedVersion="3" background="1"
  saveData="1">
  <textPr prompt="0" codePage="437" sourceFile="C:\Desktop\text data.txt"
    delimiter="|">
    <textFields count="5">
      <textField/>
      <textField type="text" position="7"/>
      <textField type="text" position="28"/>
      <textField position="36"/>
      <textField type="text" position="41"/>
    </textFields>
  </textPr>
</connection>
```

example]

Parent Elements
connection (§3.13.1)

Child Elements	Subclause
textFields (Fields)	§3.13.11

Attributes	Description
codePage (Code Page)	Code page associated with the text file. [*Example*:s of supported values include: 1256: Arabic (Windows) 775: Baltic (DOS)

Attributes	Description
	28594: Baltic (ISO) 1257: Baltic (Windows) 852: Central European (DOS) 28592: Central European (ISO) 10029: Central European (Mac) 1250: Central European (Windows) 936: Chinese Simplified (GB2312) 950: Chinese Traditional (Big5) 10082: Croatian (Mac) 866: Cyrillic (DOS) 28595: Cyrillic (ISO) 20866: Cyrillic (KOI8-R) 21866: Cyrillic (KOI8-U) 10007: Cyrillic (Mac) 1251: Cyrillic (Windows) 28603: Estonian (ISO) 863: French Canadian (DOC) 737: Greek (DOS) 28597: Greek (ISO) 10006: Greek (Mac) 1253: Greek (Windows) 869: Greek, Modern (DOS) 1255: Hebrew (Windows) 861: Icelandic (DOS) 10079: Icelandic (Mac) 932: Japanese (Shift-JIS) 949: Korean 1361: Korean (Johab) 28605: Latin 9 (ISO) 865: Nordic (DOS) 855: OEM Cyrillic 437: OEM United States 860: Portuguese (DOS) 10010: Romanian (Mac) 20261: T.61 874: Thai (Windows) 857: Turkish (DOS) 28599: Turkish (ISO) 10081: Turkish (Mac) 1254: Turkish (Windows) 10017: Ukrainian (Mac) 65000: Unicode (UTF-7) 65001: Unicode (UTF-8) 20127: US-ASCII 1258: Vietnamese (Windows) 850: Western European (DOS)

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>28591: Western European (ISO)
10000: Western European (Mac)
1252: Western European (Windows)

The possible values for this attribute are defined by the XML Schema unsignedInt datatype.</td></tr>
<tr><td>comma (Comma is Delimiter)</td><td>Flag indicating whether to treat comma characters as field delimiters.

The possible values for this attribute are defined by the XML Schema boolean datatype.</td></tr>
<tr><td>consecutive (Consecutive Delimiters)</td><td>Flag indicating whether consecutive delimiters should be treated as just one delimiter. If this flag is true than it's possible or even likely that some rows will return more fields than others, and these fields will always fill cells in the worksheet from left to right.

The possible values for this attribute are defined by the XML Schema boolean datatype.</td></tr>
<tr><td>decimal (Decimal Separator)</td><td>The decimal separator character. This and the thousands attribute are used only when data in the text file contains decimal and thousands separators that are different from those used on the computer, due to a different language setting being used.

The following table shows the results when you import text into a spreadsheet application using various separators. Numeric results are displayed in the rightmost column.

<table>
<tr><th>System decimal separator</th><th>System thousands separator</th><th>Text file decimal separator value</th><th>Text file thousands Separator value</th><th>Text imported</th><th>Cell value (d</th></tr>
<tr><td>Period</td><td>Comma</td><td>Comma</td><td>Period</td><td>123.123,45</td><td>123,123.45 (n</td></tr>
<tr><td>Period</td><td>Comma</td><td>Comma</td><td>Comma</td><td>123.123,45</td><td>123.123,45 (t</td></tr>
<tr><td>Comma</td><td>Period</td><td>Comma</td><td>Period</td><td>123,123.45</td><td>123,123.45 (n</td></tr>
<tr><td>Period</td><td>Comma</td><td>Period</td><td>Comma</td><td>123 123.45</td><td>123 123.45 (t</td></tr>
<tr><td>Period</td><td>Comma</td><td>Period</td><td>Space</td><td>123 123.45</td><td>123,123.45 (n</td></tr>
</table>
Strings values of this attribute are expected to be one character in length.

The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).</td></tr>
<tr><td>delimited (Delimited File)</td><td>true if the file is Tab or character delimited. false if the file should be parsed according to fixed length fields.

The possible values for this attribute are defined by the XML Schema boolean datatype.</td></tr>
</table>

Attributes	Description
delimiter (Custom Delimiter)	User-specified character to be treated as a field delimiter. Only single characters are supported. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
fileType (File Type)	Ignorable attribute with enum value that equals either "Macintosh," "Windows (ANSI)," or "MS-DOS (PC-8)" (see the definition of ST_FileType) and determines the kind of character set to use during import. Only one of fileType and codePage should be specified for a textPr. The possible values for this attribute are defined by the ST_FileType simple type (§3.18.30).
firstRow (First Row)	Indicates at what row of the file to start the data import. All unsignedInt values are valid, although it's possible that firstRow will be higher than the number of rows in the text file, in which case no data will be imported. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
prompt (Prompt for File Name)	Flag indicating whether the user wants to be prompted for the file name on refresh. If `false`, then the user is not prompted. If `true` or not present, then the user is prompted. The possible values for this attribute are defined by the XML Schema boolean datatype.
qualifier (Qualifier)	Character used as the text string qualifier. The possible values for this attribute are defined by the ST_Qualifier simple type (§3.18.63).
semicolon (Semicolon is Delimiter)	Flag indicating whether to treat semicolon characters as field delimiters. The possible values for this attribute are defined by the XML Schema boolean datatype.
sourceFile (Source File Name)	Path to the text file to use to import external data. May be expressed in URI or system-specific file path notation. [*Note*: Applications can decide what forms of URI they support, and whether system-specific file path notations will be supported. *end note*] The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
space (Space is Delimiter)	Flag indicating whether to treat space characters as field delimiters. The possible values for this attribute are defined by the XML Schema boolean datatype.
tab (Tab as	Flag indicating whether to treat tab characters as field delimiters. If `false`, then tabs will

Attributes	Description
Delimiter)	not be used as delimiters. If `true` or not present, then they will be used as delimiters. The possible values for this attribute are defined by the XML Schema boolean datatype.
thousands (Thousands Separator)	The thousands separator character. This and the decimal attribute are used only when data in the text file contains decimal and thousands separators that are different from those used on the computer, due to a different language setting being used. Please refer to the decimal attribute description above for a Table describing the behavior. Strings values of this attribute are expected to be one character in length. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextPr">
   <sequence>
      <element name="textFields" minOccurs="0" maxOccurs="1" type="CT_TextFields"/>
   </sequence>
   <attribute name="prompt" use="optional" type="xsd:boolean" default="true"/>
   <attribute name="fileType" use="optional" type="ST_FileType" default="win"/>
   <attribute name="codePage" use="optional" type="xsd:unsignedInt" default="1252"/>
   <attribute name="firstRow" use="optional" type="xsd:unsignedInt" default="1"/>
   <attribute name="sourceFile" use="optional" type="ST_Xstring" default=""/>
   <attribute name="delimited" use="optional" type="xsd:boolean" default="true"/>
   <attribute name="decimal" use="optional" type="ST_Xstring" default="."/>
   <attribute name="thousands" use="optional" type="ST_Xstring" default=","/>
   <attribute name="tab" use="optional" type="xsd:boolean" default="true"/>
   <attribute name="space" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="comma" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="semicolon" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="consecutive" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="qualifier" use="optional" type="ST_Qualifier" default="doubleQuote"/>
   <attribute name="delimiter" use="optional" type="ST_Xstring"/>
</complexType>
```

3.13.13 webPr (Web Query Properties)

This element specifies the properties for a web query source. A web query will retrieve data from HTML tables, and can also supply HTTP "Get" parameters to be processed by the web server in generating the HTML by including the parameters and parameter elements.

Here's an example of a web query connection:

[*Example*:

```
<connection id="1" name="Connection" type="4" refreshedVersion="0"
  background="1" saveData="1">
  <webPr sourceData="1" parsePre="1" consecutive="1"
    url="http://ServerName/Image%20Library/Forms/AllItems.aspx" htmlTables="1">
    <tables count="1">
      <s v="contentthumbnail"/>
    </tables>
  </webPr>
</connection>
```

end example]

Parent Elements
connection (§3.13.1)

Child Elements	Subclause
tables (Tables)	§3.13.9

Attributes	Description
consecutive (Consecutive Delimiters)	Flag indicating whether consecutive delimiters should be treated as just one delimiter. The possible values for this attribute are defined by the XML Schema boolean datatype.
editPage (Edit Query URL)	The URL of the user-facing web page showing the web query data. This URL is persisted in the case that sourceData="true" and url has been redirected to reference an XML file. Then the user-facing page can be shown in the UI, and the XML data can be retrieved behind the scenes. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
firstRow (Use First Row)	Flag indicating whether to parse all tables inside a PRE block with the same width settings as the first row. The possible values for this attribute are defined by the XML Schema boolean datatype.
htmlFormat (HTML Formatting Handling)	How to handle formatting from the HTML source when bringing web query data into the worksheet. Relevant when sourceData is True. Values are as follows: 1. None - no formatting at all 2. RTF - honor just rich text formatting 3. All - honor all html formatting. The possible values for this attribute are defined by the ST_HtmlFmt simple type

Attributes	Description
	(§3.18.43).
htmlTables (HTML Tables Only)	Flag indicating whether web queries should only work on HTML tables. The possible values for this attribute are defined by the XML Schema boolean datatype.
parsePre (Parse PRE)	Flag indicating whether data contained within HTML <PRE> tags in the web page is parsed into columns when you import the page into a query table. The possible values for this attribute are defined by the XML Schema boolean datatype.
post (Web Post)	Returns or sets the string used with the post method of inputting data into a web server to return data from a web query. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
sourceData (Import XML Source Data)	Flag indicating that XML source data should be imported instead of the HTML table itself. Used when a web query exists to an HTML table with the following attribute. `<TABLE ... o:WebQuerySourceHRef="http://..." ... > ... </TABLE>` The possible values for this attribute are defined by the XML Schema boolean datatype.
textDates (Dates as Text)	Flag indicating whether dates should be imported into cells in the worksheet as text rather than dates. The possible values for this attribute are defined by the XML Schema boolean datatype.
url (URL)	URL to use to refresh external data. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
xl2000 (Refreshed in Excel 2000)	This flag exists for backward compatibility with older existing spreadsheet files, and is set to `true` if this web query was refreshed in a spreadsheet application newer than or equal to Microsoft Excel 2000. This is an optional attribute that can be ignored. The possible values for this attribute are defined by the XML Schema boolean datatype.
xl97 (Created in Excel 97)	This flag exists for backward compatibility with older existing spreadhseet files, and is set to `true` if this web query was created in Microsoft Excel 97. This is an optional attribute that can be ignored. The possible values for this attribute are defined by the XML Schema boolean datatype.
xml (XML Source)	`true` if the web query source is XML (versus HTML), otherwise `false`. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_WebPr">
   <sequence>
      <element name="tables" minOccurs="0" maxOccurs="1" type="CT_Tables"/>
   </sequence>
   <attribute name="xml" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="sourceData" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="parsePre" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="consecutive" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="firstRow" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="xl97" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="textDates" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="xl2000" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="url" use="optional" type="ST_Xstring"/>
   <attribute name="post" use="optional" type="ST_Xstring"/>
   <attribute name="htmlTables" use="optional" type="xsd:boolean" default="false"/>
   <attribute name="htmlFormat" use="optional" type="ST_HtmlFmt" default="none"/>
   <attribute name="editPage" use="optional" type="ST_Xstring"/>
</complexType>
```

3.14 Supplementary Workbook Data

External links are used when linking the workbook to other workbooks or external data. The most frequent feature for linking a workbook to other workbooks is through the use of formulas. In this case the formula references a range or defined name in another workbook. Hyperlinks on cells and other spreadsheet objects are also considered an external link. OLE links are yet another technology used to link the workbook to another object. Finally, Dynamic Data Exchange, or DDE, servers can be used to access external data. DDE servers are accessed through formulas in the workbook.

External links are saved with the target source in a relationship file so that external resources are easily discoverable in lightweight relationship XML rather than deep in the application's XML.

For a workbook consumer.xlsx that makes use of data in another workbook called data.xlsx, the following XML would exist in consumer.xlsx to describe the external link:

[*Example*:

```
<Relationships xmlns="…">
  <Relationship Id="rId1" Type="…/externalLinkPath" Target="data.xlsx"
    TargetMode="External"/>
</Relationships>
```

end example]

And the following XML would exist to describe cached data retrieved from the external workbook:

[*Example*:

```
<externalLink xmlns="…">
  <externalBook xmlns:r="…" r:id="rId1">
    <sheetNames>
      <sheetName val="Sheet1"/>
      <sheetName val="Sheet2"/>
      <sheetName val="Sheet3"/>
    </sheetNames>
    <sheetDataSet>
      <sheetData sheetId="0"/>
      <sheetData sheetId="1"/>
      <sheetData sheetId="2">
        <row r="11">
          <cell r="B11">
            <v>47</v>
          </cell>
        </row>
        <row r="12">
          <cell r="B12">
            <v>19</v>
          </cell>
        </row>
        <row r="13">
          <cell r="B13">
            <v>38</v>
          </cell>
        </row>
      </sheetData>
    </sheetDataSet>
  </externalBook>
</externalLink>
```

end example]

The Supplementary Workbook Data section of SpreadsheetML is complimentary to the External Data Connnections (§3.13) section in maintaining all the information about external information that impacts the workbook.

3.14.1 cell (External Cell Data)

This element is used to store cached values from external sources such as other workbooks. Formulas from external cells are not stored in the consuming workbook. Also, for this context, the attribute t cannot have a value of `inlineStr`. Rich text is not supported in this context either.

Parent Elements

Parent Elements
row (§3.14.12)

Child Elements	Subclause
v (Cell Value)	§3.3.1.93

Attributes	Description

r (Reference)	Describes the cell location in the external book.
	
	[*Example*:
	
	`<cell r="B12">`
	`  <v>74</v>`
	`</cell>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_CellRef simple type (§3.18.8).

t (Type)	Indicates the data type of the cell value. The possible values for this attribute are defined by the ST_CellType simple type (§3.18.12).
vm (Value Metadata)	The index of the cell's value metadata, if any exists. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ExternalCell">
   <sequence>
      <element name="v" type="ST_Xstring" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="r" type="ST_CellRef" use="optional"/>
   <attribute name="t" type="ST_CellType" use="optional" default="n"/>
   <attribute name="vm" type="xsd:unsignedInt" use="optional" default="0"/>
</complexType>
```

3.14.2 ddeItem (DDE Item definition)

This element represents a DDE item.

Parent Elements
ddeItems (§3.14.3)

Child Elements	Subclause
values (DDE Name Values)	§3.14.19

Attributes	Description
advise (Advise)	Specifies whether the DDE server should notify the application when the external data changes. Default value is `false`. The possible values for this attribute are defined by the XML Schema boolean datatype.
name (DDE Name)	Specifies the DDE item name. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
ole (OLE)	Set to `true` if this is item uses OLE. Default value is `false`. The possible values for this attribute are defined by the XML Schema boolean datatype.
preferPic (Data is an Image)	Set to `true` if data from this DDE item is an image format. Default value is `false`. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DdeItem">
    <sequence>
        <element name="values" type="CT_DdeValues" minOccurs="0" maxOccurs="1"/>
    </sequence>
    <attribute name="name" type="ST_Xstring" default="0"/>
    <attribute name="ole" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="advise" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="preferPic" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.14.3 ddeItems (DDE Items Collection)

This element serves as a collection for ddeItem elements.

Parent Elements
ddeLink (§3.14.4)

Child Elements	Subclause
ddeItem (DDE Item definition)	§3.14.2

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DdeItems">
   <sequence>
      <element name="ddeItem" type="CT_DdeItem" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.14.4 ddeLink (DDE Connection)

This element represents a connection to an external Dynamic Data Exchange (DDE) server. DDE is a method of sending data between applications using Windows messages according to a documented protocol that has been stable since about 1990.

The hierarchy of names defined by a DDE server is Application, Topics, and Items. Topics often correspond to units such as files or documents or database names, and Items refer to subsets of the data such as cell ranges, rows, fields, columns. DDE items can have multiple values as well.

[*Example*:

Data connectivity can use a number of different technologies.The following is just one example of a spreadsheetML fragment describing the product Microsoft Excel being used as a DDE server to provide data to the current spreadsheet document:

```
<ddeLink xmlns:r="…" ddeService="excel" ddeTopic="[ddesource.xls]Sheet1">
  <ddeItems>
    <ddeItem name="R1C1" advise="1"/>
    <ddeItem name="StdDocumentName" ole="1" advise="1"/>
  </ddeItems>
</ddeLink>
```

end example]

Parent Elements
externalLink (§3.14.8)

Child Elements	Subclause
ddeItems (DDE Items Collection)	§3.14.3

Attributes	Description
ddeService (Service name)	Service name (i.e., application name) for the DDE connection. This is a required attribute. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

Attributes	Description
ddeTopic (Topic for DDE server)	Describes something for the DDE application to which the channel pertains— usually a document of that application. This is a required attribute. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DdeLink">
   <sequence>
      <element name="ddeItems" type="CT_DdeItems" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="ddeService" type="ST_Xstring" use="required"/>
   <attribute name="ddeTopic" type="ST_Xstring" use="required"/>
</complexType>
```

3.14.5 definedName (Defined Name)

This element contains information about a named range in an external workbook.

Parent Elements
definedNames (§3.14.6)

Attributes	Description
name (Defined Name)	The defined name. This attribute is required. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

refersTo (Refers To)	Name range definition string.
	
	[*Example*:
	
	`<definedNames>`
	`  <definedName name="namedrange"`
	`  refersTo="='Sheet1'!$D$5:$D$10"/>`
	`</definedNames>`
	
	…end example]
	
	The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

sheetId (Sheet Id)	The index of the worksheet that the named range applies to for named ranges that are scoped to a particular worksheet rather than the full workbook. This attribute is optional.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ExternalDefinedName">
   <attribute name="name" type="ST_Xstring" use="required"/>
   <attribute name="refersTo" type="ST_Xstring" use="optional"/>
   <attribute name="sheetId" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

3.14.6 definedNames (Named Links)

This element is a collection of the defined names associated with the supporting workbook.

Parent Elements
externalBook (§3.14.7)

Child Elements	Subclause
definedName (Defined Name)	§3.14.5

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ExternalDefinedNames">
   <sequence>
      <element name="definedName" type="CT_ExternalDefinedName" minOccurs="0"
        maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.14.7 externalBook (External Workbook)

This element represents an external workbook which is supplying data to the current workbook.

Parent Elements
externalLink (§3.14.8)

Child Elements	Subclause
definedNames (Named Links)	§3.14.6
sheetDataSet (Cached Worksheet Data)	§3.14.14
sheetNames (Supporting Workbook Sheet Names)	§3.14.16

Attributes	Description

Attributes	Description
id (Relationship to supporting book file path) Namespace: .../officeDocument/2006/relationships	Relationship ID that references a link in the relationships collection. The target attribute in the associated relationship will specify the worksheet XML file in the current spreadsheetML document ZIP archive that makes use of this externalbook. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ExternalBook">
   <sequence>
      <element name="sheetNames" type="CT_ExternalSheetNames" minOccurs="0" maxOccurs="1"/>
      <element name="definedNames" type="CT_ExternalDefinedNames" minOccurs="0" maxOccurs="1"/>
      <element name="sheetDataSet" type="CT_ExternalSheetDataSet" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute ref="r:id" use="required"/>
</complexType>
```

3.14.8 externalLink (External Reference)

This element is a container for specific types of external links.

Parent Elements
Root element of SpreadsheetML External Workbook References part

Child Elements	Subclause
ddeLink (DDE Connection)	§3.14.4
externalBook (External Workbook)	§3.14.7
extLst (Future Feature Data Storage Area)	§3.2.10
oleLink (OLE Link)	§3.14.11

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ExternalLink">
   <choice>
      <element name="externalBook" type="CT_ExternalBook" minOccurs="0" maxOccurs="1"/>
      <element name="ddeLink" type="CT_DdeLink" minOccurs="0" maxOccurs="1"/>
      <element name="oleLink" type="CT_OleLink" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </choice>
</complexType>
```

3.14.9 oleItem (OLE Link Item)

This element represents an OLE link.

Parent Elements
oleItems (§3.14.10)

Attributes	Description
advise (Advise)	Set to `true` if the OLE server should notify the application when the external data changes. Default value is `false`. The possible values for this attribute are defined by the XML Schema boolean datatype.
icon (Icon)	Set to `true` if the object is represented by an icon. The possible values for this attribute are defined by the XML Schema boolean datatype.
name (Object Name)	The object's name. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).
preferPic (Object is an Image)	Set to `true` if the object is represented by an image. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OleItem">
   <attribute name="name" type="ST_Xstring" use="required"/>
   <attribute name="icon" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="advise" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="preferPic" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.14.10 oleItems (OLE Link Items)

This element is a collection of OLE items.

Parent Elements
oleLink (§3.14.11)

Child Elements	Subclause
oleItem (OLE Link Item)	§3.14.9

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OleItems">
   <sequence>
      <element name="oleItem" type="CT_OleItem" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.14.11 oleLink (OLE Link)

This element represents an external link to an OLE2 connection, specified by a progID/object pair.

Parent Elements
externalLink (§3.14.8)

Child Elements	Subclause
oleItems (OLE Link Items)	§3.14.10

Attributes	Description
id (OLE Link Relationship) Namespace: .../officeDocument/2006/relationships	Relationship ID that references a link in the relationships collection. The target attribute in the associated relationship will specify the external file name used for this oleLink. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
progId (OLE Link ProgID)	The progID for the OLE connection. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OleLink">
   <sequence>
      <element name="oleItems" type="CT_OleItems" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute ref="r:id" use="required"/>
   <attribute name="progId" type="ST_Xstring" use="required"/>
</complexType>
```

3.14.12 row (Row)

This element contains data for an external worksheet row.

Parent Elements
sheetData (§3.14.13)

Child Elements	Subclause
cell (External Cell Data)	§3.14.1

Attributes	Description
r (Row)	Row number of the row in the external book containing the cell data referenced. This attribute is required. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ExternalRow">
   <sequence>
      <element name="cell" type="CT_ExternalCell" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="r" type="xsd:unsignedInt" use="required"/>
</complexType>
```

3.14.13 sheetData (External Sheet Data Set)

This element contains the cached worksheet data associated with a supporting workbook.

For an example, please refer to example at the beginning of this section.

Parent Elements
sheetDataSet (§3.14.14)

Child Elements	Subclause
row (Row)	§3.14.12

Attributes	Description
refreshError (Last Refresh Resulted in Error)	Specifies that the last external data refresh for this sheet did not succeed. The possible values for this attribute are defined by the XML Schema boolean datatype.
sheetId (Sheet Id)	Index of sheet in the external workbook that is referenced and partially cached in this data set. This is a 1-based index. This attribute is required. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ExternalSheetData">
   <sequence>
      <element name="row" type="CT_ExternalRow" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="sheetId" type="xsd:unsignedInt" use="required"/>
   <attribute name="refreshError" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

3.14.14 sheetDataSet (Cached Worksheet Data)

This element serves as the collection for 1 or more sheetData elements.

Parent Elements
externalBook (§3.14.7)

Child Elements	Subclause
sheetData (External Sheet Data Set)	§3.14.13

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ExternalSheetDataSet">
   <sequence>
      <element name="sheetData" type="CT_ExternalSheetData" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

3.14.15 sheetName (Sheet Name)

Name of a worksheet in the supporting workbook

Parent Elements
sheetNames (§3.14.16)

Attributes	Description
val (Sheet Name Value)	Name of the sheet. This attribute is required. The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ExternalSheetName">
   <attribute name="val" type="ST_Xstring"/>
</complexType>
```

3.14.16 sheetNames (Supporting Workbook Sheet Names)

This element is the container for all of the worksheet names in a supporting workbook.

Parent Elements
externalBook (§3.14.7)

Child Elements	Subclause
sheetName (Sheet Name)	§3.14.15

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ExternalSheetNames">
   <sequence>
      <element name="sheetName" minOccurs="1" maxOccurs="unbounded" type="CT_ExternalSheetName"/>
   </sequence>
</complexType>
```

3.14.17 val (DDE Link Value)

This element specifies a value associated with a particular DDE item.

Here's an example of how values, value, and val elements are written out in the spreadsheetML for a ddeItem supplied by a DDE server. In this example different cells in the workbook are bound to these specific DDE items

[*Example*:

```
<ddeLink xmlns:r="…" ddeService="StockSrv" ddeTopic="Prices">
  <ddeItems>
    <ddeItem name="Bread" advise="1">
      <values>
        <value>
          <val>3.5</val>
        </value>
      </values>
    </ddeItem>
    <ddeItem name="Milk" advise="1">
      <values>
        <value>
          <val>5.7400000000000002</val>
        </value>
      </values>
    </ddeItem>
```

```
      <ddeItem name="MSFT" advise="1">
        <values>
          <value>
            <val>54.130000000000003</val>
          </value>
        </values>
      </ddeItem>
      <ddeItem name="StdDocumentName" ole="1" advise="1"/>
    </ddeItems>
  </ddeLink>
```

example]

The possible values for this element are defined by the ST_Xstring simple type (§3.18.96).

Parent Elements
value (§3.14.18)

3.14.18 value (Value)

This element contains a value associated with a particular DDE item. This serves as a container for the val element.

Parent Elements
values (§3.14.19)

Child Elements	Subclause
val (DDE Link Value)	§3.14.17

Attributes	Description
t (DDE Value Type)	Indicates the type of the DDE value. The possible values for this attribute are defined by the ST_DdeValueType simple type (§3.18.24).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DdeValue">
   <sequence>
      <element name="val" type="ST_Xstring" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="t" type="ST_DdeValueType" use="optional" default="n"/>
</complexType>
```

3.14.19 values (DDE Name Values)

This element defines a collection of values associated with DDE item.

Parent Elements
ddeItem (§3.14.2)

Child Elements	Subclause
value (Value)	§3.14.18

Attributes	Description
cols (Columns)	The number of columns of data that will be returned by the DDE server for this DDE item. The default value of this attribute is 1. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
rows (Rows)	The number of rows of data that will be returned by the DDE server for this DDE item. The default value of this attribute is 1. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DdeValues">
    <sequence>
        <element name="value" minOccurs="1" maxOccurs="unbounded" type="CT_DdeValue"/>
    </sequence>
    <attribute name="rows" type="xsd:unsignedInt" use="optional" default="1"/>
    <attribute name="cols" type="xsd:unsignedInt" use="optional" default="1"/>
</complexType>
```

3.15 Volatile Dependencies

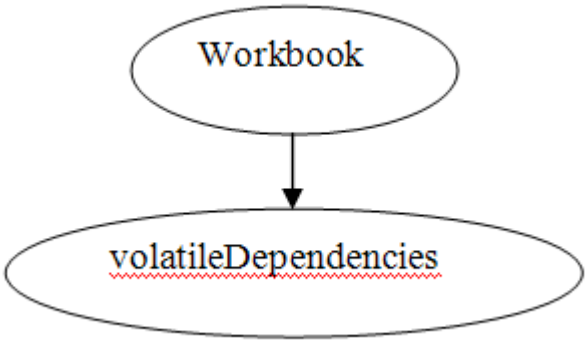
The volatileDependencies part provides a cache of data that supports Real Time Data (RTD) and CUBE functions in the workbook. Both of these types of functions require connectivity to external servers to retrieve their data. For RTD functions, an RTD interface defines how data is provided on the server, and how it is retrieved on the client. Similarly, CUBE functions access data in OLAP cubes via their own function syntax. The volatileDependencies part provides that cache of data and supporting information about these functions and their data servers and connections. This allows the spreadsheet application to work with cached values when recalculating the workbook when the external server is not available.

[*Note*: How users of SpreadsheetML access RTD data depends on the integration the user's spreadsheet application provides for RTD. *end note*]

[*Note*: Data connectivity can use a number of different technologies. One example of potential values stored in this attribute can be found at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbaxl11/html/xlobjIRtdServer_HV03085058.asp *end note*]

File Architecture

The workbook holds the relationship to the volatile dependencies part.



Illustration

The following image shows an example implementation of CUBE and Real Time Data (RTD) functions in a worksheet.

	A
1	=RTD("jrtdx.rtd",,"aaa")
2	
3	=CUBEMEMBER("xlextdat9 Adventure Works DW Adventure Works","[Department].[Departments].[Corporate]")
4	=CUBEVALUE("xlextdat9 Adventure Works DW Adventure Works",A3)
5	=CUBESET("xlextdat9 Adventure Works DW Adventure Works","[Customer].[Customer Geography].[All Customers].[United Kingdom].children","Set")
6	=CUBERANKEDMEMBER("xlextdat9 Adventure Works DW Adventure Works",$A$3,ROW(A1))
7	=CUBESETCOUNT(A3)
8	=CUBEMEMBERPROPERTY("xlextdat9 Adventure Works DW Adventure Works","[Product].[Product].[All Products].[Blade]","Class")
9	=CUBEKPIMEMBER("xlextdat9 Adventure Works DW Adventure Works","Growth in Customer Base",2)
10	

The following example shows the XML that describes the functions in the illustration.

[*Example*:

```
<volTypes xmlns="…">
  <volType type="realTimeData">
    <main first="jrtdx.rtd">
      <tp t="s">
        <v>aaa: 4447</v>
        <stp/>
        <stp>aaa</stp>
        <tr r="A1" s="1"/>
      </tp>
    </main>
  </volType>
  <volType type="olapFunctions">
    <main first="xlextdat9 Adventure Works DW Adventure Works">
      <tp t="e">
        <v>#N/A</v>
        <stp>1</stp>
        <tr r="A6" s="1"/>
        <tr r="A9" s="1"/>
        <tr r="A8" s="1"/>
        <tr r="A5" s="1"/>
        <tr r="A4" s="1"/>
        <tr r="A3" s="1"/>
      </tp>
    </main>
  </volType>
</volTypes>
```

end example]

While RTD and Cube functions share the cache, there are differences in how the data is interpreted. For example, RTD dependencies, volTypes/volType/main@first specifies the ProgId of the RTD server. Whereas for OLAP dependencies, main@first indicates the connection name.

3.15.1 main (Main)

Represents dependency information for all topics within a type that share the same first string or function argument.

Parent Elements
volType (§3.15.5)

Child Elements	Subclause
tp (Topic)	§3.15.3

Attributes	Description

first (First String)	Specifies the first string of all topics within this main. This string corresponds to the first argument to the RTD or CUBE function.
	
	For RTD functions, this argument represents the progID of the IRTDServer.
	
	[*Example*:
	`<main first="jrtdx.rtd">`
	end example]
	
	For CUBE functions, this argument represents the CUBE connection.
	
	[*Example*:
	`<main first="xlextdat9 Adventure Works DW Adventure Works">`
	end example]
	
	For more information on RTD and CUBE functions in SpreadsheetML, see §3.17 in Formulas.
	
	The possible values for this attribute are defined by the ST_Xstring simple type (§3.18.96).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_VolMain">
   <sequence>
      <element name="tp" type="CT_VolTopic" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="first" type="ST_Xstring" use="required"/>
</complexType>
```

3.15.2 stp (Strings in Subtopic)

Represents all strings in the topic except for the first. An stp is allocated for each additional argument. For example, for the topic {"progid","","foo"}, there would be two STPs: "" and "foo".

For Cube functions, value of "1" indicates that all of the related cells with calling cube functions have been refreshed.

The possible values for this element are defined by the ST_Xstring simple type (§3.18.96).

Parent Elements
tp (§3.15.3)

3.15.3 tp (Topic)

Represents dependency information for all topics within a volatile dependency type that share the same first string or argument.

For the RTD function, this collection will contain the remaining parameters of the function, and indicate the last known value and data type of that value.

Parent Elements
main (§3.15.1)

Child Elements	Subclause
stp (Strings in Subtopic)	§3.15.2
tr (References)	§3.15.4
v (Cell Value)	§3.3.1.93

Attributes	Description

t (Type)	Specifies the type of the cell value. This value corresponds to the type of data returned by the RTD or CUBE function.
	
	In the following RTD example, the value "aaa: 4447" has a string data type.
	
	[*Example*:
	`<tp t="s">`
	`  <v>aaa: 4447</v>`
	`</tp>`
	end example]
	
	For Cube functions, this attribute can be ignored when stp value is "1".
	
	The possible values for this attribute are defined by the ST_VolValueType simple type (§3.18.93).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_VolTopic">
   <sequence>
      <element name="v" type="ST_Xstring" minOccurs="1" maxOccurs="1"/>
      <element name="stp" type="ST_Xstring" minOccurs="0" maxOccurs="unbounded"/>
      <element name="tr" type="CT_VolTopicRef" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="t" type="ST_VolValueType" use="optional" default="n"/>
</complexType>
```

3.15.4 tr (References)

Represents the reference to a cell that depends on this topic. Each topic may have one or more cells dependencies.

For CUBE functions, each <tr> element contains a cell whose cube function call dependent on the connection in main@first.

Parent Elements
tp (§3.15.3)

Attributes	Description
r (Reference)	Specifies a reference to the cell location. The location is scoped to the sheet specified in s. [*Example*: `<tr r="A6" s="1"/>` *end example*] The possible values for this attribute are defined by the ST_CellRef simple type (§3.18.8).
s (Sheet Id)	Specifies the sheet index. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_VolTopicRef">
   <attribute name="r" type="ST_CellRef" use="required"/>
   <attribute name="s" type="xsd:unsignedInt" use="required"/>
</complexType>
```

3.15.5 volType (Volatile Dependency Type)

Represents dependency information for a specific type or external data server. There is no limit on the number of external dependencies that may exist for a workbook in SpreadsheetML.

Parent Elements
volTypes (§3.15.6)

Child Elements	Subclause
main (Main)	§3.15.1

Attributes	Description
type (Type)	Specifies the type of the external dependency. [*Example*: <volType type="olapFunctions"> *end example*] The possible values for this attribute are defined by the ST_VolDepType simple type (§3.18.92).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_VolType">
   <sequence>
      <element name="main" type="CT_VolMain" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="type" type="ST_VolDepType" use="required"/>
</complexType>
```

3.15.6 volTypes (Volatile Dependency Types)

Represents the collection of external dependencies for a workbook. This element defines the structure of the volatilateDependencies part. There can only be one volatileDependencies part for each workbook. However, the part may contain one or more dependency types.

The volatileDependencies part stores the following information for Real Time Data (RTD) and CUBE functions:

- Cached values
- Parameters used
- Connection and Server names

[*Example*: *Outline of XML Structure*

```
<volTypes xmlns="…">
  <volType type="realTimeData">
  </volType>
  <volType type="olapFunctions">
  </volType>
</volTypes>
```

end example]

Parent Elements
Root element of SpreadsheetML Volatile Dependencies part

Child Elements	Subclause

Child Elements	Subclause
extLst (Future Feature Data Storage Area)	§3.2.10
volType (Volatile Dependency Type)	§3.15.5

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_VolTypes">
   <sequence>
      <element name="volType" type="CT_VolType" minOccurs="1" maxOccurs="unbounded"/>
      <element name="extLst" minOccurs="0" type="CT_ExtensionList"/>
   </sequence>
</complexType>
```

3.16 Custom XML Mappings

Custom XML Mappings enable binding of arbitrary XML data structures and arbitrary XML schema definitions to the workbook. Once a DataBinding has been established, then various XML nodes can be mapped to table columns, ranges of cells, or even single cells (for non-repeating attributes and elements). Once an XML Mapping is fully defined, the application is able to import and export XML instance structures according to the schema definition.

While the original schema or XML definition may reside on disk or at some file location outside the workbook, a copy of the schema is stored in the workbook.

Every time an XML instance or schema is added to the workbook, a new map object is created which ties together the schemas and where the various elements are mapped in the workbook.

[*Example*:

```
<MapInfo SelectionNamespaces="">
  <Schema ID="Schema1">
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element nillable="true" name="Root">
        <xsd:complexType>
          <xsd:sequence minOccurs="0">
            <xsd:element minOccurs="0" nillable="true" name="EmployeeInfo"
              form="unqualified">
```

```
            <xsd:complexType>
              <xsd:sequence minOccurs="0">
                <xsd:element minOccurs="0" nillable="true" type="xsd:string"
                  name="Name" form="unqualified"></xsd:element>
                <xsd:element minOccurs="0" nillable="true" type="xsd:date"
                  name="Date" form="unqualified"></xsd:element>
                <xsd:element minOccurs="0" nillable="true" type="xsd:integer"
                  name="Code" form="unqualified"></xsd:element>
              </xsd:sequence>
            </xsd:complexType></xsd:element>

            <xsd:element minOccurs="0" maxOccurs="unbounded" nillable="true"
              name="ExpenseItem" form="unqualified">
              <xsd:complexType>
                <xsd:sequence minOccurs="0">
                  <xsd:element minOccurs="0" nillable="true" type="xsd:date"
                    name="Date" form="unqualified"></xsd:element>
                  <xsd:element minOccurs="0" nillable="true" type="xsd:string"
                    name="Description" form="unqualified"></xsd:element>
                  <xsd:element minOccurs="0" nillable="true" type="xsd:double"
                    name="Amount" form="unqualified"></xsd:element>
                </xsd:sequence>
              </xsd:complexType></xsd:element>
          </xsd:sequence>
          <xsd:attribute name="Currency" form="unqualified"
            type="xsd:string"></xsd:attribute>
          <xsd:attribute name="Approved" form="unqualified"
            type="xsd:string"></xsd:attribute>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </Schema>
  <Map ID="1" Name="Root_Map" RootElement="Root" SchemaID="Schema1"
    ShowImportExportValidationErrors="false" AutoFit="true" Append="false"
    PreserveSortAFLayout="true" PreserveFormat="true">
    <DataBinding ConnectionID="1" FileBinding="true" DataBindingLoadMode="1"/>
  </Map>
</MapInfo>
```

end example]

For XML mapped into a SpreadsheetML Table there will also be additional information in the SpreadsheetML file which refers back to the XML Map and XPath of the element or attribute mapped. This information is stored in a xmlColumnPr element, under the tableColumn node.

[*Example*:

```
<table xmlns="…" id="1" name="Table1" displayName="Table1" ref="A1:H11"
  tableType="xml" totalsRowShown="0" connectionId="1">
  <tableColumns count="5">
    <tableColumn id="1" uniqueName="Name" name="Name">
      <xmlColumnPr mapId="1" xpath="/Root/EmployeeInfo/Name"
        xmlDataType="string"/>
    </tableColumn>
    <tableColumn id="2" uniqueName="Date" name="Date">
      <xmlColumnPr mapId="1" xpath="/Root/EmployeeInfo/Date"
        xmlDataType="date"/>
    </tableColumn>
    <tableColumn id="3" uniqueName="Code" name="Code">
      <xmlColumnPr mapId="1" xpath="/Root/EmployeeInfo/Code"
        xmlDataType="integer"/>
    </tableColumn>
    <tableColumn id="4" uniqueName="Description" name="Description">
      <xmlColumnPr mapId="1" xpath="/Root/ExpenseItem/Description"
        xmlDataType="string"/>
    </tableColumn>
    <tableColumn id="5" uniqueName="Amount" name="Amount">
      <xmlColumnPr mapId="1" xpath="/Root/ExpenseItem/Amount"
        xmlDataType="double"/>
    </tableColumn>
  </tableColumns>
  <tableStyleInfo name="TableStyleMedium9" showFirstColumn="0"
    showLastColumn="0" showRowStripes="1" showColumnStripes="0"/>
</table>
```

end example]

For XML mapped into a single SpreadsheetML cell there will also be additional information in the TableSingleCells part which refers back to the XML Map and XPath of the element or attribute mapped. This information is stored in the xmlPr element under the xmlCellPr node.

[*Example*:

```
<singleXmlCells xmlns="…">
  <singleXmlCell id="2" name="Table2" displayName="Table2" r="D19"
    connectionId="1">
    <xmlCellPr id="1" uniqueName="Currency">
      <xmlPr mapId="1" xpath="/Root/@Currency" xmlDataType="string"/>
    </xmlCellPr>
  </singleXmlCell>
```

```
    <singleXmlCell id="3" name="Table3" displayName="Table3" r="D20"
      connectionId="1">
      <xmlCellPr id="1" uniqueName="Approved">
        <xmlPr mapId="1" xpath="/Root/@Approved" xmlDataType="string"/>
      </xmlCellPr>
    </singleXmlCell
    <singleXmlCell id="4" name="Table4" displayName="Table4" r="D18"
      connectionId="1">
      <xmlCellPr id="1" uniqueName="Name">
        <xmlPr mapId="1" xpath="/Root/EmployeeInfo/Name" xmlDataType="string"/>
      </xmlCellPr>
    </singleXmlCell>
  </singleXmlCells>
```

end example]

3.16.1 DataBinding (XML Mapping)

This element contains properties which specify how the XML mapping should work.

[*Example*:

```
  <DataBinding ConnectionID="1" FileBinding="true" DataBindingLoadMode="1"/>
```

end example]

Parent Elements
Map (§3.16.2)

Child Elements	Subclause
Any element from any namespace	n/a

Attributes	Description
ConnectionID (Reference to Connection ID)	Specifies the Connection ID to the external connection in the External Data Connections part. Required if FileBinding is true. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
DataBindingLoadMode (XML Data Loading Behavior)	Specifies the mode for loading XML data related to this DataBinding. Supported values are as follows:

Attributes	Description
	0 - None 1 - Normal 2 - Delay Load 3 - Asynchronous 4 - Object Model The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
DataBindingName (Unique Identifer)	Specifies the data binding name. These must be unique for each DataBinding. [*Example*: `<DataBinding DataBindingName="Binding1" FileBinding="true" FileBindingName="Binding1" DataBindingLoadMode="1"/>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
FileBinding (Binding to External File)	Specifies whether the data should be retrieved directly from an XML file. The path to the file is in the corresponding connection element The possible values for this attribute are defined by the XML Schema boolean datatype.
FileBindingName (File Binding Name)	Specifies the file binding name. These must be unique for each DataBinding. [*Example*: `<DataBinding DataBindingName="Binding1" FileBinding="true" FileBindingName="Binding1" DataBindingLoadMode="1"/>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DataBinding">
   <sequence>
      <any/>
   </sequence>
   <attribute name="DataBindingName" type="xsd:string" use="optional"/>
   <attribute name="FileBinding" type="xsd:boolean" use="optional"/>
   <attribute name="ConnectionID" type="xsd:unsignedInt" use="optional"/>
   <attribute name="FileBindingName" type="xsd:string" use="optional"/>
   <attribute name="DataBindingLoadMode" type="xsd:unsignedInt" use="required"/>
</complexType>
```

3.16.2 Map (XML Mapping Properties)

This element contains all of the properties related to the XML map, and the behaviors expected during data refresh operations.

[*Example*:

```
<Map ID="1" Name="Root_Map" RootElement="Root" SchemaID="Schema1"
  ShowImportExportValidationErrors="false" AutoFit="true" Append="false"
  PreserveSortAFLayout="true" PreserveFormat="true">
  <DataBinding ConnectionID="1" FileBinding="true" DataBindingLoadMode="1"/>
</Map>
```

end example]

Parent Elements
MapInfo (§3.16.3)

Child Elements	Subclause
DataBinding (XML Mapping)	§3.16.1

Attributes	Description
Append (Append Data to Table)	Specifies whether XML data should overwrite or be appended to the end of the table or range of mapped cells when data is refreshed. The possible values for this attribute are defined by the XML Schema boolean datatype.
AutoFit (AutoFit Table on Refresh)	Specifies whether columns should be resized to fit the XML data after a data refresh operation. The possible values for this attribute are defined by the XML Schema boolean datatype.
ID (XML Mapping ID)	Specifies the ID of the XML map. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
Name (XML Mapping Name)	Specifies the name of the XML map. The possible values for this attribute are defined by the XML Schema string datatype.
PreserveFormat (Preserve Cell Formatting)	Specifies whether cell number formatting in the sheet should be preserved during data refresh operations, or whether the number formatting defined by the XML data type should be used. The possible values for this attribute are defined by the XML Schema boolean datatype.
PreserveSortAFLayout (Preserve AutoFilter State)	Specifies whether to keep the filter state of the Table or cell range intact during a data refresh. The possible values for this attribute are defined by the XML Schema boolean datatype.
RootElement (Root	Specifies the names of the root XML element.

Attributes	Description
Element Name)	The possible values for this attribute are defined by the XML Schema string datatype.
SchemaID (Schema Name)	Specifies the unique name of the schema used for the mapping. The possible values for this attribute are defined by the XML Schema string datatype.
ShowImportExportValidationErrors (Show Validation Errors)	Specifies whether validation errors should be displayed during data refresh or data export. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Map">
   <sequence>
      <element name="DataBinding" type="CT_DataBinding" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="ID" type="xsd:unsignedInt" use="required"/>
   <attribute name="Name" type="xsd:string" use="required"/>
   <attribute name="RootElement" type="xsd:string" use="required"/>
   <attribute name="SchemaID" type="xsd:string" use="required"/>
   <attribute name="ShowImportExportValidationErrors" type="xsd:boolean" use="required"/>
   <attribute name="AutoFit" type="xsd:boolean" use="required"/>
   <attribute name="Append" type="xsd:boolean" use="required"/>
   <attribute name="PreserveSortAFLayout" type="xsd:boolean" use="required"/>
   <attribute name="PreserveFormat" type="xsd:boolean" use="required"/>
</complexType>
```

3.16.3 MapInfo (XML Mapping)

This element acts as the container for all of the XML schemas and maps attached to the SpreadsheetML document.

Parent Elements
Root element of SpreadsheetML Custom XML Mappings part

Child Elements	Subclause
Map (XML Mapping Properties)	§3.16.2
Schema (XML Schema)	§3.16.4

Attributes	Description
SelectionNamespaces (Prefix Mappings for XPath Expressions)	Specifies namespaces for use in XPath expressions when it is necessary to define new namespaces externally. Namespaces are defined in the XML style, as a space-separated list of namespace declaration attributes

Attributes	Description

	[*Example*: The following example contains elements that belong to "a" and "b", in addition to elements that do not belong to any namespace.
	
	`<?xml version="1.0"?>`
	`<root>`
	`<branch>branch</branch>`
	`<a:root xmlns:a="http://myserver.com">`
	`<a:branch>a-branch</a:branch>`
	`<b:branch xmlns:b="http://yourserver.com">`
	`b-branch</b:branch>`
	`</a:root>`
	`</root>`
	end example]
	
	Note: This is used when writing Xpath expressions at runtime against the XML instance structures, because the Xpath expressions use namespace prefixes instead of the fully spelled out namespace.
	
	The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MapInfo">
   <sequence>
      <element name="Schema" type="CT_Schema" minOccurs="1" maxOccurs="unbounded"/>
      <element name="Map" type="CT_Map" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="SelectionNamespaces" type="xsd:string" use="required"/>
</complexType>
```

3.16.4 Schema (XML Schema)

This element contains the XML tree for an attached schema.

Parent Elements
MapInfo (§3.16.3)

Child Elements	Subclause
Any element from any namespace	n/a

Attributes	Description
ID (Schema ID)	Specifies the unique name or ID for this attached schema. [*Example*: `ID = "Schema1"` *end example*]

Attributes	Description
	The possible values for this attribute are defined by the XML Schema string datatype.

Namespace (Schema Root Namespace)	Specifies the namespace used by the schema.
	
	[*Example*:
	
	`<MapInfo SelectionNamespaces="…">`
	`  <Schema ID="Schema1" Namespace="…">`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

SchemaRef (Schema Reference)	The schemaRef attribute is used in the specific case where the schema definition happens to include another XSD that contributes to the same namespace. The value of this attribute is the relative path to a "root" XSD on disk which in turn references the other XDS files contributing type definitions to the same namespace. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Schema">
   <sequence>
      <any/>
   </sequence>
   <attribute name="ID" type="xsd:string" use="required"/>
   <attribute name="SchemaRef" type="xsd:string" use="optional"/>
   <attribute name="Namespace" type="xsd:string" use="optional"/>
</complexType>
```

3.17 Formulas

3.17.1 Introduction

A SpreadsheetML *formula* is an equation that performs a calculation that typically involves the values of one or more cells in one or more worksheets.

A formula is an expression that can contain the following: constants, operators, cell references, calls to functions, and names.

[*Example*: Consider the formula `PI()*(A2^2)`. In this case,

- `PI()` results in a call to the function `PI`, which returns the value of π.
- The cell reference A2 returns the value in that cell.
- 2 is a numeric constant.
- The caret (^) operator raises its left operand to the power of its right operand.
- The parentheses, `(` and `)`, are used for grouping.

- The asterisk (*) operator performs multiplication of its two operands.

end example]

3.17.2 Syntax

The general syntax of a formula is as follows:

formula:
expression

expression:
(*expression*)
constant
prefix-operator *expression*
expression *infix-operator* *expression*
expression *postfix-operator*
cell-reference
function-call
name

where *expression* is an arbitrarily complex expression involving constants (§3.17.2.1), operators (§0), cell references (§3.17.2.3), calls to functions (§3.17.2.4), and names (§3.17.2.5).

A *token* is the minimal lexical element of a formula. The categories of tokens are: constants (except for *array-constant*), operators, cell references, function names, names, and punctuators. The *punctuator*s are:

- Left parenthesis (`(`) and right parenthesis (`)`) used for expression grouping and in a function call.
- Comma (`,`) used in a function call and an *array-constant*.
- Left brace (`{`), right brace (`}`), and semicolon (`;`) used in an *array-constant*.

In a formula, an arbitrary number of space characters (`U+0020`) can precede the first token or follow the final token. An arbitrary number of space characters can separate two adjacent tokens, except that no space characters shall separate a *function-name* from the left parenthesis (`(`) that follows it. Such space characters have no effect on the semantics of a formula; however, such spaces shall be distinguished from the space operator (§0).

All arithmetic terms in an *expression* are real numbers. [*Example*: In the expression `1/3`, although the operands appear to be integers, they are, in fact, real numbers, and the result is `0.33`…, not `0`, as would result from integer division. *end example*]

As ranges of data are fundamental to spreadsheet calculations, many SpreadsheetML functions are able to take arrays as inputs and to return arrays as outputs. The way in which formula return values are returned into the worksheet differs based on whether the formula in a given cell was *array entered* or not, but intermediate calculations are always done for the full arrays when they are used as inputs. The differences with an array-entered formula are:

- No implicit intersection is performed on cell range inputs.

- The results of the formula's calculation can be returned across multiple cells in the sheet.

When a range of cell references is used in a formula that is array-entered in an area larger than that range, the excess cells take on a value of #N/A.

[*Example*: Here are some formulas taking array constants and ranges:

- (B2:B4*C2:C4)+10.5 performs three calculations: (B2*C2)+10.5, (B3*C3)+10.5, and (B4*C4)+10.5.
- SQRT({1,2,3,4}) returns 1 when entered normally.
- SQRT({1,2,3,4}) returns 1 when array-entered into a single cell, but if it's array-entered in four or more cells in a contiguous row, it will return 1, 1.41, 1.73, and 2 in the first four cells, respectively, and #N/A in any additional cells in the horizontal range for which it was array-entered. (For display purposes, the values returned have been truncated to two decimal places.)
- SUM(SQRT({1,2,3,4})) returns 6.14 when entered normally, since array calculations are always performed by the SQRT function, and the array output is understood as a valid input by the SUM function.

With A1:A4 holding the values 1, 2, 3, and 4, respectively:

- SQRT(A1:A4) entered normally will do implicit intersection if it is in any of the rows 1–4, and return the SQRT of the number in the same row.
- SQRT(A1:A4) returns 1 when array-entered into a single cell, since it does not do implicit intersection in this case. If it's array-entered in multiple cells in a contiguous column, it will return 1, 1.41, 1.73, 2, #N/A, …, respectively, in the cells in its vertical output range. *end example*]

3.17.2.1 Constants

A *constant* is a predefined value that is not calculated, and, therefore, does not change. A constant has the following form:

constant:
error-constant
logical-constant
numerical-constant
string-constant
array-constant

error-constant:
#DIV/0! | #N/A | #NAME? | #NULL! | #NUM! | #REF! | #VALUE!

logical-constant:
FALSE
TRUE

numerical-constant:
whole-number-part [.] [*exponent-part*]
. *fractional-part* [*exponent-part*]
whole-number-part . *fractional-part* [*exponent-part*]

whole-number-part:
digit-sequence

fractional-part:
digit-sequence

exponent-part:
e [*sign*] *digit-sequence*
E [*sign*] *digit-sequence*

sign:
+
–

digit-sequence:
a series of one or more decimal digits

string-constant:
" [*string-chars*] "

string-chars
string-char
string-chars string-char

string-char
""
any character except "

To include a double-quote character (") in *string-chars*, precede it with another double-quote character. [*Example*: "ab""cd" contains the characters ab"cd, and """abcd""" contains the characters "abcd". *end example*]

An *array constant* is a list of one or more constants organized in one or two dimensions, and delimited by braces. An array constant has the following form:

array-constant:
{ *constant-list-rows* }

constant-list-rows:
constant-list-row
constant-list-rows ; *constant-list-row*

constant-list-row:
constant
constant-list-row , *constant*

An *array-constant* shall not contain

- An *array-constant*.
- Columns or rows of unequal length.

Any *numerical-constant* in an *array-constant* can be preceded immediately by a *prefix-operator*.

The *constant*s in an *array-constant* can have different types.

[G*uidance* An implementation is encouraged to not unnecessarily limit the number of rows and columns in an *array-constant*. *end guidance*]

[*Example*: `{1,3.5,TRUE,"Hello"}` is a 1x4 array of constants.

To represent the values 10, 20, 30, and 40, as a 1x4 array, use `{10,20,30,40}`.

To represent the values 10, 20, 30, and 40 in the first row, and 50, 60, 70, and 80 in the second row, use the following 2x4 array constant: `{10,20,30,40;50,60,70,80}`. *end example*]

error-constant is described in §3.17.2.7.

Each *constant* has a corresponding type (§3.17.2.6), as follows:

Constant Form	Type
array-constant	array
error-constant	error
logical-constant	logical
numerical-constant	number
string-constant	text

In the context of cell formulas and values in SpreadsheetML, the following definition of precision shall apply:

By default, default representation of precision shall be as defined by the XML schema double type http://www.w3.org/TR/xmlschema-2/#double. The default is therefore 53-bits of mantissa precision.

An application that uses XML schema double can optionally state the precision in the Additional Characteristics part by writing out the number of bits in the mantissa and exponent.

A compliant consumer shall parse numbers of arbitrary precision without error.

3.17.2.2 Operators

An *operator* is a symbol that specifies the type of operation to perform on one or more operands. There are arithmetic, comparison, text, and reference operators.

infix-operator:
: | , | space | ^ | * | / | + | - | & | = | <> | < | <= | > | >=

postfix-operator:
%

prefix-operator:
-

The *operator*s permitted in *expression* are:

Operators			
Family	Operator	Description	Precedence

Operators			
Reference operators	:	Binary range operator, which takes two cell reference (§3.17.2.3) operands, and results in one reference to the cells inclusive of, and between, those references. [*Example*: `SUM(B5:C15)`, which references 11 cells. *end example*]	highest
	,	Binary union operator, which takes two cell reference (§3.17.2.3) operands, and results in one reference to all those, possibly non-contiguous, cells. [*Example*: `SUM((B5:B15,D5:D15))`, which references 22 cells, 11 from column B, and 11 from column D. The grouping parentheses are necessary to indicate that the comma is an operator rather than a punctuator separating two arguments. *end example*]	
	space	Binary intersection operator, which takes two cell reference (§3.17.2.3) operands, and results in one reference to those, possibly non-contiguous, cells that are common. If the intersection is empty, the result is `#NULL!`. [*Example*: `COUNT((B1:C1) (C1:D1))`, which results in a reference to `C1`, while `COUNT((B1:D1) (B1,D1))` results in a single reference to `B1` and `D1`. *end example*]	
Arithmetic operators	-	Unary minus	
	%	Percentage (unary postfix), which divides its operand by 100. [*Example*: `10.5%`, which results in `0.105`. *end example*]	
	^	Exponentiation	
	*	Multiplication	
	/	Division	
	+	Addition	
	-	Subtraction	
Text operator	&	Text concatenation (Each of the two operands is converted to text, if necessary, before concatenation.)	
Comparison operators	=	Equal-to	lowest
	<>	Not-equal-to	
	<	Less-than	

Operators			
	<=	Less-than or equal-to	
	>	Greater-than	
	>=	Greater-than-or-equal-to	

expression can contain grouping parentheses to document the default precedence or to override it.

*operator*s in *expression* having the same precedence associate left-to-right.

[*Example*: Given that cell E38 contains the value 4, and cell F38 contains the value 2, the formula

```
((-1+E38^2)*3-F38)/2
```

produces the result 21.5. *end example*]

The comparison operators yield TRUE for true and FALSE for false. An expression with value 0 tests logically false while one with any non-zero value tests true.

For any given operator in an expression, if only one operand is an error value, the result is that error value. If more than one operand has an error value and those error values are the same, the result is that error value. If more than one operand has an error value and those error values are not all the same, as to which of those error values is used as the result is unspecified.

It the semantics of an operator having a given operand are not specified by this Standard, the result is #VALUE!. [*Example*: "abc"+1 results in #VALUE!, and "abc"/0 results in #VALUE! rather than #DIV/0!. *end example*]

3.17.2.3 Cell References

Each set of horizontal cells in a worksheet is a *row*, and each set of vertical cells is a *column*. A cell's row and column combination designates the location of that cell. [G*uidance* An implementation is encouraged to not unnecessarily limit the number of rows and columns in a worksheet. *end guidance*]

A *cell reference* designates one or more cells on the same worksheet. Using references, one can:

- Use data contained in different parts of the same worksheet in a single formula.
- Use the value from a single cell in several formulas.
- Refer to cells on other sheets in the same workbook, and even to other workbooks. (References to cells in other workbooks are called *links*.)

A cell reference has the following form:

cell-reference:
name
[[[[[*workbook-name*]] [*sheet-name* :]] *sheet-name* !] *A1-reference*
[[[[[*workbook-name*]] [*sheet-name* :]] *sheet-name* !]
A1-reference : *A1-reference*
[[[[[*workbook-name*]] [*sheet-name* :]] *sheet-name* !] *R1C1-reference*
[[[[[*workbook-name*]] [*sheet-name* :]] *sheet-name* !]
R1C1-reference : *R1C1-reference*

workbook-name:
book-name-start-character [*book-name-characters*]

book-name-start-character:
any character except ', *, [,], :, and ?

book-name-characters:
book-name-characters book-name-character

book-name-character:
any character except *, [,], :, and ?

sheet-name:
sheet-name-start-character [*sheet-name-characters*]

sheet-name-start-character:
any character except ', *, [,], \, :, /, and ?

sheet-name-characters:
sheet-name-characters sheet-name-character

sheet-name-character:
any character except *, [,], \, :, /, and ?

A *relative cell reference* is based on the relative position of the cell that contains the formula and the cell to which the reference refers. If the position of the cell that contains the formula changes, the reference is changed along with it.

An *absolute cell reference* always refers to the absolute location of a cell. If the position of the cell that contains the formula changes, the absolute reference remains the same.

A *mixed cell reference* has either an absolute column and relative row, or an absolute row and relative column.

It is possible to process the same cell or set of cells on multiple worksheets within a workbook, using a *3-D reference*. A reference of this type is made up of the cell reference, preceded by a range of worksheet names, and an exclamation mark character (!), in that order. A 3-D reference can be used to refer to cells on other sheets, to defined names, and to create formulas by using the following functions: AVERAGE, AVERAGEA, COUNT, COUNTA, MAX, MAXA, MIN, MINA, PRODUCT, STDEV, STDEVA, STDEVP, STDEVPA, SUM, VAR, VARA, VARP, and VARPA.

3-D references shall not be used in multi-cell formulas.

By default, a cell reference is understood to refer to one or more cells in the current worksheet. However, a cell reference can be preceded by its parent worksheet name and an exclamation mark (!), in that order. This allows

cells in one worksheet to be referenced in another worksheet of the same workbook. [*Example*: The cell reference `MonthlyTotals!D1:D12` might be used from within a sibling (or the same) worksheet of `MonthlyTotals` to refer to those 12 cells. *end example*]

An *area* is a set of rectangular-shaped contiguous cells. An area can be a single cell. [*Example*: `A5` and `B6:C10` each designate one area, and `D3:D5,E12:F15` designates two areas (the comma (`,`) being the union operator). *end example*] [*Note*: The number of areas designated by a cell reference can be obtained by calling the function AREAS (§3.17.7.10). *end note*]

There are two cell reference styles: A1 (§3.17.2.3.1) and R1C1 (§3.17.2.3.2).

3.17.2.3.1 A1-Style Cell References

A cell reference using the A1 reference style has the following form:

A1-reference:
A1-column
A1-row
A1-column *A1-row*

A1-column:
A1-relative-column
A1-absolute-column

A1-relative-column:
A Latin letter A–Z
The Latin letters AA–AZ, BA–BZ, …, ZA–ZZ, AAA-AAZ, ABA–ABZ, …, and so on

A1-absolute-column:
$ *A1-relative-column*

A1-row:
A1-relative-row
A1-absolute-row

A1-relative-row:
A positive decimal number

A1-absolute-row:
$ *relative-row*

In this style, each row has a numeric heading numbered sequentially from the top down, starting at 1. Each column has an alphabetic heading named sequentially from left-to-right, A–Z, then AA–AZ, BA–BZ, …, ZA–ZZ, AAA–AAZ, ABA–ABZ, and so on. Column letters are not case-sensitive.

A relative reference to a single cell is written as its column letter immediately followed by its row number. A relative reference to a whole row is written as its row number. A relative reference to a whole column is written as its column letter. A reference to a range of two or more cells is written as two single-cell references separated by the binary range operator (`:`). An absolute A1 reference is made up of a cell's column letter followed by its row number, with each being preceded by a dollar character (`$`). [*Example*: `A2`, `B34`, and `B5:D8` are relative A1 references. `$A$2`, `$B$34`, and `$B$5:$D$8` are absolute A1 references. `$A2`, `B$34`, and `$B5:D$8` are mixed A1 references. *end example*]

[*Example*: `SUM(Sheet2:Sheet13!B5)` adds all the values contained in cell B5 on all the worksheets between and including Sheet2 and Sheet13. *end example*]

For rules on how deal with potential ambiguities between cell references and defined names, see §3.17.5.1.

3.17.2.3.2 R1C1-Style Cell Reference

A cell reference using the R1C1 reference style has the following form:

R1C1-reference:
R1C1-row
R1C1-column
R1C1-row *R1C1-column*

R1C1-row:
R1C1-relative-row
R1C1-absolute-row

R1C1-relative-row:
R [*R1C1-relative-number*]

R1C1-absolute-row:
R
R *R1C1-absolute-number*

R1C1-column:
R1C1-relative-column
R1C1-absolute-column

R1C1-relative-column:
C [*R1C1-relative-number*]

R1C1-absolute-column:
C
C *R1C1-absolute-number*

R1C1-relative-number:
An optionally signed decimal number

R1C1-absolute-number:
A positive decimal number

In this style, each row has a numeric heading numbered sequentially from the top down, starting at 1. Each column has a numeric heading numbered sequentially from left-to-right, starting at 1.

A whole row is referenced by omitting the column, and a whole column is referenced by omitting the row. An absolute row or column reference uses absolute row or column numbers, respectively. A relative row or column reference uses, respectively, row or column offsets from the cell containing the formula, with a negative offset indicating a row to the left or a column above, and a positive offset indicateing a row to the right or a column below. Specifying an offset of zero is equivalent to omitting that offset and its delimiting brackets. [*Example*: `R[-2]C` refers to the cell two rows up and in the same column, `R[2]C[2]` refers to the cell two rows down and two columns to the right, R2C2 refers to the cell in the second row and in the second column, `R[-1]` refers to the entire row above the active cell, and R refers to the current row. *end example*]

The R1C1 alternate reference style can only be used at runtime. See §3.17.6.1 for XML-related details.

3.17.2.4 Functions

A *function* is a named formula that takes zero or more arguments, performs an operation, and, optionally, returns a result. A function call has the following form:

function-call:
function-name ([*argument-list*])

function-name:
predefined-function-name
user-defined-function-name

predefined-function-name:
ABS | ACOS | ACOSH | any of the other functions defined in §3.17.7

user-defined-function-name:
letter [*user-defined-name-characters*]

user-defined-name-characters:
user-defined-name-characters user-defined-name-character

user-defined-name-character:
letter
any decimal digit 0–9
.

letter:
any Latin letter A–Z, a–z

argument-list:
argument
argument-list , *argument*

argument:
expression

*predefined-function-name*s and *user-defined-function-name*s are not case-sensitive.

A *user-defined-function-name* shall not have any of the following forms:

- TRUE or FALSE
- *name*
- *cell-reference*

[*Guidance*: An implementation is encouraged to support *user-defined-function-name*s at least as long as 255 characters. *end guidance*]

The semantics of a call to a function having a *user-defined-function-name* are unspecified.

[*Example*: Here are some function calls: `PI(), POWER(A1,B3)`, and `SUM(C6:C10)`. *end example*]

An argument to a function can be a call to a function. That is, function calls can nest. [G*uidance* An implementation is encouraged to support at least 64 levels of nested function calls. *end guidance*]

Some functions take a variable number of arguments. This is indicated in the Syntax sections of §3.17.7 by their having *argument-list* as all, or the trailing part, of their argument list. The total number of arguments that shall be passed to such functions is at least 1.

[*Guidance* An implementation is encouraged to support function calls having at least 255 arguments. *end guidance*]

Expressions can have one or more values. Scalar expressions designate a single value, and cell references and array constants can designate multiple values. In the case of a multi-value expression, the way in which this is handled by a function when passed as an argument depends on a number of factors.

Most functions and operators expect either single- or multi-valued arguments and perform all of the array calculations whenever multi-valued arguments are present. [*Example*: `SQRT({1;2;3;4})`; see the examples in §3.17.2. *end example*]

When a function expects a single-valued argument but a multi-valued expression is passed, an attempt can be made to convert that set of values to a single value. For an array value or constant, the value of the expression is the value of the first element within that array value or constant. For a cell range, the first element can be used, or implicit intersection can be performed—the exact behavior is unspecified.

When a function expects a multi-valued argument but a single-valued expression is passed, that single-valued argument is treated as a 1x1 array.

For rules on how deal with potential ambiguities between function names and defined names, see §3.17.5.1.

3.17.2.5 Names

A *name* is an alias for a constant, a cell reference, or a formula. [*Note*: A name in a formula can make it easier to understand the purpose of that formula. For example, the formula `SUM(FirstQuarterSales)` is easier to identify than `SUM(C20:C30)`. *end note*]

Here is the syntax for *name*:

name:
[*workbook-name* !] *name-start-character* [*name-characters*]

name-start-character:
letter
_
\

name-characters:
name-characters name-character

name-character:
letter
any decimal digit 0–9
_
.

*name*s are not case-sensitive.

All *name*s within a workbook shall be unique. If the same *name*s are defined in two workbooks, both *name*s can be used in the same context be prefixing them with their corresponding workbook name and an exclamation mark (`!`). [*Example*: `SUM(Sales.xlsx!ProjectedSales)` refers to the named range `ProjectedSales` in the workbook named `Sales.xlsx`. *end example*]

A *name* shall not have any of the following forms:

- TRUE or FALSE
- *user-defined-function-name*
- *cell-reference*

[*Guidance* An implementation is encouraged to support *name*s at least as long as 255 characters. *end guidance*]

For rules on how deal with potential ambiguities between function names and defined names, or between cell references and defined names, see §3.17.5.1.

3.17.2.6 Types and Values

Each *expression* has a type. SpreadsheetML formulas support the following types: array, error, logical, number, and text.

An array value or constant represents a collection of one or more elements, whose values can have any type (i.e., the elements of an array need not all have the same type).

An error value (§3.17.2.7) or constant represents an error, and can have any value defined for *error-constant* (§3.17.2.1).

A logical value or constant represents a truth value, and can have any value defined for *logical-constant* (§3.17.2.1).

A numeric value or constant represents a real number, and can have any value defined for *numeric-constant* (§3.17.2.1). The term "number" is used as a generic name for any expression of type *number*.

A text value or constant represents arbitrary text, and can have any value defined for *string-constant* (§3.17.2.1). The term "string" is used as a generic name for any expression of type *text*.

An implementation is permitted to provide an implicit conversion from *string-constant* to number. However, the rules by which such conversions take place are implementation-defined. [*Example*: An implementation might choose to accept "123"+10 by converting the string "123" to the number 123. Such conversions might be locale-specific in that a *string-constant* such as "10,56" might be converted to 10.56 in some locales, but not in others, depending on the radix point character. *end example*]

[*Guidance* An implementation is encouraged to support strings at least as long as 32,767 characters. *end guidance*]

A complex number is represented as a string in one of two equivalent text formats: *x* + *y*`i` or *x* + *y*`j`, where *x* is the real part, and *y* is the imaginary part. [*Example*: `"3+4i"` and `"-2.5-34.6j"` *end example*]

3.17.2.7 Single- and Multi-Cell Formulas

A single-cell formula is applied to a single cell while a multi-cell formula is applied to a range of cells as a group.

When a single-cell formula results in a single value, the designated cell takes on that value. [*Example*: When cell `A10` contains `SIN(0.3)`, the result stored in that cell is `0.295520207`. *end example*]

When a multi-cell formula results in a single value, each of the designated cells takes on that value. [*Example*: When the group of cells `A10:A12` contains `SIN(0.3)`, the result stored in each of those cells is `0.295520207`. *end example*]

When a single-cell formula results in multiple values, the designated cell takes on the first of those values. [*Example*: When cell `A10` contains `SIN({0.3,0.4,0.5})`, the result stored in that cell is `0.295520207` (`SIN(0.3)`). *end example*]

When a multi-cell formula results in multiple values (such as when the multi-cell formula is array-entered), the designated cells take on corresponding values, according to the shape of the cell group and the values. Specifically,

- If the cell group and values have the same shape (i.e., the same number of rows and columns), each cell takes on the value corresponding to its relative position.
- If the cell group has fewer columns than the values, the left-most columns of the values are stored in the cells.
- If the cell group has fewer rows than the values, the top-most rows of the values are stored in the cells.
- If the cell group has more columns than the values, each cell takes on the value corresponding to its relative position, except that
    - For a cell group 1x*N* array or a two-dimensional array, the excess right-most cells take on an unspecified value.
    - For a cell group *N*x1 array, the excess columns are clones of the first column.
- If the cell group has more rows than the values, each cell takes on the value corresponding to its relative position, except that:
    - For a cell group *N*x1 array or a two-dimensional array, the excess bottom-most cells take on an unspecified value.
    - For a cell group 1x*N* array, the excess rows are clones of the first row.

[*Example*: Case 1: The 1x3 group of cells `A20:C20` has applied to it the formula `SIN({0.3,0.4,0.5})`. The number of rows and columns in the group exactly matches the number of rows and columns in the result. Those cells then contain 0.295520207, 0.389418342, and 0.479425539, which correspond to `SIN(0.3)`, `SIN(0.4)`, and `SIN(0.5)`, respectively.

Case 2: The 1x2 group of cells A20:B20 has applied to it the formula SIN({0.3,0.4,0.5}). The number of columns in the group is less than the number of columns in the result. (The number of rows is the same in each.) Those cells then contain 0.295520207 and 0.389418342, which correspond to SIN(0.3) and SIN(0.4), respectively, the left-most part of the set of values.

Case 3: The 1x4 group of cells A20:D20 has applied to it the formula SIN({0.3,0.4,0.5}). The number of columns in the group is greater than the number of columns in the result. (The number of rows is the same in each.) Those cells then contain 0.295520207, 0.389418342, 0.479425539, and an unspecified value, which correspond to SIN(0.3), SIN(0.4), and SIN(0.5), respectively, with the fourth value being unspecified.

Case 4: The 2x2 group of cells A30:B31 has applied to it the formula SIN({0.1,0.2,0.3}). The number of columns in the group is less than the number of columns in the result. As a result, the cells in row 30 contain 0.295520207 and 0.389418342, which correspond to SIN(0.3) and SIN(0.4), respectively. The number of rows in the group is greater than the number of rows in the result, so the cells in 31 are a copy of the cells in row 30. The left-most part of the set of values is propagated into the cells.

Case 5: The 2x2 group of cells A40:B41 has applied to it the formula SIN({0.1,0.2,0.3; 0.4,0.5,0.6; 0.7,0.8,0.9}). The number of columns in the group is less than the number of columns in the result. As a result, the left-most column values are stored. The number of rows in the group is less than the number of rows in the result. As a result, the top-most column values are stored. *example*]

3.17.3 Error values

The evaluation of an expression can result in an error having one of a number of *error value*s. These error values are:

Error Value	Reason for Occurrence
#DIV/0!	Intended to indicate when any number, including zero, is divided by zero. [*Note*: However, any error code divided by zero results in that error code. *end note*]
#N/A	Intended to indicate when a designated value is not available. [*Example*: Some functions, such as SUMX2MY2, perform a series of operations on corresponding elements in two arrays. If those arrays do not have the same number of elements, then for some elements in the longer array, there are no corresponding elements in the shorter one; that is, one or more values in the shorter array are not available. *end example*] This error value can be produced by calling the function NA (§3.17.7.221).
#NAME?	Intended to indicate when what looks like a name is used, but no such name has been defined. [*Example*: XYZ/3, where XYZ is not a defined name. Total is & A10, where neither Total nor is is a defined name. Presumably, "Total is " & A10 was intended. SUM(A1C10), where the range A1:C10 was intended. *end example*]
#NULL!	Intended to indicate when two areas are required to intersect, but do not. [*Example*: In the case of SUM(B1 C1), the space between B1 and C1 is treated as the binary intersection operator, when a comma was intended. *end example*]
#NUM!	Intended to indicate when an argument to a function has a compatible type, but has a

Error Value	Reason for Occurrence
	value that is outside the domain over which that function is defined. (This is known as a *domain error*.) [*Example*: Certain calls to ASIN, ATANH, FACT, and SQRT might result in domain errors. *end example*] Intended to indicate that the result of a function cannot be represented in a value of the specified type, typically due to extreme magnitude. (This is known as a *range error*.) [*Example*: FACT(1000) might result in a range error. *end example*]
#REF!	Intended to indicate when a cell reference is invalid. [*Example*: If a formula contains a reference to a cell, and then the row or column containing that cell is deleted, a #REF! error results. If a worksheet does not support 20,001 columns, OFFSET(A1,0,20000) will result in a #REF! error. *end example*]
#VALUE!	Intended to indicate when an incompatible type argument is passed to a function, or an incompatible type operand is used with an operator. [*Example*: In the case of a function argument, text was expected, but a number was provided *end example*]

Each error value has a corresponding *error-constant* (§3.17.2.1).

[*Note*: A number of functions operate on error values: They include ERROR.TYPE (§3.17.7.109), ISERR (§3.17.7.174), ISERROR (§3.17.7.175), and ISNA (§3.17.7.178). *end note*]

3.17.4 Dates and Times

Each unique instant in SpreadsheetML time is represented as a distinct non-negative numeric *serial value*, which is made up of an integer date component and a fractional time component. As dates and times are numeric values, they can take part in arithmetic operations.

Numerous functions take as arguments one or more serial values or strings representing dates and/or times. Functions that care only about the date shall ignore any time information that is provided. Functions that care only about the time shall ignore any date information that is provided.

3.17.4.1 Date Representation

Going forward in time, the date component of a serial value increases by 1 each day.

There are two different bases for serial values:

- In the *1900 date base system*, the lower limit is January 1, 1900, which has serial value 1. The upper-limit is December 31, 9999, which has serial value 2,958,465.
- In the *1904 date base system*, the lower limit is January 1, 1904, which has serial value 0. The upper-limit is December 31, 9999, which has serial value 2,957,003.

A serial value outside of the range for its date base system is ill-formed.

As to which date base system an implementation uses by default or whether it allows its users to switch between date base systems, is unspecified. See §3.17.6.7 for XML-related details. [*Note*: As the XML allows either date base system to be used, an implementation must be able to deal with both systems. *end note*]

For legacy reasons, an implementation using the 1900 date base system shall treat 1900 as though it was a leap year. [*Note*: That is, serial value 59 corresponds to February 28, and serial value 61 corresponds to March 1, the next day, allowing the (non-existent) date February 29 to have the serial value 60. *end note*] A consequence of this is that for dates between January 1 and February 28, `WEEKDAY` shall return a value for the day immediately prior to the correct day, so that the (non-existent) date February 29 has a day-of-the-week that immediately follows that of February 28, and immediately precedes that of March 1.

[*Example*: For the 1900 date base system:

`DATEVALUE("01-Jan-1900")` results in the serial value `1.0000000`…
`DATEVALUE("03-Feb-1910")` results in the serial value `3687.0000000`…
`DATEVALUE("01-Feb-2006")` results in the serial value `38749.0000000`…
`DATEVALUE("31-Dec-9999")` results in the serial value `2958465.0000000`…

For the 1904 date base system:

`DATEVALUE("01-Jan-1904")` results in the serial value `0.0000000`…
`DATEVALUE("03-Feb-1910")` results in the serial value `2225.0000000`…
`DATEVALUE("01-Feb-2006")` results in the serial value `37287.0000000`…
`DATEVALUE("31-Dec-9999")` results in the serial value `2957003.0000000`…

end example]

3.17.4.2 Time Representation

The time component of a serial value ranges in value from 0–0.99999999, and represents times from 0:00:00 (12:00:00 AM) to 23:59:59 (11:59:59 P.M.), respectively.

Going forward in time, the time component of a serial value increases by 1/86,400 each second. [*Note*: As such, the time 12:00 has a serial value time component of 0.5. *end note*]

[*Example*:

`TIMEVALUE("00:00:00")` results in the serial value `0.0000000`…
`TIMEVALUE("00:00:01")` results in the serial value `0.0000115`…
`TIMEVALUE("10:05:54")` results in the serial value `0.4207639`…
`TIMEVALUE("12:00:00")` results in the serial value `0.5000000`…
`TIMEVALUE("23:59:59")` results in the serial value `0.9999884`…

end example]

3.17.4.3 Combined Date and Time Representation

Any date component can be added to any time component to produce a serial value for that date/time combination.

[*Example*: For the 1900 date base system:

`DATE(1910,2,3)+TIME(10,5,54)` results in the serial value `3687.4207639…`
`DATE(1900,1,1)+TIME(12,0,0)` results in the serial value `1.5000000…`
`DATE(9999,12,31)+TIME(23,59,59)` results in the serial value `2958465.9999884…`

For the 1904 date base system:

`DATE(1910,2,3)+TIME(10,5,54)` results in the serial value `2225.4207639…`
`DATE(1904,1,1)+TIME(12,0,0)` results in the serial value `0.5000000…`
`DATE(9999,12,31)+TIME(23,59,59)` results in the serial value `2957003.9999884…`

end example]

3.17.5 Limits and Precision

3.17.5.1 Limits

In SpreadsheetML, cell references range from column A1–A1048576 (column A:A) to column XFD1–XFD1048576 (column XFD:XFD).

An implementation can extend this range. However, to avoid ambiguities, it is necessary to ensure that defined names are distinct from cell references, or that one takes precedence over the other. With this in mind, the following rules apply:

- A producer or consumer shall consider a defined name of the form used by cells in the range A1–XFD1048576 to be invalid.
- All other names outside this range can be defined names and shall override a cell reference if an ambiguity exists.

[*Example*: `LOG10` is always a cell reference, `LOG10(…)` is always a formula, and `LOGO1000` can be a defined name that overrides a cell reference. *end example*]

3.17.5.2 Precision

In order to clarify the semantics of cell formulas and values in SpreadsheetML, it is necessary to specify the precision of the numbers being represented in the file format. These numbers are therefore regarded as ranging over a specific value space, which defaults to the following:

The *value space* consists of the values m × 2^n, where m is an integer whose absolute value is less than 2^53, and n is an integer between -1075 and 970, inclusive. m is herein referred to as the *binary mantissa*, and n is herein referred to as the *binary exponent*.

[*Note*: The default precision is patterned after the IEEE double-precision 64-bit floating-point type [IEEE 754-1985]. *end note*]

Implementing applications can use the characteristics markup (§7.7) to specify other value spaces to replace the default in a given workbook. When present in the workbook, the value space defined using the characteristics markup overrides the default value space.

Regardless of the specific value space in use, values shall have a lexical representation as described in §3.17.5.3. Any numerical expression conforming to this lexical description is valid. However, numbers of higher precision than available in the value space, and numbers that lie outside the range representable in the value space shall be loaded as numbers in the value space or otherwise handled according to the prescription in §3.17.5.4.

3.17.5.3 Lexical Representation

The value space shall have a lexical representation consisting of a base 10 mantissa followed, optionally, by the character "E" or "e", followed by a base 10 exponent. The exponent shall be an integer. The mantissa shall be a decimal number. The representations for exponent and mantissa shall follow the lexical rules for integer and decimal below. If the "E" or "e" and the following exponent are omitted, an exponent value of 0 is assumed.

Lexical representations for zero can take a positive or negative sign.

[*Example*: -1E4, 1267.43233E12, 12.78e-2, 12 , -0, and 0 are all valid literals for numbers in the default value space. 4503599627370497.5 is also a valid literal, although it represents the same value as 4503599627370497 (2^52 + 1) in the default value space (as explained in §3.17.5.4). *end example*]

An *Integer* has a lexical representation consisting of a finite-length sequence of decimal digits (#x30–#x39) with an optional leading sign. If the sign is omitted, "+" is assumed. [*Example*: -1, 0, 12678967543233, +100000. *end example*]

A *Decimal Number* has a lexical representation consisting of a finite-length sequence of decimal digits (#x30–#x39) separated by a period as a decimal indicator. An optional leading sign is allowed. If the sign is omitted, "+" is assumed. Leading and trailing zeroes are optional. If the fractional part is zero, the period and following zero(s) can be omitted. [*Example*: -1.23, 12678967.543233, +100000.00, 210. *end example*]

3.17.5.4 Interpretation

Strings that are valid according to the lexical definition in §3.17.5.3 shall be interpreted as values in the value space as follows:

18. The mantissa shall be interpreted as a real number expressed in base 10
19. The exponent shall be interpreted as an integer expressed in base 10
20. The raw value for a numerical expression shall be interpreted as mantissa x $10^{exponent}$
21. If the raw value is larger than the largest value in the value space (2^1023 - 1, by default), or smaller than the smallest value in the value space (- 2^1023 + 1, by default), then a consuming application shall treat this as equivalent to the error value `#NUM!` (§3.17.3). Otherwise the value in the value space that is closest to the raw value is chosen as the interpretation. In the case that two values are equally close, the one with the smaller absolute value is chosen.

3.17.6 XML Representation

3.17.6.1 Cell Reference Style

A workbook saved with reference style A1 (§3.17.2.3.1), shall have the refMode attribute of the calcPr element (§3.2.2) in the Workbook part's XML omitted or set to A1. A workbook saved with reference style R1C1 (§3.17.2.3.2) shall have that refMode attribute set to R1C1. [*Example*: With R1C1 mode set, here is how the XML might look:

```
<workbook …>
  …
  <calcPr calcId="122211" fullCalcOnLoad="1" refMode="R1C1"/>
  …
</workbook>
```

end example]

Regardless of the value of the refMode attribute, cell references shall be stored in XML in the A1 form. [*Note*: This attribute's value tells an implementation which reference style to use at runtime. *end note*]

3.17.6.2 Scalar Formulas

A scalar formula shall be represented in a worksheet's XML by an f element that contains the text of the formula, and a v element that contains the text version of the last computed value for that formula. This pair of elements shall be inside a c element, which is, in turn, shall be inside a row element. [*Example*: Consider the scalar formula SQRT(C2^2+D2^2), where C2 refers to a cell containing the number 12.5, and D2 refers to a cell containing the number 9.6. The corresponding XML might be as follows:

```
<row r="2" spans="2:4">
  <c r="B2" s="40">
    <f>SQRT(C2^2+D2^2)</f>
    <v>15.761027885261798</v>
  </c>
  <c r="C2" s="0">
    <v>12.5</v>
  </c>
  <c r="D2" s="0">
    <v>9.6</v>
  </c>
</row>
```

In the scalar formula CONCATENATE("The total is ",C7," units"), C7 refers to a cell containing the number 23. The corresponding XML might be as follows:

```
<row r="7" spans="2:4" ht="285">
  <c r="B7" s="4" t="str">
    <f>CONCATENATE("The total is ",C7," units")</f>
    <v>The total is 23 units</v>
  </c>
  <c r="C7" s="0">
    <v>23</v>
  </c>
</row>
```

As the function CONCATENATE returns a string, the value for the cell's t attribute is `str`.

end example]

3.17.6.3 Array Formulas

An array-entered formula shall be represented in XML just like other formulas, except that the array-entered formula's f element shall contain an attribute t, whose value shall be `array`.

For a single-cell formula, the r attribute shall designate that cell. [*Example*: Consider the array formula `SUM(C11:C12*D11:D12)`. The corresponding XML might be as follows:

```
<row r="11" spans="2:4" ht="300">
  <c r="B11" s="16">
    <f t="array" r="B11">SUM(C11:C12*D11:D12)</f>
    <v>110</v>
  </c>
  <c r="C11" s="4">
    <v>10</v>
  </c>
  <c r="D11" s="0">
    <v>3</v>
  </c>
</row>
<row r="12" spans="2:4" ht="285">
  <c r="C12" s="4">
    <v>20</v>
  </c>
  <c r="D12" s="0">
    <v>4</v>
  </c>
</row>
```

As this formula is a single-cell formula, the r attribute contains the name of that cell, B11. *end example*]

For a multi-cell formula, the r attribute of the top-left cell of the range of cells to which that formula applies shall designate the range of cells to which that formula applies. The c elements for all cells except the top-left cell in that range shall not have an f element; however, they shall each have a v element. [*Example*: Consider the array formula A1:A3*B1:B3, which is applied to the cell range C1:C3. The corresponding XML might be as follows:

```
<row r="1" spans="1:3">
  <c r="A1" s="0">
    <v>112</v>
  </c>
  <c r="B1" s="0">
    <v>2.34</v>
  </c>
  <c r="C1" s="0">
    <f t="array" r="C1:C3">A1:A3*B1:B3</f>
    <v>262.08</v>
  </c>
</row>
<row r="2" spans="1:3">
  <c r="A2" s="0">
    <v>209</v>
  </c>
  <c r="B2" s="0">
    <v>1.28</v>
  </c>
  <c r="C2" s="0">
    <v>267.52</v>
  </c>
</row>
<row r="3" spans="1:3">
  <c r="A3" s="0">
    <v>128</v>
  </c>
  <c r="B3" s="0">
    <v>3.12</v>
  </c>
  <c r="C3" s="0">
    <v>399.36</v>
  </c>
</row>
```

As this formula is a multi-cell formula, the r attribute of cell C1 contains the name of that cell range, C1:C3, and cells C2 and C3 do not have an f element. *end example*]

3.17.6.4 Formula Evaluation Order

The order in which formulas are evaluated is determined by the order of their corresponding c elements in the calcChain element of the Calculation Chain part (§3.6.2).

3.17.6.5 Name Representation

A formula can contain one or more names. These names shall be defined in the Worksheet part's XML with each being the subject of a definedName element, inside a definedNames element. [*Example*: Consider the scalar formula `SUM(value1,value2)`. The corresponding XML might be as follows:

```
<definedNames>
  <definedName name="value1" localSheetId="0">Sheet2!$B$2</definedName>
  <definedName name="value2" localSheetId="0">Sheet2!$B$3</definedName>
</definedNames>
…
<c r="E5" s="0">
  <f ce="1">SUM(value1,value2)</f>
  <v>8</v>
</c>
```

end example]

Each name shall be the subject of an lpstr element in the Application-Defined File Properties part.

```
<TitlesOfParts>
  <vt:vector … baseType="lpstr">
    <vt:lpstr>Sheet1</vt:lpstr>
    <vt:lpstr>Sheet2</vt:lpstr>
    <vt:lpstr>Sheet3</vt:lpstr>
    <vt:lpstr>value1</vt:lpstr>
    <vt:lpstr>value2</vt:lpstr>
  </vt:vector>
</TitlesOfParts>
```

3.17.6.6 Value Representation

The most recent value of a formula shall be stored in the corresponding v element, as follows:

Result Type	Representation
array	The text form of the array's value.
error	The text form of the error value.
logical	The text `0` for FALSE and `1` for TRUE.
number	The unformatted text form of the number, as accurately as possible.
text	All of the characters in the text.

3.17.6.7 Dates and Times

As a date and/or time is represented by a number, a date/time serial value shall be stored in XML as the unformatted text form of that number, as accurately as possible.

The date base system is recorded in the Workbook part's XML by the presence or absence of the date1904 attribute of the workbookPr element. A value of 1 for this attribute indicates 1904. [*Example*:

```
1900: <workbookPr showObjects="all"/>
1904: <workbookPr date1904="1" showObjects="all"/>
```

end example]

3.17.7 Predefined Function Definitions

The **Syntax** entry for each function defined in this clause corresponds to a call to that function. The names in any **Syntax** entry typeset as in *number* and *string-1*, are parameter names for that function, and are local to that function definition's description.

When the type of an argument passed to a function is incompatible with the type expected the error value #VALUE! is returned by that function.

The set of predefined functions is divided into the following functional categories:

Category	Formulas
Cube	CUBEKPIMEMBER (§3.17.7.65), CUBEMEMBER (§3.17.7.65), CUBEMEMBERPROPERTY (§3.17.7.66), CUBERANKEDMEMBER (§3.17.7.67), CUBESET (§3.17.7.68), CUBESETCOUNT (§3.17.7.69), CUBEVALUE (§3.17.7.70)
Database	DAVERAGE (§3.17.7.77), DCOUNT (§3.17.7.80), DCOUNTA (§3.17.7.81), DGET (§3.17.7.90), DMAX (§3.17.7.91), DMIN (§3.17.7.92), DPRODUCT (§3.17.7.96), DSTDEV (§3.17.7.97), DSTDEVP (§3.17.7.98), DSUM (§3.17.7.99), DVAR (§3.17.7.101), and DVARP (§3.17.7.102).
Date and Time	DATE (§3.17.7.73), DATEDIF (§3.17.7.75), DATEVALUE (§3.17.7.75), DAY (§3.17.7.77), DAYS360 (§3.17.7.79), EDATE (§3.17.7.103), EOMONTH (§3.17.7.105), HOUR (§3.17.7.142), MINUTE (§3.17.7.211), MONTH (§3.17.7.217), NETWORKDAYS (§3.17.7.223), NOW (§3.17.7.231), SECOND (§3.17.7.284), TIME (§3.17.7.320), TIMEVALUE (§3.17.7.321), TODAY (§3.17.7.322), WEEKDAY (§3.17.7.340), WEEKNUM (§3.17.7.342), WORKDAY (§3.17.7.343), YEAR (§3.17.7.346), and YEARFRAC (§3.17.7.348)
Engineering	BESSELI (§3.17.7.23), BESSELJ (§3.17.7.24), BESSELK (§3.17.7.25), BESSELY (§3.17.7.26), BIN2DEC (§3.17.7.29), BIN2HEX (§3.17.7.30), BIN2OCT (§3.17.7.31), COMPLEX (§3.17.7.45), CONVERT (§3.17.7.48), DEC2BIN (§3.17.7.83), DEC2HEX (§3.17.7.85), DEC2OCT (§3.17.7.86), DELTA (§3.17.7.88), ERF (§3.17.7.107), ERFC (§3.17.7.108), GESTEP (§3.17.7.135), HEX2BIN (§3.17.7.138), HEX2DEC (§3.17.7.140), HEX2OCT (§3.17.7.141), IMABS (§3.17.7.148), IMAGINARY (§3.17.7.149), IMARGUMENT (§3.17.7.150), IMCONJUGATE (§3.17.7.151), IMCOS (§3.17.7.152), IMDIV (§3.17.7.153), IMEXP (§3.17.7.154), IMLN (§3.17.7.155), IMLOG10 (§3.17.7.156), IMLOG2 (§3.17.7.157), IMPOWER (§3.17.7.158), IMPRODUCT (§3.17.7.159), IMREAL

Category	Formulas
	(§3.17.7.160), IMSIN (§3.17.7.161), IMSQRT (§3.17.7.162), IMSUB (§3.17.7.163), IMSUM (§3.17.7.164), OCT2BIN (§3.17.7.233), OCT2DEC (§3.17.7.235), and OCT2HEX (§3.17.7.236).
Financial	ACCRINT (§3.17.7.2), ACCRINTM (§3.17.7.2), AMORDEGRC (§3.17.7.7), AMORLINC (§3.17.7.7), COUPDAYBS (§3.17.7.58), COUPDAYS (§3.17.7.57), COUPDAYSNC (§3.17.7.59), COUPNCD (§3.17.7.59), COUPNUM (§3.17.7.60), COUPPCD (§3.17.7.61), CUMIPMT (§3.17.7.71), CUMPRINC (§3.17.7.72), DB (§3.17.7.80), DDB (§3.17.7.82), DISC (§3.17.7.90), DOLLARDE (§3.17.7.95), DOLLARFR (§3.17.7.95), DURATION (§3.17.7.100), EFFECT (§3.17.7.105), FV (§3.17.7.128), FVSCHEDULE (§3.17.7.128), INTRATE (§3.17.7.170), IPMT (§3.17.7.170), IRR (§3.17.7.171), ISPMT (§3.17.7.182), MDURATION (§3.17.7.206), MIRR (§3.17.7.214), NOMINAL (§3.17.7.225), NPER (§3.17.7.232), NPV (§3.17.7.232), ODDFPRICE (§3.17.7.206), ODDFYIELD (§3.17.7.238), ODDLPRICE (§3.17.7.239), ODDLYIELD (§3.17.7.240), PMT (§3.17.7.250), PPMT (§3.17.7.253), PRICE (§3.17.7.253), PRICEDISC (§3.17.7.254), PRICEMAT (§3.17.7.255), PV (§3.17.7.260), RATE (§3.17.7.266), RECEIVED (§3.17.7.267), SLN (§3.17.7.289), SYD (§3.17.7.311), TBILLEQ (§3.17.7.315), TBILLPRICE (§3.17.7.315), TBILLYIELD (§3.17.7.316), VDB (§3.17.7.338), XIRR (§3.17.7.345), XNPV (§3.17.7.345), YIELD (§3.17.7.349), YIELDDISC (§3.17.7.349), and YIELDMAT (§3.17.7.350).
Information	CELL (§3.17.7.34), ERROR.TYPE (§3.17.7.109), INFO (§3.17.7.167), ISBLANK (§3.17.7.172), ISERR (§3.17.7.174), ISERROR (§3.17.7.175), ISEVEN (§3.17.7.176), ISLOGICAL (§3.17.7.177), ISNA (§3.17.7.178), ISNONTEXT (§3.17.7.179), ISNUMBER (§3.17.7.180), ISODD (§3.17.7.181), ISREF (§3.17.7.182), ISTEXT (§3.17.7.184), N (§3.17.7.221), NA (§3.17.7.222), and TYPE (§3.17.7.330).
Logical	AND (§3.17.7.8), FALSE (§3.17.7.116), IF (§3.17.7.146), IFERROR (§3.17.7.147), NOT (§3.17.7.229), OR (§3.17.7.242), and TRUE (§3.17.7.327).
Lookup and Reference	ADDRESS (§3.17.7.6), AREAS (§3.17.7.10), CHOOSE (§3.17.7.39), COLUMN (§3.17.7.42), COLUMNS (§3.17.7.43, GETPIVOTDATA (§3.17.7.136), HLOOKUP (§3.17.7.142), HYPERLINK (§3.17.7.144), INDEX (§3.17.7.165), INDIRECT (§3.17.7.166), LOOKUP (§3.17.7.200), MATCH (§3.17.7.202), OFFSET (§3.17.7.241), ROW (§3.17.7.278), ROWS (§3.17.7.279), RTD (§3.17.7.280), TRANSPOSE (§3.17.7.324), and VLOOKUP (§3.17.7.339).
Math and Trig	ABS (§3.17.7.1), ACOS (§3.17.7.3), ACOSH (§3.17.7.5), ASIN (§3.17.7.12), ASINH (§3.17.7.13), ATAN (§3.17.7.14), ATAN2 (§3.17.7.15), ATANH (§3.17.7.16), CEILING (§3.17.7.33), COMBIN (§3.17.7.44), COS (§3.17.7.50), COSH (§3.17.7.51), DEGREES (§3.17.7.87), EVEN (§3.17.7.110), EXP (§3.17.7.112), FACT (§3.17.7.114), FACTDOUBLE (§3.17.7.115), FLOOR (§3.17.7.124), GCD (§3.17.7.133), INT (§3.17.7.168), LCM (§3.17.7.188), LN (§3.17.7.194), LOG (§3.17.7.195), LOG10 (§3.17.7.196), MDETERM (§3.17.7.205), MINVERSE (§3.17.7.213), MMULT (§3.17.7.214), MOD (§3.17.7.216), MROUND (§3.17.7.219), MULTINOMIAL (§3.17.7.220), ODD (§3.17.7.237), PI (§3.17.7.249), POWER (§3.17.7.251), PRODUCT

Category	Formulas
	(§3.17.7.257), QUOTIENT (§3.17.7.261), RADIANS (§3.17.7.263), RAND (§3.17.7.264), RANDBETWEEN (§3.17.7.265), ROMAN (§3.17.7.274), ROUND (§3.17.7.275), ROUNDDOWN (§3.17.7.276), ROUNDUP (§3.17.7.277), SERIESSUM (§3.17.7.285), SIGN (§3.17.7.286), SIN (§3.17.7.287), SINH (§3.17.7.288), SQRT (§3.17.7.293), SQRTPI (§3.17.7.294), SUBTOTAL (§3.17.7.302), SUM (§3.17.7.303), SUMIF (§3.17.7.304), SUMIFS (§3.17.7.305), SUMPRODUCT (§3.17.7.306), SUMSQ (§3.17.7.307), SUMX2MY2 (§3.17.7.308), SUMX2PY2 (§3.17.7.309), SUMXMY2 (§3.17.7.310), TAN (§3.17.7.313), TANH (§3.17.7.314), and TRUNC (§3.17.7.329).
Statistical	AVEDEV (§3.17.7.17), AVERAGE (§3.17.7.18), AVERAGEA (§3.17.7.19),AVERAGEIF (§3.17.7.20), AVERAGEIFS (§3.17.7.21), BETADIST (§3.17.7.27), BETAINV (§3.17.7.28), BINOMDIST (§3.17.7.32), CHIDIST (§3.17.7.36), CHIINV (§3.17.7.37), CHITEST (§3.17.7.38), CONFIDENCE (§3.17.7.47), CORREL (§3.17.7.49), COUNT (§3.17.7.52), COUNTA (§3.17.7.53), COUNTBLANK (§3.17.7.54), COUNTIF (§3.17.7.55),COUNTIFS (§3.17.7.56), COVAR (§3.17.7.62), CRITBINOM (§3.17.7.64), DEVSQ (§3.17.7.89), EXPONDIST (§3.17.7.113), FDIST (§3.17.7.117), FINV (§3.17.7.120), FISHER (§3.17.7.121), FISHERINV (§3.17.7.122), FORECAST (§3.17.7.125), FREQUENCY (§3.17.7.126), FTEST (§3.17.7.127), GAMMADIST (§3.17.7.129), GAMMAINV (§3.17.7.131), GAMMALN (§3.17.7.132), GEOMEAN (§3.17.7.134), GROWTH (§3.17.7.136), HARMEAN (§3.17.7.138), HYPGEOMDIST (§3.17.7.144), INTERCEPT (§3.17.7.169), KURT (§3.17.7.186), LARGE (§3.17.7.187), LINEST (§3.17.7.193), LOGEST (§3.17.7.197), LOGINV (§3.17.7.198), LOGNORMDIST (§3.17.7.199), MAX (§3.17.7.202), MAXA (§3.17.7.204), MEDIAN (§3.17.7.206), MIN (§3.17.7.210), MINA (§3.17.7.210), MODE (§3.17.7.217), NEGBINOMDIST (§3.17.7.223), NORMDIST (§3.17.7.225), NORMINV (§3.17.7.226), NORMSDIST (§3.17.7.227), NORMSINV (§3.17.7.228), PEARSON (§3.17.7.244), PERCENTILE (§3.17.7.244), PERCENTRANK (§3.17.7.245), PERMUT (§3.17.7.246), POISSON (§3.17.7.250), PROB (§3.17.7.256), QUARTILE (§3.17.7.260), RANK (§3.17.7.266), RSQ (§3.17.7.280), SKEW (§3.17.7.280), SLOPE (§3.17.7.290), SMALL (§3.17.7.291), STANDARDIZE (§3.17.7.295), STDEV (§3.17.7.295), STDEVA (§3.17.7.296), STDEVP (§3.17.7.297), STDEVPA (§3.17.7.298), STEYX (§3.17.7.299), TDIST (§3.17.7.317), TINV (§3.17.7.322), TREND (§3.17.7.324), TRIMMEAN (§3.17.7.327), TTEST (§3.17.7.330), VAR (§3.17.7.335), VARA (§3.17.7.335), VARP (§3.17.7.336), VARPA (§3.17.7.337), WEIBULL (§3.17.7.343), and ZTEST (§3.17.7.351).
Text and Data	ASC (§3.17.7.11), BAHTTEXT (§3.17.7.22), CHAR (§3.17.7.35), CLEAN (§3.17.7.40), CODE (§3.17.7.41), CONCATENATE (§3.17.7.46), DOLLAR (§3.17.7.93), EXACT (§3.17.7.111), FIND (§3.17.7.118), FINDB (§3.17.7.119), FIXED (§3.17.7.123), JIS (§3.17.7.185), LEFT (§3.17.7.189), LEFTB (§3.17.7.190), LEN (§3.17.7.191), LENB (§3.17.7.192), LOWER (§3.17.7.200), MID (§3.17.7.207), MIDB (§3.17.7.209), PHONETIC (§3.17.7.248), PROPER (§3.17.7.259), REPLACE (§3.17.7.269), REPLACEB (§3.17.7.270), REPT

Category	Formulas
	(§3.17.7.271), RIGHT (§3.17.7.272), RIGHTB (§3.17.7.273), SEARCH (§3.17.7.282), SEARCHB (§3.17.7.283), SUBSTITUTE (§3.17.7.300), T (§3.17.7.311), TEXT (§3.17.7.318), TRIM (§3.17.7.325), UPPER (§3.17.7.332), and VALUE (§3.17.7.334).

3.17.7.1 ABS

Syntax:

ABS (*x*)

Description: Computes the absolute value of *x*.

Arguments:

Name	Type	Description
x	number	The value whose absolute value is to be determined.

Return Type and Value: number – The absolute value of *x*.

[*Example*:

ABS(10.5) results in 10.5
ABS(0) results in 0
ABS(-10.5) results in 10.5

end example]

3.17.7.2 ACCRINT

Syntax:

ACCRINT (*issue* , *first-interest* , *settlement* , *rate* , [*par*] , *frequency* [, [*basis*]])

Description: Computes the accrued interest for a security that pays periodic interest.

Mathematical Formula:

$$ACCRINT = par \times \frac{rate}{frequency} \times \sum_{i=1}^{NC} \frac{A_i}{NL_i}$$

where:

Ai = number of accrued days for the i^{th} quasi-coupon period within odd period.
NC = number of quasi-coupon periods that fit in odd period. If this number contains a fraction, raise it to the next whole number.
NLi = normal length in days of the i^{th} quasi-coupon period within odd period.

Arguments:

Name	Type	Description
issue	number	The security's issue date.
first-interest	number	The security's first interest date.
settlement	number	The security's settlement date.
rate	number	The security's annual coupon rate.
par	number	The security's par value. If omitted, 1,000 is used.
frequency	number	The number of coupon payments per year. For annual payments, frequency is 1; for semiannual payments, frequency is 2; for quarterly payments, frequency is 4. *frequency* is truncated to an integer.
basis	number	The truncated integer type of day count basis to use, as follows: `0` or omitted: US (NASD) 30/360 1: Actual/actual 2: Actual/360 3: Actual/365 4: European 30/360

Time information in the date arguments is ignored.

Return Type and Value: number – The accrued interest for a security that pays periodic interest.

However, if

- *issue*, *first-interest*, or *settlement* is out of range for the current date base value, `#NUM!` is returned
- *issue* ≥ *settlement*, `#NUM!` is returned
- *rate* or *par* ≤ 0, `#NUM!` is returned
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned
- *basis* < 0 or *basis* > 4, `#NUM!` is returned

[*Example*:

`ACCRINT(DATE(2006,3,1),DATE(2006,9,1),DATE(2006,5,1),0.1,1100,2,0)` results in `18.33`
`ACCRINT(DATE(2006,3,1),DATE(2006,9,1),DATE(2006,5,1),0.1,,2,0)` results in `16.67`

end example]

3.17.7.3 ACCRINTM

Syntax:

ACCRINTM (*issue* , *settlement* , *rate* , [[*par*] [, [*basis*]]])

Description: Computes the accrued interest for a security that pays interest at maturity.

Mathematical Formula:

$$ACCRINTM = par \times rate \times \frac{A}{D}$$

where:

A = Number of accrued days counted according to a monthly basis. For interest at maturity items, the number of days from the issue date to the maturity date is used.
D = Annual Year Basis.

Arguments:

Name	Type	Description
issue	number	The security's issue date.
settlement	number	The security's settlement date.
rate	number	The security's annual coupon rate.
par	number	The security's par value. If omitted, 1,000 is used.
basis	number	The truncated integer type of day count basis to use, as follows:

Value	Day Count Basis
0 or omitted	US (NASD) 30/360
1	Actual/actual
2	Actual/360
3	Actual/365
4	European 30/360

Time information in the date arguments is ignored.

Return Type and Value: number – The accrued interest for a security that pays interest at maturity.

However, if

- *issue* or *settlement* is out of range for the current date base value, #NUM! is returned
- *issue* ≥ *settlement*, #NUM! is returned

- *rate* or *par* ≤ 0, #NUM! is returned
- *basis* < 0 or *basis* > 4, #NUM! is returned

[*Example*:

ACCRINTM(DATE(2006,3,1),DATE(2006,5,1),0.1,1100,0) results in 18.33333333
ACCRINTM(DATE(2006,3,1),DATE(2006,5,1),0.1,,0) results in 16.66666667
ACCRINTM(DATE(2006,3,1),DATE(2006,5,1),0.1,) results in 16.66666667

end example]

3.17.7.4 ACOS

Syntax:

ACOS (*x*)

Description: Computes the arc cosine of *x*.

Arguments:

Name	Type	Description
x	number	The value whose arc cosine is to be determined.

Return Type and Value: number – The arc cosine of *x*.

However, if *x* is outside the interval [-1,+1], #NUM! is returned

[*Example*:

ACOS(-1) results in 3.141592654
ACOS(0) results in 1.570796327
ACOS(1) results in 0

end example]

3.17.7.5 ACOSH

Syntax:

ACOSH (*x*)

Description: Computes the inverse hyperbolic cosine of *x*.

Arguments:

Name	Type	Description

Name	Type	Description
x	number	The value whose inverse hyperbolic cosine is to be determined.

Return Type and Value: number – The inverse hyperbolic cosine of *x*.

However, if *x* < 1, `#NUM!` is returned.

[*Example*:

`ACOSH(1)` results in `0`
`ACOSH(10)` results in `2.993222846`
`ACOSH(100)` results in `5.298292366`

end example]

3.17.7.6 ADDRESS

Syntax:

```
ADDRESS ( row-number , col-number [ , [ ref-type ] [ , [ A1-ref-style-flag ]
    [ , sheet-name ] ] ] )
```

Description: Creates a cell address, given the specified row and column numbers.

Arguments:

Name	Type	Description
row-number	number	The number of the row.
col-number	number	The number of the column.
ref-type	number	The type of reference to return, as follows: 1 or omitted – Absolute row and column 2 – Absolute row; relative column 3 – Relative row; absolute column 4 – Relative row and column
A1-ref-style-flag	logical	The style of the reference. If TRUE or omitted, an A1-style reference (§3.17.2.3.1) is returned; otherwise, an R1C1-style reference (§3.17.2.3.2) is returned.
sheet-name	text	The name of the worksheet to be used. If omitted, no

Name	Type	Description
		sheet name is used.

Return Type and Value: text – A cell address, given the specified row and column numbers.

However, if

- *row-number* or *col-number* < 1, `#NUM!` is returned.
- *ref-type* is outside the range 1–4, `#NUM!` is returned.

[*Example*:

In A1-reference style mode:

`ADDRESS(5,7,1)` results in `$G$5`
`ADDRESS(5,7,2)` results in `G$5`
`ADDRESS(5,7,3)` results in `$G5`
`ADDRESS(5,7,4)` results in `G5`
`ADDRESS(5,7,,,"Sheet1")` results in `Sheet1!$G$5`

In R1C1-reference style mode:

`ADDRESS(5,7,1,FALSE)` results in `R5C7`
`ADDRESS(5,7,2,FALSE)` results in `R5C[7]`
`ADDRESS(5,7,3,FALSE)` results in `R[5]C7`
`ADDRESS(5,7,4,FALSE)` results in `R[5]C[7]`

end example]

3.17.7.7 AMORDEGRC

Syntax:

AMORDEGRC (*cost* , *date-purchased* , *first-period* , *salvage* , *period* , *rate* [, [*basis*]])

Description: Computes the depreciation for each accounting period. (This function is provided for the French accounting system. If an asset is purchased in the middle of the accounting period, the prorated depreciation is taken into account. The function is similar to `AMORLINC` (§3.17.7.7), except that a depreciation coefficient is applied in the calculation depending on the life of the assets.)

Arguments:

Name	Type	Description
cost	number	The cost of the asset.

Name	Type	Description
date-purchased	number	The date of the purchase of the asset.
first-period	number	The date of the end of the first period.
salvage	number	The salvage value at the end of the life of the asset.
period	number	The period.
rate	number	The rate of depreciation.
basis	number	The truncated integer type of day count basis to use, as follows: `0` or omitted – US (NASD) 30/360 1 – Actual/actual 2 – Actual/360 3 – Actual/365 4 – European 30/360

Return Type and Value: number – The depreciation for each accounting period.

However, if

- *cost*, *salvage*, *period*, *or rate* < 0, `#NUM!` is returned.
- *date-purchased* or *first-period* is out of range for the current date base value, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.
- The life of the assets is between 0 and 1, 1 and 2, 2 and 3, or 4 and 5, `#NUM!` is returned.

This function returns the depreciation until the last period of the life of the assets or until the cumulated value of depreciation is greater than the cost of the assets minus the salvage value.

The depreciation coefficients are:

Life of assets (1/rate)	Depreciation Coefficient
Between 3 and 4 years	1.5
Between 5 and 6 years	2
More than 6 years	2.5

The depreciation rate grows to 50 percent for the period preceding the last period, and grows to 100 percent for the last period.

[*Example*:

`AMORDEGRC(2400,DATE(2008,8,19),DATE(2008,12,31),300,1,0.15,1)` results in `776.00`

end example]

3.17.7.8 AMORLINC

Syntax:

```
AMORLINC ( cost , date-purchased , first-period , salvage , period ,
    rate [ , [ basis ] ] )
```

Description: Computes the depreciation for each accounting period. (This function is provided for the French accounting system. If an asset is purchased in the middle of the accounting period, the prorated depreciation is taken into account.)

Arguments:

Name	Type	Description
cost	number	The cost of the asset.
date-purchased	number	The date of the purchase of the asset.
first-period	number	The date of the end of the first period.
salvage	number	The salvage value at the end of the life of the asset.
period	number	The period.
rate	number	The rate of depreciation.
basis	number	The truncated integer type of day count basis to use, as follows: **Value** / **Day Count Basis** `0` or omitted / US (NASD) 30/360 1 / Actual/actual 2 / Actual/360 3 / Actual/365 4 / European 30/360

Return Type and Value: number – The depreciation for each accounting period.

However, if:

- *cost*, *salvage*, *period*, or *rate* < 0, `#NUM!` is returned.

- *date-purchased* or *first-period* is out of range for the current date base value, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`AMORLINC(2400,DATE(2008,8,19),DATE(2008,12,31),300,1,0.15,1)` results in `360.00`

end example]

3.17.7.9 AND

Syntax:

AND (*argument-list*)

Description: Tests if all *argument*s in *argument-list* are TRUE.

Arguments:

Name	Type	Description
argument-list	logical, array, or cell reference	The *argument*s in *argument-list* designate the values to be tested. For an array or cell reference, a cell that contains text or is empty shall be ignored.

Return Type and Value: logical – TRUE if all *argument*s in *argument-list* are TRUE; otherwise, FALSE.

However, if no logical values are found, the return value is unspecified.

[*Example*:

`AND(TRUE)` results in TRUE
`AND(TRUE,FALSE)` results in FALSE
`AND(10>5,3=1+2,5)` results in TRUE
`AND({10,5,6,7},TRUE,E6:F6)` results in TRUE, when E6 contains TRUE and F6 contains 10

end example]

3.17.7.10 AREAS

Syntax:

AREAS (*reference*)

Description: Finds the number of areas (§3.17.2.3) designated by *reference*.

Arguments:

Name	Type	Description
reference	reference	A reference to a single cell or to a range of cells that can refer to multiple areas.

Return Type and Value: number – The number of areas designated by *reference*.

However, if the reference designates no areas, `#NUM!` is returned.

[*Example*:

`AREAS(E312)` results in `1`
`AREAS(E311:F313)` results in `1`
`AREAS((E312:F314,G316:H316,G311))` results in 3, given the union of the three areas
`AREAS((E312:F314 E313:F314 F312:F314))` results in 1, given the intersection of the three areas

end example]

3.17.7.11 ASC

Syntax:

ASC (*string*)

Description: For double-byte character set (DBCS) languages, converts all full-width (double-byte) characters to half-width (single-byte) characters.

Arguments:

Name	Type	Description
string	text	Designate the text to be converted. If *string* does not contain any full-width characters, nothing in *string* is converted.

Return Type and Value: text – The text resulting from the conversion.

[*Example*:

`ASC("ABC")` results in ABC
`ASC("エクセル")` results in ｴｸｾﾙ

end example]

3.17.7.12 ASIN

Syntax:

ASIN (*x*)

Description: Computes the arc sine of *x*.

Arguments:

Name	Type	Description
x	number	The value whose arc sine is to be determined.

Return Type and Value: number – The arc sine of *x*.

However, if *x* is outside the interval [-1,+1], #NUM! is returned.

[*Example*:

ASIN(-1) results in -1.570796327
ASIN(0) results in 0
ASIN(1) results in 1.570796327

end example]

3.17.7.13 ASINH

Syntax:

ASINH (*x*)

Description: Computes the inverse hyperbolic cosine of *x*.

Arguments:

Name	Type	Description
x	number	The value whose inverse hyperbolic sine is to be determined.

Return Type and Value: number – The inverse hyperbolic cosine of *x*.

[*Example*:

ASINH(1) results in 0.881373587
ASINH(10) results in 2.99822295
ASINH(100) results in 5.298342366
ASINH(0.5) results in 0.481211825

end example]

3.17.7.14 ATAN

Syntax:

```
ATAN ( x )
```

Description: Computes the arc tangent of *x*.

Arguments:

Name	Type	Description
x	number	The value whose arc tangent is to be determined.

Return Type and Value: number – The arc tangent of *x*.

[*Example*:

`ATAN(-1)` results in `-0.785398163`
`ATAN(0)` results in `0`
`ATAN(1)` results in `0.785398163`
`ATAN(-10)` results in `1.471127674`
`ATAN(10)` results in `1.471127674`

end example]

3.17.7.15 ATAN2

Syntax:

```
ATAN2 ( x , y )
```

Description: Computes the arc tangent of the coordinates *x* and *y*.

Arguments:

Name	Type	Description
x	number	The first coordinate.
y	number	The second coordinate.

Return Type and Value: number – The arc tangent of *x*.

However, if both *x* and *y* are zero, `#DIV/0!` is returned.

[*Example*:

`ATAN2(1,1)` results in `0.785398163`

ATAN2(-2,2) results in 2.35619449
ATAN2(3,-3) results in -0.785398163

end example]

3.17.7.16 ATANH

Syntax:

ATANH (*x*)

Description: Computes the inverse hyperbolic tangent of *x*.

Arguments:

Name	Type	Description
x	number	The value whose inverse hyperbolic tangent is to be determined.

Return Type and Value: number – The inverse hyperbolic tangent of *x*.

However, if *x* is outside the interval [-1,+1], #NUM! is returned.

[*Example*:

ATANH(-0.999999) results in -7.254328619
ATANH(0) results in 0
ATANH(0.999999) results in 7.254328619

end example]

3.17.7.17 AVEDEV

Syntax:

AVEDEV (*argument-list*)

Description: Computes the average of the absolute deviations of a set of data points from their mean. AVEDEV is a measure of the variability in a data set.

Mathematical Formula:

The number of combinations is as follows, where *number* = *n* and *number-chosen* = *k*:

$$\binom{n}{k} = \frac{P_{k,n}}{k!} = \frac{n!}{k!(n-k)!}$$

Arguments:

Name	Type	Description
argument-list	logical, number, name, array, or reference that contains a number. The list can be a single argument that is an array or a reference to an array.	The *argument*s in *argument-list* designate the values for which the average of the absolute deviations is to be computed. Logical values and text representations of numbers occurring directly in the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0value 0 are included.

Return Type and Value: number – The average of the absolute deviations of a set of data points from their mean.

[*Example*:

AVEDEV(-3.5,1.4,6.9,-4.5) results in 4.075
AVEDEV({-3.5,1.4,6.9,-4.5}) results in 4.075

end example]

3.17.7.18 AVERAGE

Syntax:

AVERAGE (*argument-list*)

Description: Computes the arithmetic mean of the numeric values of its arguments.

Arguments:

Name	Type	Description
argument-list	logical, number, name, text, or reference that contains a number.	The *argument*s in *argument-list* designate the values to be averaged. An argument that is a logical value or the text representation of a number shall be counted. If an array or cell reference argument contains logical values, text, or empty cells, those values shall be ignored; however, cells having the value 0value 0 shall be counted. [*Note*: The function AVERAGEA (§3.17.7.18) does include cell reference arguments that refer to logical values or text representations of numbers. *end note*]

Return Type and Value: number – The arithmetic mean of the values of its arguments.

[*Example*:

AVERAGE(1,2,3,4,5) results in 3
AVERAGE({1,2;3,4}) results in 2.5
AVERAGE({1,2,3,4,5},6,"7") results in 4
AVERAGE({1,"2",TRUE,4}) results in 2.5, as the logical value and numeric text are ignored

end example]

3.17.7.19 AVERAGEA

Syntax:

AVERAGEA (*argument-list*)

Description: Computes the arithmetic mean of the values of its arguments.

Arguments:

Name	Type	Description
argument-list	logical, number, text, or reference that contains a number.	The *argument*s in *argument-list* designate the values to be averaged. An argument that is a logical value or the text representation of a number shall be counted. Arguments with value TRUE evaluate to 1; arguments with value FALSE evaluate to 0. An array or cell reference argument that contains text evaluates to 0. If an argument is an array or reference, only values in that array or reference are used. Empty cells and text values in the array or reference are ignored.

[*Note*: The function AVERAGE (§3.17.7.18) does not include cell reference arguments that refer to logical values or text representations of numbers. *end note*]

Return Type and Value: number – The arithmetic mean of the values of its arguments.

[*Example*:

AVERAGEA(10,E1), where E1 is an empty cell, results in 10, as E1 is ignored
AVERAGEA(10,E2), where E2 contains TRUE, results in 5.5
AVERAGEA(10,E3), where E3 contains FALSE, results in 5

end example]

3.17.7.20 AVERAGEIF

Syntax:

AVERAGEIF (*cell-range* , *selection-criteria* [, *average-range*])

Description: Applies selection criteria on the values in one range of cells and averages the values of the cells in a corresponding range.

Arguments:

Name	Type	Description
cell-range	reference	The range of cells to be inspected. Cells in *cell-range* that contain TRUE or FALSE are ignored. If a cell is an empty cell, it is ignored.
selection-criteria	number, expression, reference, text	Designates the cells that are to be averaged. In the case of text, *selection-criteria* can consist of any comparison operator followed by the operand against which each cell's value is to be compared. *selection-criteria* can include one or more wildcard characters, question mark (?) and asterisk (*). A question mark matches any single character; an asterisk matches any sequence of characters. To search for a question mark, asterisk, or tilde character, prefix that character with a tilde (~). If a cell in *selection-criteria* is empty, it is treated as if it contained 0.
average-range	reference	Designates the cells whose values are averaged. In this case, *average-range* need not have the same size and shape as *cell-range*. The actual cells that are averaged are determined by using the top, left cell in *average-range* as the beginning cell, and then including cells that correspond in size and shape to *cell-range*. If *average-range* is omitted, *cell-range* also designates the cells whose values are averaged. If a cell is an empty cell, it is ignored.

Return Type and Value: number – The average of the values of the cells corresponding to those selected.

However, if no cells in the range meet the criteria, the return value is unspecified.

[*Example*: Assuming A2:A4 contains 10, 20, and 30:

AVERAGEIF(A2:A4,“>15”) results in 25, the average of 20 and 30.

end example]

3.17.7.21 AVERAGEIFS

Syntax:

AVERAGEIFS (*average-range* , *cell-range-1* , *selection-criteria-1*
[, *cell-range-2* , *selection-criteria-2* [, …]])

Description: The average of the values of all cells that meet multiple criteria.

Arguments:

Name	Type	Description
average-range	reference	Designates the cells whose values are averaged. In this case, *average-range* need not have the same size and shape as *cell-range-1* through *cell-range-n*. The actual cells that are added are determined by using the top, left cell in *average-range* as the beginning cell, and then including cells that correspond in size and shape to *cell-range-1* through *cell-range-n*. If a cell in *average-range* is empty, that cell is ignored. Each cell in *average-range* is used in the average calculation only if all of the corresponding criteria specified are true for that cell.
cell-range-1	number, expression, reference, text	Designates the first range of cells to be inspected.
selection-criteria-1	reference, text	*selection-criteria-1* specifies the criteria for the first range of cells that will be averaged. In the case of text, *selection-criteria-1* can consist of any comparison operator followed by the operand against which each cell's value is to be compared. If a cell in any selection criteria range is empty, it is treated as if its value was 0. Cells that contain TRUE evaluate to 1; cells in any range that contain FALSE evaluate to 0. *selection-criteria-1* can include one or more wildcard characters, question mark (?) and asterisk (*). A question mark matches any single character; an asterisk matches any sequence of characters. To search for a question mark, asterisk, or tilde character, prefix that character with a tilde (~).
cell-range-n	number, expression, reference, text	The optional arguments *selection-criteria-2* through *selection-criteria-n* have corresponding arguments *cell-range-2* through *cell-range-n*, and have the same semantics as *selection-criteria-1* and *cell-range-1*, respectively.
selection-criteria-n	reference, text	

Return Type and Value: number – The average of the cells corresponding to those selected.

However, if

- Cells in *average-range* are empty or contain text values that cannot be translated into numbers, the return value is unspecified.
- There are no cells that meet all the criteria, the return value is unspecified.

[*Example*: Given the following data:

	A	B	C	D
1	Student	First Quiz Grade	Second Quiz Grade	Final Exam Grade
2	Emilio	75	85	87
3	Julie	94	80	88
4	Hans	86	93	Incomplete
5	Frederique	Incomplete	75	75

`AVERAGEIFS(B2:B5,B2:B5,">70",B2:B5,"<90")` results in `80.5` (the average for all students all first quiz grades that are between 70 and 90)

`AVERAGEIFS(D2:D5,D2:D5,"<>Incomplete",D2:D5,">80")` results in `87.5` (the average for all students all first quiz grades that are above 80 and not marked "Incomplete")

`AVERAGEIFS(B2:D5,B2:B5,"<>Incomplete",C2:C5,"<>Incomplete",D2:D5, "<>Incomplete")` results in 82.375 (the average grades for all students who do not have incomplete grades)

end example]

3.17.7.22 BAHTTEXT

Syntax:

`BAHTTEXT` (*number*)

Description: Produces a string containing *number* formatted according to the Thai convention.

Arguments:

Name	Type	Description
number	number	The value to be formatted.

Return Type and Value: text – The text containing *number* formatted.

[*Example*:

`BAHTTEXT(1234)` results in หนึ่งพันสองร้อยสามสิบสี่บาทถ้วน

end example]

3.17.7.23 BESSELI

Syntax:

```
BESSELI ( x , n )
```

Description: The modified Bessel function In(x), which is equivalent to the Bessel function Jn(x) evaluated for purely imaginary arguments.

Mathematical Formula:

The n-th order modified Bessel function of the variable x is:

$$I_n(x) = (i)^{-n} J_n(ix)$$

Arguments:

Name	Type	Description
x	number	The value at which to evaluate the function.
n	number	The order of the Bessel function. This value is truncated to an integer.

Return Type and Value: number – The Bessel function In(x).

However, if *n* < 0, `#NUM!` is returned.

[*Example*:

`BESSELI(-5.6,0)` results in `46.73755194`
`BESSELI(2.345,5)` results in `0.023137792`

end example]

3.17.7.24 BESSELJ

Syntax:

```
BESSELJ ( x , n )
```

Description: The Bessel function Jn(x).

Mathematical Formula:

The n-th order Bessel function of the variable x is:

$$J_n(x) = \sum_{k=0}^{\infty} \frac{(-1)^k}{k!\Gamma(n+k+1)} \left(\frac{x}{2}\right)^{n+2k}$$

where:

$$\Gamma(n+k+1) = \int_0^{\infty} e^{-x} x^{n+k} dx$$

is the Gamma function.

Arguments:

Name	Type	Description
x	number	The value at which to evaluate the function.
n	number	The order of the Bessel function. This value is truncated to an integer.

Return Type and Value: number – The Bessel function Jn(x).

However, if *n* < 0, `#NUM!` is returned.

[*Example*:

`BESSELJ(-5.6,0)` results in `0.026970887`
`BESSELJ(2.345,5)` results in `0.014627862`

end example]

3.17.7.25 BESSELK

Syntax:

```
BESSELK ( x , n )
```

Description: The modified Bessel function Kn(x), which is equivalent to using the Bessel function Jn(x) and Yn(x).

Mathematical Formula:

The n-th order modified Bessel function of the variable x is:

$$K_n(x) = \frac{p}{2} i^{n+1} [J_n(ix) + iY_n(ix)]$$

where Jn and Yn are the J and Y Bessel functions, respectively.

Arguments:

Name	Type	Description
x	number	The value at which to evaluate the function.
n	number	The order of the Bessel function. This value is truncated to an integer.

Return Type and Value: number – The Bessel function Kn(x).

However, if *n* < 0, `#NUM!` is returned.

[*Example*:

`BESSELK(2.345,5)` results in `3.904137225`

end example]

3.17.7.26 BESSELY

Syntax:

```
BESSELY ( x , n )
```

Description: Weber's Bessel function Yn(x).

Mathematical Formula:

The n-th order Bessel function of the variable x is:

$$Y_n(x) = \lim_{\nu \to n} \frac{J_\nu(x)\cos(\nu\pi) - J_{-\nu}(x)}{\sin(\nu\pi)}$$

Arguments:

Name	Type	Description
x	number	The value at which to evaluate the function.
n	number	The order of the Bessel function. This value is truncated to an integer.

Return Type and Value: number – The Weber's Bessel function Yn(x).

However, if *n* < 0, `#NUM!` is returned.

[*Example*:

`BESSELY(2.345,5)` results in `-4.98977884`

end example]

3.17.7.27 BETADIST

Syntax:

BETADIST (*x* , *alpha* , *beta* [, [*A*] , [*B*]])

Description: Computes the cumulative beta probability density function.

Arguments:

Name	Type	Description
x	number	The value between A and B at which to evaluate the function.
alpha	number	A parameter of the distribution.
beta	number	A parameter of the distribution.
A	number	The lower bound to the interval of x. If omitted, the lower bound is 0.
B	number	The upper bound to the interval of x. If omitted, the upper bound is 1.

Return Type and Value: number – The cumulative beta probability density function.

However, if

- *alpha* or *beta* ≤ 0, `#NUM!` is returned.
- $x < A$, $x > B$, or $A = B$, `#NUM!` is returned.

[*Example*:

`BETADIST(0.5,1,2)` results in `0.75`
`BETADIST(0.5,1,2,-4.5,7.3)` results in `0.66791152`
`BETADIST(0.5,1,2,,2.3)` results in `0.387523629`

end example]

3.17.7.28 BETAINV

Syntax:

BETAINV (*probability* , *alpha* , *beta* [, [*A*] , [*B*]])

Description: Computes the inverse of the cumulative distribution function for a specified beta distribution. Given a value for *probability*, BETAINV is used to seek for the value x such that BETADIST(*x*, *alpha*, *beta*,

A, *B*) = *probability*. Thus, precision of BETAINV depends on precision of BETADIST. BETAINV uses an iterative search technique.

Arguments:

Name	Type	Description
probability	number	A probability associated with the beta distribution.
alpha	number	A parameter of the distribution.
beta	number	A parameter of the distribution.
A	number	The lower bound to the interval of *x*. If omitted, the lower bound is 0.
B	number	The upper bound to the interval of *x*. If omitted, the upper bound is 1.

Return Type and Value: number – The inverse of the cumulative distribution function for a specified beta distribution.

However, if

- *alpha* or *beta* ≤ 0, #NUM! is returned.
- *probability* < 0 or *probability* > 1, #NUM! is returned.
- The search has not converged after some implementation-defined number of iterations, #N/A is returned.

[*Example*:

BETAINV(0.5,1,2) results in 0.29289341
BETAINV(0.5,1,2,-4.5,7.3) results in -1.043857765
BETAINV(0.5,1,2,,2.3) results in 0.673654842

end example]

3.17.7.29 BIN2DEC

Syntax:

BIN2DEC (*number*)

Description: Makes the decimal equivalent of *number*.

Arguments:

Name	Type	Description
number	number	A 10-digit binary number that is to be converted to a

Name	Type	Description
		decimal string. If *number* has less than 10 digits, leading zero digits are implied until it has exactly 10 digits. The 10 digits use twos-complement representation with the left-most bit (10th bit from the right) representing the sign bit.

Return Type and Value: number – The decimal equivalent of *number*.

However, if

- *number* contains one or more non-binary digits, `#NUM!` is returned.
- *number* contains more than 10 binary digits; that is, *number* is outside the range `1000000000` (-512 decimal) to `0111111111` (511 decimal), inclusive, `#NUM!` is returned.

[*Example*:

`BIN2DEC(111)` results in 7
`BIN2DEC(11111111)` results in 255
`BIN2DEC(1111111110)` results in -2
`BIN2DEC(1000000000)` results in -512

end example]

3.17.7.30 BIN2HEX

Syntax:

`BIN2HEX ( number [ , num-hex-digits ] )`

Description: Makes the uppercase hexadecimal equivalent of *number*, with the result having *num-hex-digits* digits.

Arguments:

Name	Type	Description
number	number	A 10-digit binary number that is to be converted to a hexadecimal string. If *number* has less than 10 digits, leading zero digits are implied until it has exactly 10 digits. The 10 digits use twos-complement representation with the left-most bit (10th bit from the right) representing the sign bit.
num-hex-digits	number	The number of digits in the result, with leading zeros added as necessary. However, if *number* is negative, *num-hex-digits* is ignored and the result has 10 digits. If *num-*

Name	Type	Description
		hex-digits is omitted, the minimum number of digits is used in the result. *num-hex-digits* is truncated to an integer.

Return Type and Value: text – The uppercase hexadecimal equivalent of *number*.

However, if

- *number* contains one or more non-binary digits, `#NUM!` is returned.
- *number* contains more than 10 binary digits; that is, *number* is outside the range `1000000000` (200 hex, -512 decimal) to `0111111111` (1FF hex, 511 decimal), inclusive, `#NUM!` is returned.
- *number* needs more digits that *num-hex-digits*, `#NUM!` is returned.
- *num-hex-digits* ≤ 0 or > 10, `#NUM!` is returned.

[*Example*:

`BIN2HEX(1)` results in `1`
`BIN2HEX(1,4)` results in `0001`
`BIN2HEX(111111)` results in `3F`
`BIN2HEX(1111000000)` results in `FFFFFFFFC0`
`BIN2HEX(1000000000,3)` results in `FFFFFFFE00`

end example]

3.17.7.31 BIN2OCT

Syntax:

`BIN2OCT ( number [ , num-oct-digits ] )`

Description: Makes the octal equivalent of *number*, with the result having *num-oct-digits* digits.

Arguments:

Name	Type	Description
number	number	A 10-digit binary number that is to be converted to an octal string. If *number* has less than 10 digits, leading zero digits are implied until it has exactly 10 digits. The 10 digits use twos-complement representation with the left-most bit (10th bit from the right) representing the sign bit.
num-oct-digits	number	*num-oct-digits* is the number of digits in the result, with leading zeros added as necessary. However, if *number* is negative, *num-oct-digits* is ignored and the result has

Name	Type	Description
		10 digits. If *num-oct-digits* is omitted, the minimum number of digits is used in the result. *num-oct-digits* is truncated to an integer.

Return Type and Value: text – The octal equivalent of *number*.

However, if

- *number* contains one or more non-binary digits, `#NUM!` is returned.
- *number* contains more than 10 binary digits; that is, *number* is outside the range `1000000000` (1000 octal, -512 decimal) to `0111111111` (0777 octal, 511 decimal), inclusive, `#NUM!` is returned.
- *number* needs more digits that *num-oct-digits*, `#NUM!` is returned.
- *num-oct-digits* < 0 or > 10, `#NUM!` is returned.

[*Example*:

`BIN2OCT(1)` results in `1`
`BIN2OCT(1,4)` results in `0001`
`BIN2OCT(111111)` results in `77`
`BIN2OCT(1111000000)` results in `7777777700`
`BIN2OCT(1000000000,3)` results in `7777777000`

end example]

3.17.7.32 BINOMDIST

Syntax:

`BINOMDIST` (*number-successes* , *number-trials* , *success-probability* , *cumulative-flag*)

Description: Computes the individual term binomial distribution probability.

Mathematical Formula:

The binomial probability mass function is:

$$b(x;n,p) = \binom{n}{x} p^x (1-p)^{n-x}$$

where:

$$\binom{n}{x}$$

is `COMBIN(n,x)`.

The cumulative binomial distribution is:

$$B(x;n,p) = \sum_{y=0}^{x} b(y;n,p)$$

Arguments:

Name	Type	Description
number-successes	number	The number of successes in *number-trials*, truncated to an integer.
number-trials	number	The number of independent trials, truncated to an integer.
success-probability	number	The probability of success on each trial.
cumulative-flag	logical	Determines the form of the function. If TRUE, then the cumulative distribution function is returned, which is the probability that there are at most *number-successes* successes; if FALSE, the probability mass function is returned, which is the probability that there are *number-successes* successes.

Return Type and Value: number – The individual term binomial distribution probability.

However, if

- *number-successes* < 0 or *number-successes* > *number-trials*, `#NUM!` is returned.
- *success-probability* < 0 or *success-probability* > 1, `#NUM!` is returned.

[*Example*:

`BINOMDIST(6,10,0.5,FALSE)` results in `0.205078125`
`BINOMDIST(6,10,0.5,TRUE)` results in `0.828125`

end example]

3.17.7.33 CEILING

Syntax:

`CEILING ( x , significance )`

Description: Computes a value that is *x* rounded-up, away from zero, to the nearest multiple of *significance*. Regardless of the sign of *x*, a value is rounded up when adjusted away from zero.

Arguments:

Name	Type	Description
x	number	The value to be rounded
significance	number	The multiple to which *x* is to be rounded.

Return Type and Value: number – The rounded-up value of *x*.

However, if *x* and *significance* have different signs, `#NUM!` is returned.

[*Example*:

`CEILING(2.5,1)` rounds 2.5 up to nearest multiple of 1; that is, to 3
`CEILING(-2.5,-2)` rounds -2.5 up to nearest multiple of -2; that is, to `-4`
`CEILING(1.5,0.1)` rounds 1.5 up to the nearest multiple of 0.1; that is, to `1.5`
`CEILING(0.234,0.01)` rounds 0.234 up to the nearest multiple of 0.01; that is, to `0.24`

end example]

3.17.7.34 CELL

Syntax:

`CELL` (*category* [, *reference*])

Description: Retrieves information about the formatting, location, or contents of the upper-left cell indicated by *reference*. *category* indicates the kind of information to be retrieved.

Arguments:

Name	Type	Description
category	text	The category string as defined in the table following.
reference	reference	Refers to the cell whose category information is being requested. If *reference* is a cell range, the first cell in that range is the cell whose category information is being requested. If *reference* is omitted, the information retrieved pertains to the most recent cell whose value was changed. For the category `"format"`, if *reference* designates a cell formatted with a built-in number format, the number format string is as defined in the table following.

category	Meaning	Result Type
`"address"`	Reference of the first cell in *reference*.	text
`"col"`	Column number of the cell in *reference*.	number

category	Meaning	Result Type
"color"	1 if the cell is formatted in color for negative values; otherwise, 0. 0 if the cell does not contain a number.	number
"contents"	Value of the upper-left cell in *reference*.	Text or number
"filename"	Fully qualified filename of the file that contains *reference*. However, if the worksheet that contains *reference* has not yet been saved, the filename is an empty string.	text
"format"	Number format of the cell. (See the table of formats below.) The number format string has "-" appended if the cell is formatted in color for negative values. The number format string has "()" appended if the cell is formatted in color for positive or all values.	text
"parentheses"	1 if the cell is formatted with parentheses for positive or all values; otherwise, 0. 0 if the cell does not contain a number.	number
"prefix"	Text value corresponding to the label prefix of the cell, as follows: • Single quotation mark (') if the cell contains left-aligned text • Double quotation mark (") if the cell contains right-aligned text • Caret (^) if the cell contains centered text • Backslash (\) if the cell contains fill-aligned text • Empty string if the cell contains anything else	text
"protect"	0 if the cell is not locked; otherwise, 1.	number
"row"	Row number of the cell in reference.	number
"type"	Text value corresponding to the type of data in the cell. • "b" (blank) if the cell is empty • "l" (label) if the cell contains a text constant • "v" (value) if the cell contains anything else	text
"width"	Column width of the cell rounded off to an integer. Each unit of column width is equal to the width of one character in the default font size.	number

Format	Number Format String
0	"F0"
0.00	"F2"
#,##0	",0"
#,##0.00	",2"
$#,##0_);($#,##0)	"C0"

Format	Number Format String
$#,##0_);[Red]($#,##0)	"C0-"
$#,##0.00_);($#,##0.00)	"C2"
$#,##0.00_);[Red]($#,##0.00)	"C2-"
0%	"P0"
0.00%	"P2"
0.00E+00	"S2"
General	"G"
# ?/? # ??/??	"G"
d-mmm-yy dd-mmm-yy	"D1"
d-mmm dd-mmm	"D2"
mmm-yy	"D3"
m/d/yy m/d/yy h:mm mm/dd/yy	"D4"
mm/dd	"D5"
h:mm:ss AM/PM	"D6"
h:mm AM/PM	"D7"
h:mm:ss	"D8"
h:mm	"D9"

Return Type and Value: various (see table above) – The value corresponding to *category*, and whose type is shown in the category value table above.

However, if *category* is invalid, #VALUE! is returned.

[*Example*:

CELL("address",A10) might result in $E$289
CELL("contents",A10:B10), results in xxx, when A10 contains xxx, and B10 contains anything
CELL("filename",A10) might result in E:\Formulas\[Test.xlsx]Sheet1
CELL("format",A10) results in G, when A10 contains xxx
CELL("format",A10) results in F2-, when A10 contains (123.00)
CELL("format",A10) results in C3-, when A10 contains $123,456.780
CELL("format",A10) results in S3, when A10 contains 1.235E+05
CELL("prefix",A10) results in ', when A10 contains xxx
CELL("type",A10) results in l, when A10 contains xxx

end example]

3.17.7.35 CHAR

Syntax:

CHAR (*x*)

Description: Determines the character that is represented by the value *number*.

Arguments:

Name	Type	Description
x	number	A value in the range 1–255, which designates the character.

Return Type and Value: text – The character represented by the value *number*.

[*Example*:

CHAR(65) results in A
CHAR(A10) results in A, when A10 contains 65

end example]

3.17.7.36 CHIDIST

Syntax:

CHIDIST (*x* , *degrees-freedom*)

Description: Computes the one-tailed probability of the chi-squared distribution.

Mathematical Formula:

CHIDIST = P(X>x), where X is a $\chi 2$ random variable.

Arguments:

Name	Type	Description
x	number	The value at which the distribution is to be evaluated.
degrees-freedom	number	The number of degrees of freedom, truncated to an integer.

Return Type and Value: number – The one-tailed probability of the chi-squared distribution.

However, if

- $x < 0$, #NUM! is returned.
- *degrees-freedom* < 1 or *degrees-freedom* > 10^{10}, #NUM! is returned.

[*Example*:

CHIDIST(3.5,4) results in 0.47787835
CHIDIST(12.34,7) results in 0.089917721

end example]

3.17.7.37 CHIINV

Syntax:

CHIINV (*probability* , *degrees-freedom*)

Description: Computes the inverse of the one-tailed probability of the chi-squared distribution. Given a value for *probability*, CHIINV seeks for a value *x* such that CHIDIST(*x*, *degrees-freedom*) = *probability*. Thus, precision of CHIINV depends on precision of CHIDIST. CHIINV uses an iterative search technique.

Arguments:

Name	Type	Description
probability	number	A probability associated with the chi-squared distribution.
degrees-freedom	number	The number of degrees of freedom, truncated to an integer.

Return Type and Value: number – The inverse of the one-tailed probability of the chi-squared distribution.

However, if

- *probability* < 0 or *probability* > 1, #NUM! is returned.
- *degrees-freedom* < 1 or *degrees-freedom* ≥ 10^{10}, #NUM! is returned.
- The search has not converged after some implementation-defined number of iterations, #N/A is returned

[*Example*:

CHIINV(0.5,4) results in 3.356694001
CHIINV(0.3,7) results in 8.38343064

end example]

3.17.7.38 CHITEST

Syntax:

CHITEST (*actual-range* , *expected-range*)

Description: Computes the test for independence. CHITEST returns the value from the chi-squared distribution for the statistic and the appropriate degrees of freedom.

Mathematical Formula:

The χ2 test first calculates a χ2 statistic using the formula:

$$\chi^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{\left(A_{ij} - E_{ij}\right)^2}{E_{ij}}$$

where:

Aij = actual frequency in the i-th row, j-th column
Eij = expected frequency in the i-th row, j-th column
r = number or rows
c = number of columns

CHITEST uses the χ2 distribution with an appropriate number of degrees of freedom, df. If r > 1 and c > 1, then df = (r - 1)(c - 1). If r = 1 and c > 1, then df = c - 1 or if r > 1 and c = 1, then df = r - 1.

Arguments:

Name	Type	Description
actual-range	reference	The range of data that contains observations to test against expected values.
expected-range	reference	The range of data that contains the ratio of the product of row totals and column totals to the grand total.

Return Type and Value: number – The value from the chi-squared distribution for the statistic and the appropriate degrees of freedom.

However, if:

- The number of rows and columns is exactly one, the return value is unspecified.
- *actual-range* and *expected-range* have a different number of data points, the return value is unspecified.

[*Example*: Given the following data:

	A	B	C

	A	B	C
1	Men (Actual)	Women (Actual)	Description
2	58	35	Agree
3	11	25	Neutral
4	10	23	Disagree
5	Men (Expected)	Women (Expected)	Description
6	45.35	47.65	Agree
7	17.56	18.44	Neutral
8	16.09	16.91	Disagree

`CHITEST(A2:B4,A6:B8)` results in `0.000308`

end example]

3.17.7.39 CHOOSE

Syntax:

CHOOSE (*index* , *argument-list*)

Description: Selects the *argument* in *argument-list* that corresponds by position to *index*.

Arguments:

Name	Type	Description
index	number	An index into *argument-list*, truncated to an integer. The value of *index* shall be in the position range 1–*n*, where *argument-1* is position 1, *argument-2* is position 2, and so on up to *argument-n*. If *index* is an array, the value or every element in that array is evaluated, and if the formula was array-entered, the result is an array of chosen values.
argument-list	any	The *argument*s in any given *argument-list* need not all have the same type.

Return Type and Value: any, including array – The *argument* in *argument-list* that corresponds by position to *index*.

However, if the value of *index* is not an index into *argument-list*, #VALUE! is returned.

[*Example*:

CHOOSE(E7,F7,G7,H7,I7,J7,K7,L7) results in Monday, when E7 contains 2, and the cells F7:L7 each contain the names of the week, from Sunday to Saturday

SUM(CHOOSE(E1,F20:G20,H20:J24)) results in the sum of the elements designated by F20:G20 or H20:J24, as determined by the value of E1

If B9:B11 contain 1, 3, and 3, respectively, and CHOOSE(B9:B11,10,20,30) is array-entered into 3 cells, the values of those 3 cells is 10, 30, and 30, respectively.

end example]

3.17.7.40 CLEAN

Syntax:

CLEAN (*string*)

Description:

Makes a string that is a copy of *string* with all so-called "non-printable" characters—those with internal values in the range U+0000–001F—removed.

Arguments:

Name	Type	Description
string	text	Designate the string to be cleaned.

Return Type and Value: text – The trimmed copy of *string*.

[*Example*:

CLEAN("A" & CHAR(2) & "BC") results in ABC, which is stored in A10
LEN(A10) results in 3

end example]

3.17.7.41 CODE

Syntax:

CODE (*string*)

Description: Determines the numeric code of the first character in *string*.

Arguments:

Name	Type	Description

Name	Type	Description
string	text	Designates a string containing one or more characters.

Return Type and Value: number – The numeric code of the first character in *string*.

However, if *string* is empty, #VALUE! is returned.

[*Example*:

CODE("abc") results in 97
CODE(A10) results in 97, when A1 contains abc

end example]

3.17.7.42 COLUMN

Syntax:

COLUMN ([*reference*])

Description: Finds the number of the column(s) corresponding to *reference*.

Arguments:

Name	Type	Description
reference	reference	A reference to a single cell or to a range of contiguous cells. If omitted, the behavior is as if *reference* referred to the cell containing the formula.

Return Type and Value: number – If *reference* refers to a single cell or to a single column of cells, the corresponding column is returned. If *reference* refers to a range of cells involving multiple columns, a horizontal array of the corresponding columns as numbers is returned.

However, if the range of cells referred to by *reference* is not contiguous, #REF! is returned.

[*Example*:

COLUMN() results in 4, when the cell containing the formula is in column 4
COLUMN(E17:E19) results in 5
COLUMN(E16:F17) results in a horizontal array containing 5and 6, respectively

end example]

3.17.7.43 COLUMNS

Syntax:

`COLUMNS ( array )`

Description: Finds the number of columns corresponding to *array*.

Arguments:

Name	Type	Description
array	array, reference	Any array.

Return Type and Value: number – The number of columns corresponding to *array*.

However, if the range of cells referred to by *array* is not contiguous, `#NULL!` is returned.

[*Example*:

`COLUMNS(E16:F16)` results in 2
`COLUMNS(E16:G18)` results in 3
`COLUMNS({1,2;3,4})` results in 2

end example]

3.17.7.44 COMBIN

Syntax:

`COMBIN ( number , number-chosen )`

Description: Computes the possible number of groups of size *number-chosen* that can be formed from *number* objects. [*Note*: A combination is any set or subset of objects, regardless of their internal order. Combinations are distinct from permutations, for which the internal order is significant. *end note*]

Mathematical Formula:

The number of combinations is as follows, where *number* = n and *number-chosen* = k:

$$\binom{n}{k} = \frac{P_{k,n}}{k!} = \frac{n!}{k!(n-k)!}$$

where:

$$P_{k,n} = \frac{n!}{(n-k)!}$$

Arguments:

Name	Type	Description
number	number	The total number of objects available, truncated to an integer.
number-chosen	number	The number of objects in each combination, truncated to an integer.

Return Type and Value: number – The number of different combinations of *number-chosen* in *number*.

However, if

- *number* < 0, `#NUM!` is returned.
- *number-chosen* < 0, `#NUM!` is returned.
- *number* < *number-chosen*, `#NUM!` is returned.

[*Example*:

`COMBIN(8,2)` results in `28`
`COMBIN(10,4)` results in `210`
`COMBIN(6,5)` results in `6`

end example]

3.17.7.45 COMPLEX

Syntax:

`COMPLEX ( real-number , imaginary-number [ , suffix ] )`

Description: Makes a complex number in *x + yi* or *x + yj* text format from the arguments.

Arguments:

Name	Type	Description
real-number	number	The real number coefficient.
imaginary-number	number	The imaginary number coefficient.
suffix	text	"`i`" or "`j`". If omitted, "`i`" is used.

Return Type and Value: text – The complex number string specified by the arguments.

If *real-number* has the value 0 and *imaginary-number* has a non-zero value, the resulting string contains just the real number. If *real-number* has a non-zero value and *imaginary-number* has a zero value, the resulting string contains just the imaginary number and suffix. If both *real-number* and *imaginary-number* have a zero value, the resulting string is "`0`".

However, if *suffix* is neither "`i`" nor "`j`", `#VALUE!` is returned.

[*Example*:

`COMPLEX(-3.5,19.6)` results in `-3.5+19.6i`
`COMPLEX(3.5,-19.6,"j")` results in `3.5-19.6j`
`COMPLEX(3.5,0)` results in `3.5`
`COMPLEX(0,2.4)` results in `2.4i`
`COMPLEX(0,0)` results in `0`

end example]

3.17.7.46 CONCATENATE

Syntax:

CONCATENATE (*argument-list*)

Description: Makes a string that is the concatenation of all the strings corresponding to the *argument*s in *argument-list*, taken left-to-right.

Arguments:

Name	Type	Description
argument-list	text	Each *argument* in *argument-list* shall designate a string.

Return Type and Value: text – The concatenated string.

[*Example*:

`CONCATENATE("text")` results in `text`
`CONCATENATE("The total is ",A10," units")` results in `The total is 43 units`, when `A10` contains `43`
`CONCATENATE(3," + ",4," = ",3+4)` results in `3 + 4 = 7`

end example]

3.17.7.47 CONFIDENCE

Syntax:

CONFIDENCE (*alpha* , *standard-dev* , *size*)

Description: Computes a value that can be used to construct a confidence interval for a population mean.

Arguments:

Name	Type	Description
alpha	number	The significance level used to compute the confidence level.
standard-dev	number	The population standard deviation for the data range.
size	number	The sample size, truncated to an integer.

Return Type and Value: number – A value that can be used to construct a confidence interval for a population mean.

However, if

- *alpha* ≤ 0 or *alpha* ≥ 1, `#NUM!` is returned.
- *standard-dev* ≤ 0, `#NUM!` is returned.
- *size* < 1, `#NUM!` is returned.

[*Example*:

`CONFIDENCE(0.4,5,12)` results in `1.214775614`
`CONFIDENCE(0.75,9,7)` results in `1.083909234`

end example]

3.17.7.48 CONVERT

Syntax:

`CONVERT` (*number* , *from-unit* , *to-unit*)

Description: Converts a number from one measurement system to another.

Arguments:

Name	Type	Description
number	number	The value to be converted from *from-unit*s to *to-unit*s.
from-unit	text	The unit to be converted from, where the valid string values are shown in the tables below.
to-unit	text	The unit to be converted to, where the valid string values are shown in the tables below.

Weight and Mass	
Unit String	*Meaning*
g	Gram

Weight and Mass	
lbm	Pound mass (avoirdupois)
ozm	Ounce mass (avoirdupois)
sg	Slug
u	U (atomic mass unit)

Distance	
Unit String	*Meaning*
ang	Angstrom
ft	Foot
in	Inch
m	Meter
mi	Statute mile
Nmi	Nautical mile
Pica	Pica (1/72 inch)
yd	Yard

Time	
Unit String	*Meaning*
day	Day
hr	Hour
mn	Minute
sec	Second
yr	Year

Pressure	
Unit String	*Meaning*
at or atm	Atmosphere
mmHg	mm of Mercury
P or p	Pascal

Force	
Unit String	*Meaning*

Force	
dy or dyn	Dyne
lbf	Pound force
N	Newton

Energy	
Unit String	*Meaning*
BTU or btu	BTU
c	Thermodynamic calorie
cal	IT calorie
e	Erg
ev or eV	Electron volt
flb	Foot-pound
HPh or hh	Horsepower-hour
J	Joule
Wh or wh	Watt-hour

Power	
Unit String	*Meaning*
H or hp	Horsepower
W or w	Watt

Magnetism	
Unit String	*Meaning*
ga	Gauss
T	Tesla

Temperature	
Unit String	*Meaning*
C or cel	Degrees Celsius
F or fah	Degrees Fahrenheit
K or kel	Degrees Kelvin

Liquid Measure	
Unit String	*Meaning*
`cup`	Cup
`gal`	U.S. Gallon
`l` or `lt`	Liter
`oz`	Fluid ounce
`pt`or `us_pt`	U.S. pint
`qt`	U.S. Quart
`tbs`	Tablespoon
`tsp`	Teaspoon
`uk_pt`	U.K. pint

The following abbreviated unit prefixes can be used with any metric unit:

Abbreviated Unit Prefixes	
Prefix String	*Meaning*
`E`	exa (1E+18)
`P`	peta (1E+15)
`T`	tera (1E+12)
`G`	giga (1E+09)
`M`	mega (1E+06)
`k`	kilo (1E+03)
`h`	hecto (1E+02)
`e`	dekao (1E+01)
`d`	deci (1E-01)
`c`	centi (1E-02)
`m`	milli (1E-03)
`u`	micro (1E-06)
`n`	nano 1E-09)
`p`	pico (1E-12)
`f`	femto (1E-15)
`a`	atto (1E-18)

Unit names and prefixes are case-sensitive.

Return Type and Value: number – The value of *number* in *from-unit*s converted to *to-unit*s.

However, if

- The value of *from-unit* or *to-unit* is invalid, #N/A is returned.
- The *from-unit* and *to-unit* are from different measurement categories, #N/A is returned.
- The value of *from-unit* or *to-unit* has an abbreviated unit prefix, yet none is supported for that unit, #N/A is returned.

[*Example*:

CONVERT(10,"ozm","g") results in 283.4951521
CONVERT(1,"yd","mm") results in 914.4000003
CONVERT(1,"yd","cm") results in 91.44000003
CONVERT(1,"yd","m") results in 0.9144
CONVERT(1,"yd","km") results in 0.0009144
CONVERT(1,"mi","Nmi") results in 0.868976242
CONVERT(1,"day","sec") results in 86400
CONVERT(0,"K","C") results in -273.15

end example]

3.17.7.49 CORREL

Syntax:

CORREL (*array-1* , *array-2*)

Description: Computes the correlation coefficient of the two cell ranges designated by *array-1* and *array-2*.

Mathematical Formula:

The equation for the correlation coefficient is:

$$Correl(X,Y) = \frac{\sum (x-\bar{x})(y-\bar{y})}{\sqrt{\sum (x-\bar{x})^2 \sum (y-\bar{y})^2}}$$

where x and y are the sample means AVERAGE(*array-1*) and AVERAGE(*array-2*).

Arguments:

Name	Type	Description
array-1	array, reference	The first cell range. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.
array-2	array, reference	The second cell range. If an array or reference argument contains text, logical values, or empty cells, those values

Name	Type	Description
		are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The correlation coefficient of the cells in two cell ranges.

However, if

- *array-1* and *array-2* have a different number of data points, the return value is unspecified.
- *array-1* and *array-2* is empty, the return value is unspecified.
- The standard deviation of the values in *array-1* or *array-2* equals zero, the return value is unspecified.

[*Example*:

`CORREL({2.532,5.621;2.1,3.4},{5.32,2.765;5.2,6.7})` results in `-0.714976`

end example]

3.17.7.50 COS

Syntax:

COS (*x*)

Description: Computes the cosine of *x*.

Arguments:

Name	Type	Description
x	number	The value whose cosine is to be determined.

Return Type and Value: number – The cosine of *x*.

[*Example*:

`COS(-1)` results in `0.540302306`
`COS(0)` results in `1`
`COS(1)` results in `0.540302306`

end example]

3.17.7.51 COSH

Syntax:

COSH (*x*)

Description: Computes the hyperbolic cosine of *x*.

Arguments:

Name	Type	Description
x	number	The value whose hyperbolic cosine is to be determined.

Return Type and Value: number – The hyperbolic cosine of *x*.

However, if the magnitude of *x* is too large, `#NUM!` is returned.

[*Example*:

`COSH(-1)` results in `1.543080635`
`COSH(0)` results in `1`
`COSH(1)` results in `1.543080635`

end example]

3.17.7.52 COUNT

Syntax:

COUNT (*argument-list*)

Description: Counts the number of *argument*s in *argument-list* that contain numbers, and the number of cells referred to by *argument*s in *argument-list*, which contain numbers.

Arguments:

Name	Type	Description
argument-list	text	Each *argument* in *argument-list* designates a value. Arguments that are numbers, logical values, dates, or text representations of numbers shall be counted. If an argument is an array or reference, only numbers in that array or reference shall be counted. Empty cells, logical values, text, or error values in the array or reference shall be ignored. [*Note*: To count logical values, text, or error values as well, use the COUNTA (§3.17.7.53) function. *end note*]

Return Type and Value: number – The numeric argument and reference to numeric argument count.

[*Example*:

COUNT(1,2,3,4,5) results in 5
COUNT({1,2,3,4,5}) results in 5
COUNT({1,2,3,4,5},6,"7") results in 7
COUNT(10,E1), where E1 is an empty cell, results in 1, as E1 is ignored
COUNT(10,E2), where E2 contains TRUE, results in 1, as E2 is ignored

end example]

3.17.7.53 COUNTA

Syntax:

COUNTA (*argument-list*)

Description: Counts the number of arguments that are not cell references, and the number of cells, referred to by arguments, which are not empty.

Arguments:

Name	Type	Description
argument-list	text	Each *argument* in *argument-list* designates a value. Arguments with values of any type shall be counted. However, empty cells shall not be counted. If an argument is an array or reference, only values in that array or reference shall be counted. Empty cells and text values in the array or reference shall be ignored. [*Note*: To exclude logical values, text, or error values, use the COUNT (§3.17.7.52) function. *end note*]

Return Type and Value: number – The number of arguments that are not cell references, and the number of cells, referred to by arguments, which are not empty.

[*Example*:

COUNTA(1,2,3,4,5) results in 5
COUNTA({1,2,3,4,5}) results in 15
COUNTA({1,2,3,4,5},6,"7") results in 7
COUNTA(10,E1), where E1 is an empty cell, results in 1, as E1 is ignored
COUNTA(10,E2), where E2 contains TRUE, results in 2, as E2 is counted

end example]

3.17.7.54 COUNTBLANK

Syntax:

COUNTBLANK (*cell-range*)

Description: Counts the number of cells in a specified range of cells, which are empty. A cell containing a formula that returns an empty string is counted, whereas a cell containing a zero value is not.

Arguments:

Name	Type	Description
cell-range	reference	Designates the range of cells to be inspected.

Return Type and Value: number – The number of empty cells in the range specified.

[*Example*:

`COUNTBLANK(A2:C2)`, where A2 and B2 are empty, but C2 is not, results in 2

end example]

3.17.7.55 COUNTIF

Syntax:

COUNTIF (*cell-range* , *selection-criteria*)

Description: Counts the number of cells in a specified range of cells, whose values meet the specified criteria.

Arguments:

Name	Type	Description
cell-range	reference	Designates the range of cells to be inspected.
selection-criteria	number, expression, reference, text	Designates the cells to be counted. In the case of text, *selection-criteria* can consist of any comparison operator followed by the operand against which each cell's value is to be compared. *selection-criteria* can include one or more wildcard characters, question mark (?) and asterisk (*). A question mark matches any single character; an asterisk matches any sequence of characters. To search for a question mark, asterisk, or tilde character, prefix that character with a tilde (~).

Return Type and Value: number – The number of cells in the range specified that meet the criteria.

[*Example*: Given that A1, B1, C1, and D1, respectively, contain the values 3, 10, 7, and 10

`COUNTIF(A1:D1,"=10")` results in 2

`COUNTIF(A1:D1,">5")` results in `30`
`COUNTIF(A1:D1,"<>10")` results in 2

Given that `A2`, `B2`, `C2`, and `D2`, respectively, contain the values `apples`, `oranges`, `grapes`, and `melons`

`COUNTIF(A2:D2,"*es")` results in 3
`COUNTIF(A2:D2,"??a*")` results in 2
`COUNTIF(A2:D2,"*l*")` results in 2

end example]

3.17.7.56 COUNTIFS

Syntax:

`COUNTIFS (` *count-range* `,` *cell-range-1* `,` *selection-criteria-1*
`[ ,` *cell-range-2* `,` *selection-criteria-2* `[ , …] ] )`

Description: Counts the number of cells within a range that meet multiple criteria.

Arguments:

Name	Type	Description
count-range	reference	Designates the cells whose values are included. *count-range* does not have to have the same size and shape as *cell-range-1* through *cell-range-n*. The actual cells that are added are determined by using the top, left cell in *count-range* as the beginning cell, and then including cells that correspond in size and shape to *cell-range-1* through *cell-range-n*.
cell-range-1	reference	Designates the first range of cells to be inspected. Each cell in a range is counted only if all of the corresponding criteria specified are true for that cell.
selection-criteria-1	number, expression, reference, text	Designates the first range of cells to be counted. In the case of text, *selection-criteria-1* can consist of any comparison operator followed by the operand against which each cell's value is to be compared. *selection-criteria* can include one or more wildcard characters, question mark (?) and asterisk (*). A question mark matches any single character; an asterisk matches any sequence of characters. To search for a question mark, asterisk, or tilde character, prefix that character with a tilde (~).
cell-range-n	reference	The optional arguments *selection-criteria-2* through *selection-criteria-n* have corresponding arguments *cell-range-2* through *cell-range-n*, and have the same semantics as *selection-criteria-1* and *cell-range-1*,
selection-criteria-n	number, expression,	

Name	Type	Description
	reference, text	respectively.

If a cell in any argument is an empty cell, it is treated as if it had the value 0.

Return Type and Value: number – The count of the cells corresponding to those selected.

[*Example*: Given the following data:

	A	B	C	D
1	Sales Person	Exceeded Tables Quota	Exceeded Chairs Quota	Exceeded Desks Quota
2	Davolio	Yes	No	No
3	Buchanan	Yes	Yes	No
4	Suyama	Yes	Yes	Yes
5	Leverling	No	Yes	Yes

`COUNTIFS(B2:D2,"=Yes")` results in 1 (counts how many times Davolio exceeded a sales quota for tables, chairs, and desks)

`COUNTIFS(B2:B5,"=Yes",C2:C5,"=Yes")` results in 2 (counts how many sales people exceeded both their tables and chairs quota)

`COUNTIFS(B5:D5,"=Yes",B3:D3,"=Yes")` results in 1 (counts how many times Leverling and Buchanan exceeded the same quota for tables, chairs, and desks)

end example]

3.17.7.57 COUPDAYBS

Syntax:

COUPDAYBS (*settlement* , *maturity* , *frequency* [, [*basis*]])

Description: Computes the number of days from the beginning of the coupon period to the settlement date.

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
frequency	number	The number of coupon payments per year. For annual payments, frequency is 1; for semiannual payments,

Name	Type	Description
		frequency is 2; for quarterly payments, frequency is 4. *frequency* is truncated to an integer.
basis	number	The truncated integer type of day count basis to use, as follows: **Value** / **Day Count Basis** 0 or omitted / US (NASD) 30/360 1 / Actual/actual 2 / Actual/360 3 / Actual/365 4 / European 30/360

Time information in the date arguments is ignored.

Return Type and Value: number – The number of days from the beginning of the coupon period to the settlement date.

However, if

- *settlement* or *maturity* is out of range for the current date base value, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`COUPDAYBS(DATE(2007,1,25),DATE(2008,11,15),2,1)` results in `71`
`COUPDAYBS(DATE(2007,1,25),DATE(2008,11,15),2)` results in `70`

end example]

3.17.7.58 COUPDAYS

Syntax:

COUPDAYS (*settlement* , *maturity* , *frequency* [, [*basis*]])

Description: Computes the number of days in the coupon period that contains the settlement date.

Arguments:

Name	Type	Description

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
frequency	number	The number of coupon payments per year. For annual payments, frequency is 1; for semiannual payments, frequency is 2; for quarterly payments, frequency is 4. *frequency* is truncated to an integer.
basis	number	The truncated integer type of day count basis to use, as follows: Value / Day Count Basis 0 or omitted — US (NASD) 30/360 1 — Actual/actual 2 — Actual/360 3 — Actual/365 4 — European 30/360

Time information in the date arguments is ignored.

Return Type and Value: number – The number of days in the coupon period that contains the settlement date.

However, if

- *settlement* or *maturity* is out of range for the current date base value, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`COUPDAYS(DATE(2007,1,25),DATE(2008,11,15),2,1)` results in `181`
`COUPDAYS(DATE(2007,1,25),DATE(2008,11,15),2)` results in `180`

end example]

3.17.7.59 COUPDAYSNC

Syntax:

COUPDAYSNC (*settlement* , *maturity* , *frequency* [, [*basis*]])

Description: Computes the number of days from the settlement date to the next coupon date.

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
frequency	number	The number of coupon payments per year. For annual payments, frequency is 1; for semiannual payments, frequency is 2; for quarterly payments, frequency is 4. *frequency* is truncated to an integer.
basis	number	The truncated integer type of day count basis to use, as follows:

Value	Day Count Basis
0 or omitted	US (NASD) 30/360
1	Actual/actual
2	Actual/360
3	Actual/365
4	European 30/360

Time information in the date arguments is ignored.

Return Type and Value: number – The number of days from the settlement date to the next coupon date.

However, if

- *settlement* or *maturity* is out of range for the current date base value, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`COUPDAYSNC(DATE(2007,1,25),DATE(2008,11,15),2,1)` results in `110`
`COUPDAYSNC(DATE(2007,1,25),DATE(2008,11,15),2)` results in `110`

end example]

3.17.7.60 COUPNCD

COUPNCD (*settlement* , *maturity* , *frequency* [, [*basis*]])

Description: Computes the next coupon date after the settlement date.

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
frequency	number	The number of coupon payments per year. For annual payments, frequency is 1; for semiannual payments, frequency is 2; for quarterly payments, frequency is 4. *frequency* is truncated to an integer.
basis	number	The truncated integer type of day count basis to use, as follows:

Value	Day Count Basis
0 or omitted	US (NASD) 30/360
1	Actual/actual
2	Actual/360
3	Actual/365
4	European 30/360

Time information in the date arguments is ignored.

Return Type and Value: number – The next coupon date after the settlement date, as a date.

However, if

- *settlement* or *maturity* is out of range for the current date base value, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`COUPNCD(DATE(2007,1,25),DATE(2008,11,15),2,1)` results in `15-May-2007`

end example]

3.17.7.61 COUPNUM

`COUPNUM ( settlement , maturity , frequency [ , [ basis ] ] )`

Description: Computes the number of coupons payable between the settlement date and maturity date, rounded up to the nearest whole coupon.

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
frequency	number	The number of coupon payments per year. For annual payments, frequency is 1; for semiannual payments, frequency is 2; for quarterly payments, frequency is 4. *frequency* is truncated to an integer.
basis	number	The truncated integer type of day count basis to use, as follows:

Value	Day Count Basis
0 or omitted	US (NASD) 30/360
1	Actual/actual
2	Actual/360
3	Actual/365
4	European 30/360

Time information in the date arguments is ignored.

Return Type and Value: number – The number of coupons payable between the settlement date and maturity date, rounded up to the nearest whole coupon.

However, if

- *settlement* or *maturity* is out of range for the current date base value, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`COUPNUM(DATE(2007,1,25),DATE(2008,11,15),2,1)` results in 4

end example]

3.17.7.62 COUPPCD

COUPPCD (*settlement* , *maturity* , *frequency* [, [*basis*]])

Description: Computes the previous coupon date before the settlement date.

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
frequency	number	The number of coupon payments per year. For annual payments, frequency is 1; for semiannual payments, frequency is 2; for quarterly payments, frequency is 4. *frequency* is truncated to an integer.
basis	number	The truncated integer type of day count basis to use, as follows: 0 or omitted – US (NASD) 30/360 1 – Actual/actual 2 – Actual/360 3 – Actual/365 4 – European 30/360

Time information in the date arguments is ignored.

Return Type and Value: number – The previous coupon date before the settlement date, as a date.

However, if

- *settlement* or *maturity* is out of range for the current date base value, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`COUPPCD(DATE(2007,1,25),DATE(2008,11,15),2,1)` results in `15-Nov-2006`

end example]

3.17.7.63 COVAR

Syntax:

COVAR (*array-1* , *array-2*)

Description: Computes covariance; that is, the average of the products of deviations for each data point pair in the two cell ranges designated by *array-1* and *array-2*.

Mathematical Formula:

The covariance is:

$$Cov(X,Y) = \frac{\sum (x-\bar{x})(y-\bar{y})}{n}$$

where x and y are the sample means AVERAGE(*array-1*) and AVERAGE(*array-2*), and n is the sample size.

Arguments:

Name	Type	Description
array-1	number, name, array, reference to number	If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.
array-2		

Return Type and Value: number – The covariance.

However, if

- *array-1* and *array-2* have a different number of data points, the return value is unspecified.
- *array-1* or *array-2* is empty, the return value is unspecified.

[*Example*:

`COVAR({2.532,5.621;2.1,3.4},{5.32,2.765;5.2,6.7})` results in `-1.375374`

end example]

3.17.7.64 CRITBINOM

Syntax:

`CRITBINOM ( number-trials , success-probability , alpha )`

Description: Computes the smallest value for which the cumulative binomial distribution is greater than or equal to a criterion value.

Arguments:

Name	Type	Description
number-trials	number	The number of Bernoulli trials.
success-	number	The probability of success on each trial.

Name	Type	Description
probability		
alpha	number	The criterion value.

Return Type and Value: number – The smallest value for which the cumulative binomial distribution is greater than or equal to a criterion value.

However, if

- *number-trials* < 0, `#NUM!` is returned.
- *success-probability* is < 0 or *success-probability* > 1, `#NUM!` is returned.
- *alpha* < 0 or *alpha* > 1, `#NUM!` is returned.

[*Example*:

`CRITBINOM(6,0.5,0.75)` results in 4
`CRITBINOM(12,0.3,0.95)` results in 6

end example]

3.17.7.65 CUBEKPIMEMBER

Syntax:

`CUBEKPIMEMBER ( connection , kpi-name , kpi-property [ , [ caption ] ] )`

Description: Fetches from the OLAP cube on the SQL Server designated by *connection*, a Key Performance Indicator (KPI) name, property, and measure, and displays the name and property in the cell. A KPI is a quantifiable measurement, such as monthly gross profit or quarterly employee turnover, used to monitor an organization's performance.

Arguments:

Name	Type	Description
connection	text	The name of the connection to the cube.
kpi-name	text	The name of the KPI in the cube.
kpi-property	number	The KPI component to be returned, truncated to integer; it shall be one of the following: 1 – The actual value 2 – A target value 3 – The state of the KPI at a specific

Name	Type	Description
		<table><tr><td></td><td>moment in time</td></tr><tr><td>4</td><td>A measure of the value over time</td></tr><tr><td>5</td><td>A relative importance assigned to the KPI</td></tr><tr><td>6</td><td>A temporal context for the KPI</td></tr></table> If 1 is specified, only *kpi-name* is displayed in the cell.
caption	text	An alternative string whose value is displayed in the cell instead of *kpi-name* and *kpi-property*.

Return Type and Value: any – The selected key performance indicator.

However, if

- *kpi-name* is invalid, the return value is unspecified.
- *kpi-property* is outside the range 1–6, #N/A is returned.
- The *connection* name is not a valid workbook connection stored in the workbook, the return value is unspecified.
- The OLAP server is not running, not available, or returns an error message, the return value is unspecified.

[*Example*:

```
CUBEKPIMEMBER("Sales","MySalesKPI",1)
CUBEKPIMEMBER("Sales","MySalesKPI",2,"Sales KPI Goal")
```

end example]

3.17.7.66 CUBEMEMBER

Syntax:

CUBEMEMBER (*connection* , *member-expression* , [, [*caption*]])

Description: Fetches from the OLAP cube on the SQL Server designated by *connection*, the member or tuple defined by *member-expression*. [*Note*: This function is used to validate that the member or tuple exists in the cube. *end note*]

When a call to CUBEMEMBER is used as an argument to another CUBExxx function, the MDX expression that identifies the member or tuple is used by that CUBExxx function, rather than the displayed value in the cell of the CUBEMEMBER function.

Arguments:

Name	Type	Description
connection	text	The name of the connection to the cube.
member-expression	text, reference, array	A multidimensional expression (MDX) that evaluates to a unique member in the cube. Alternatively, *member-expression* can be a tuple, specified as a cell range or an array constant.
caption	text	The string displayed in the cell instead of the caption from the cube (assuming it defines such a caption). When a tuple is returned, the caption used is the one for the last member in the tuple.

Return Type and Value: any – A member or tuple in a cube hierarchy.

However, if

- The connection name is not a valid workbook connection stored in the workbook, the return value is unspecified.
- The OLAP server is not running, not available, or returns an error message, the return value is unspecified.
- At least one element within the tuple is invalid, the return value is unspecified.
- The syntax of *member-expression* is incorrect, the return value is unspecified.
- The member specified by *member-expression* doesn't exist in the cube, the return value is unspecified.
- The tuple is invalid because there is no intersection for the specified values, the return value is unspecified.
- The set contains at least one member with a different dimension than the other members, the return value is unspecified.

[*Example*:

```
CUBEMEMBER("Sales","[Time].[Fiscal].[2004]")
CUBEMEMBER($A$1,D$12)
CUBEMEMBER("Sales",(B4,C6,D5),"SalesFor2004")
CUBEMEMBER("Sales",{[Products].[Food];[Time].[Fiscal].[2004]})
CUBEMEMBER($A$1,C$12:D$12)
```

end example]

3.17.7.67 CUBEMEMBERPROPERTY

Syntax:

CUBEMEMBERPROPERTY (*connection* , *member-expression* , *property*)

Description: Fetches a property of a member in the OLAP cube on an SQL Server. [*Note*: Use this function o validate that a member name exists within the cube and to return the specified property for this member. *end note*]

Arguments:

Name	Type	Description
connection	text	The name of the connection to the cube.
member-expression	text	A multidimensional expression (MDX) that evaluates to a unique member in the cube.
property	text	The name of the property returned or a reference to a cell that contains the name of the property.

Return Type and Value: any – A property of a member in the OLAP cube.

However, if

- The connection name is not a valid workbook connection stored in the workbook, the return value is unspecified.
- The OLAP server is not running, not available, or returns an error message, the return value is unspecified.
- The syntax of *member-expression* is incorrect, the return value is unspecified.
- The member specified by *member-expression* doesn't exist in the cube, the return value is unspecified.

[*Example*:

```
CUBEMEMBERPROPERTY("Sales","[Time].[Fiscal].[2004]",$A$3)
CUBEMEMBERPROPERTY("Sales","[Store].[MyFavoriteStore]",
  "[Store].[Store Name].[Store Sqft]")
```

end example]

3.17.7.68 CUBERANKEDMEMBER

Syntax:

CUBERANKEDMEMBER (*connection* , *set-expression* , *rank* [, *caption*])

Description: Fetches the n^{th}, or ranked, member in a set.

Arguments:

Name	Type	Description
connection	text	The name of the connection to the cube.

Name	Type	Description
set-expression	text	A set expression, such as "{[Item1].children}".
rank	number	Specifies the top value to return, truncated to integer. If 1, the top value is returned; if 2, the second-most top value is returned; and so on.
caption	text	The text displayed in the cell instead of the caption from the cube (assuming it defines such a caption).

Return Type and Value: any – The n^{th} member in the set.

However, if

- The connection name is not a valid workbook connection stored in the workbook, the return value is unspecified.
- The OLAP server is not running, not available, or returns an error message, the return value is unspecified. , the return value is unspecified.
- The syntax of *member-expression* is incorrect, the return value is unspecified.
- The set contains at least one member with a different dimension than the other members, the return value is unspecified.

[*Example*:

```
CUBERANKEDMEMBER("Sales",$D$4,1,"Top Month")
CUBERANKEDMEMBER("Sales",CUBESET("Sales","Summer","[2004].[June]",
  "[2004].[July]","[2004].[August]"),3,"Top Month")
```

end example]

3.17.7.69 CUBESET

Syntax:

```
CUBESET ( connection , set-expression [ , [ caption ][ , [ sort-order ]
    [ , [ sort-by ]]]] )
```

Description: Fetches from the OLAP cube on the SQL Server designated by *connection* the set of members or tuples that is defined by *set-expression*. [*Note*: Use this function to build dynamic reports that aggregate and filter data, by using the return value as a slicer in the CUBEVALUE function, the CUBERANKEDMEMBER function to choose specific members from the calculated set, and the CUBESETCOUNT function to control the size of the set. *end note*]

Arguments:

Name	Type	Description

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>connection</td><td>text</td><td>The name of the connection to the cube.</td></tr>
<tr><td>set-expression</td><td>text, reference</td><td>A set expression that results in a set of members or tuples. set-expression can also be a cell reference to range that contains one or more members, tuples, or sets included in the set.</td></tr>
<tr><td>caption</td><td>text</td><td>The text displayed in the cell instead of the caption from the cube (assuming it defines such a caption).</td></tr>
<tr><td>sort-order</td><td>text</td><td>The type of sort, if any, to perform; it can be one of the following:
<table>
<tr><th>Value</th><th>Description</th><th>sort-by argument</th></tr>
<tr><td>0 or default</td><td>Leaves the set in existing order</td><td>Ignored</td></tr>
<tr><td>1</td><td>Sorts set in ascending order by sort_by</td><td>Required</td></tr>
<tr><td>2</td><td>Sorts set in descending order by sort_by</td><td>Required</td></tr>
<tr><td>3</td><td>Sorts set in alphabetic ascending order</td><td>Ignored</td></tr>
<tr><td>4</td><td>Sorts set in alphabetic descending order</td><td>Ignored</td></tr>
<tr><td>5</td><td>Sorts set in natural ascending order</td><td>Ignored</td></tr>
<tr><td>6</td><td>Sorts set in natural descending order</td><td>Ignored</td></tr>
</table>
An alphabetic sort for a set of tuples sorts on the last element in each tuple.</td></tr>
<tr><td>sort-by</td><td>text</td><td>The value by which to sort. [Example: To get the city with the highest sales, set-expression would be a set of cities, and sort-by would be the sales measure. To get the city with the highest population, set-expression would be a set of cities, and sort-by would be the population measure. end example]</td></tr>
</table>

Return Type and Value: any – The set of members or tuples.

However, if

- The connection name is not a valid workbook connection stored in the workbook, the return value is unspecified.
- The OLAP server is not running, not available, or returns an error message, the return value is unspecified.
- The syntax of *member-expression* is incorrect, the return value is unspecified.
- The set contains at least one member with a different dimension than the other members, the return value is unspecified.
- *sort-order* is outside the range 0–6, #N/A is returned.
- *sort-order* requires *sort-by*, but *sort-by* is omitted, #VALUE! is returned.

[*Example*:

```
CUBESET("Finance","Order([Product].[Product].[Product Category]
.Members,[Measures].[Unit Sales],ASC)","Products")

CUBESET("Sales","[Product].[All Products].Children",
  "Products",1,"[Measures].[Sales Amount]")
```

end example]

3.17.7.70 CUBESETCOUNT

Syntax:

CUBESETCOUNT (*set*)

Description: Computes the number of items in a set.

Arguments:

Name	Type	Description
set	text	An expression that evaluates to a set defined by the CUBESET function.

Return Type and Value: number – The number of items in a set.

[*Example*:

```
CUBESETCOUNT(A3)
CUBESETCOUNT(CUBESET("Sales","[Product].[All Products].Children",
  "Products",1,"[Measures].[Sales Amount]"))
```

end example]

3.17.7.71 CUBEVALUE

Syntax:

CUBEVALUE (*connection* , *argument-list*)

Description: Fetches from the OLAP cube on the SQL Server designated by *connection*, the aggregated value defined by a series of member-expression *argument*s in *argument-list*.

Arguments:

Name	Type	Description
connection	text	The name of the connection to the cube.
argument-list	text, reference	Each *argument* in *argument-list* is text containing a multidimensional expression (MDX) that evaluates to a member or tuple within the cube. Alternatively, an *argument* can be a set defined with the CUBESET function. Use any *argument* as a slicer to define the portion of the cube for which the aggregated value is returned. If no measure is specified in an *argument*, the default measure for that cube is used. If a cell reference is used for an *argument*, and that cell reference contains a CUBE function, then that *argument* uses the MDX expression for the item in the referenced cell, and not the value displayed in that referenced cell.

Return Type and Value: any – The aggregated value.

However, if

- The connection name is not a valid workbook connection stored in the workbook, the return value is unspecified.
- The OLAP server is not running, not available, or returns an error message, the return value is unspecified.
- At least one element within the tuple is invalid, the return value is unspecified.
- The syntax of *member-expression* is incorrect, the return value is unspecified.
- The member specified by an *argument* doesn't exist in the cube, the return value is unspecified.
- The tuple is invalid because there is no intersection for the specified values, the return value is unspecified. (This can occur with multiple elements from the same hierarchy.)
- The set contains at least one member with a different dimension than the other members, the return value is unspecified.

[*Example*:

```
CUBEVALUE("Sales","[Measures].[Profit]","[Time].[2004]",
```

```
  "[All Product].[Beverages]")
CUBEVALUE($A$1,"[Measures].[Profit]",D$12,$A23)
CUBEVALUE("Sales",$B$7,D$12,$A23)
```

end example]

3.17.7.72 CUMIPMT

Syntax:

`CUMIPMT ( rate , nper , pv , start-period , end-period , type )`

Description: Computes the cumulative interest paid on a loan between *start-period* and *end-period*.

Arguments:

Name	Type	Description
rate	number	The interest rate.
nper	number	The total number of payment periods, truncated to integer.
pv	number	The present value.
start-period	number	The first period in the calculation. (Payment periods are numbered beginning with 1.)
end-period	number	The last period in the calculation.
type	number	The timing of the payment, truncated to integer, as follows:

Value	Timing
0	Payment at the end of the period
1	Payment at the beginning of the period

Time information in the date arguments is ignored.

Return Type and Value: number – The cumulative interest paid on a loan.

However, if

- *rate*, *nper*, or *pv* ≤ 0, #NUM! is returned.
- *start-period* < 1 or *end-period* < 1, or *start-period* > *end_period*, #NUM! is returned.
- *type* is any number other than 0 or 1, #NUM! is returned.

[*Example*:

`CUMIPMT(0.09/12,30*12,125000,13,24,0)` results in `-11135.23`
`CUMIPMT(0.09/12,30*12,125000,1,1,0)` results in `-937.50`

end example]

3.17.7.73 CUMPRINC

Syntax:

`CUMPRINC` (*rate* , *nper* , *pv* , *start-period* , *end-period* , *type*)

Description: Computes the cumulative principal paid on a loan between *start-period* and *end-period*.

Arguments:

Name	Type	Description
rate	number	The interest rate.
nper	number	The total number of payment periods, truncated to integer.
pv	number	The present value.
start-period	number	The first period in the calculation. (Payment periods are numbered beginning with 1.)
end-period	number	The last period in the calculation.
type	number	The timing of the payment, truncated to integer, as follows:

Value	Timing
`0`	Payment at the end of the period
`1`	Payment at the beginning of the period

Time information in the date arguments is ignored.

Return Type and Value: number – The cumulative principal paid on a loan.

However, if

- *rate*, *nper*, or *pv* ≤ 0, `#NUM!` is returned.
- *start-period* < 1 or *end-period* < 1, or *start-period* > *end_period*, `#NUM!` is returned.
- *type* is any number other than 0 or 1, `#NUM!` is returned.

[*Example*:

`CUMPRINC(0.09/12,30*12,125000,13,24,0)` results in `-934.11`
`CUMPRINC(0.09/12,30*12,125000,1,1,0)` results in `-68.28`

end example]

3.17.7.74 DATE

Syntax:

`DATE ( year , month , day )`

Description: Computes the serial value for the given date.

Arguments:

Name	Type	Description
year	number	A year, truncated to integer, that together with *month* and *day* specifies the date whose serial value is to be computed. For the 1900 date base system: • If *year* is in the range 0–1899, inclusive, the year shall be interpreted as *year* + 1900. • If *year* is in the range 1900–9999, inclusive, the year shall be interpreted as *year*. For the 1904 date base system: • If *year* is in the range 4–1899, inclusive, the year shall be interpreted as *year* + 1900. • If *year* is in the range 1904–9999, inclusive, the year shall be interpreted as *year*.
month	number	A month, truncated to integer, that together with *year* and *day* specifies the date whose serial value is to be computed. If *month* is in the range 1–12, the month shall be interpreted as *month*. If *month* is less than 1 or greater than 12, the month shall be interpreted as the normalized value (see below) of *month*, and the year shall be adjusted accordingly.
day	number	A day, truncated to integer, that together with *month* and *year* specifies the date whose serial value is to be computed. If *day* is in the allowable range of days for the month, the day shall be interpreted as *day*. If *day* is less than 1 or greater than the number of days in the given month, the day shall be interpreted as the normalized value (see below) of *day*, and the year and month shall be adjusted

Name	Type	Description
		accordingly.

Month and/or day normalization occurs when *month* and/or *day* is outside the range 1–12 and 1–number-of-days-in-the-given-month, respectively. Specifically, if *month* is greater than 12, that number of months shall be added to the first month in the year specified. If *month* is less than 1, the magnitude of that number of months, plus 1, is subtracted from the first month in the year specified. If *day* is greater than the number of days in the month specified, that number of days is added to the first day in the month specified. If *day* is less than 1, the magnitude of that number of months, plus 1, is subtracted from the first day in the month specified. If both *month* and *day* in the same date are normalized, *month* is normalized first.

Return Type and Value: number – The serial value for the given date.

However, if

- *year* is less than 0 or is greater-than or equal-to 10000, and the 1900 date base system is being used, `#NUM!` is returned.
- *year* is less than 4, is greater-than or equal-to 10000, is in the range 1900–1903, inclusive, and the 1904 date base system is being used, `#NUM!` is returned.

[*Example*: For the 1900 date base system:

`DATE(0,1,1)` results in a serial value of `1`
`DATE(1899,1,1)` results in a serial value of `693598`
`DATE(1900,1,1)` results in a serial value of `1`
`DATE(9999,12,31)` results in a serial value of `2958465`

For the 1904 date base system:

`DATE(4,1,1)` results in a serial value of `0`
`DATE(1899,1,1)` results in a serial value of `692136`
`DATE(1904,1,1)` results in a serial value of `0`
`DATE(9999,12,31)` results in a serial value of `2957003`

end example]

3.17.7.75 DATEDIF

Syntax:

`DATEDIF ( start-date , end-date , unit )`

Description: Calculates the number of days, months, or years between two dates.

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>start-date</td><td>number</td><td>The first date in the period, truncated to integer.</td></tr>
<tr><td>end-date</td><td>number</td><td>The last date in the period, truncated to integer.</td></tr>
<tr><td>unit</td><td>text</td><td>The count to be returned, as follows:
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td>"Y"</td><td>The number of complete years in the period.</td></tr>
<tr><td>"M"</td><td>The number of complete months in the period.</td></tr>
<tr><td>"D"</td><td>The number of days in the period.</td></tr>
<tr><td>"MD"</td><td>The difference between the days in start-date and end-date. The months and years of the dates are ignored.</td></tr>
<tr><td>"YM"</td><td>The difference between the months in start-date and end-date. The days and years of the dates are ignored.</td></tr>
<tr><td>"YD"</td><td>The difference between the days of start-date and end-date. The years of the dates are ignored.</td></tr>
</table></td></tr>
</table>

Return Type and Value: number – The number of days, months, or years between two dates, depending on the value of *unit*.

However, if

- *start-date* or *end-date* is out of range for the current date base value, #NUM! is returned.
- *start-date* ≥ *end-date* #NUM! is returned.
- *unit* is any value other than those shown in the table above, #NUM! is returned.

[*Example*:

DATEDIF(DATE(2001,1,1),DATE(2003,1,1),"Y") results in 2 complete years
DATEDIF(DATE(2001,6,1),DATE(2002,8,15),"D") results in 440 days
DATEDIF(DATE(2001,6,1),DATE(2002,8,15),"YD") results in 75 days
DATEDIF(DATE(2001,6,1),DATE(2002,8,15),"MD") results in 14 days

end example]

3.17.7.76 DATEVALUE

Syntax:

DATEVALUE (*date-time-string*)

Description: Computes the serial value of the date and/or time represented by the string *date-time-string*, taking into account the current date base value.

Arguments:

Name	Type	Description
date-time-string	text	The date and/or time whose serial value is to be computed. *date-time-string* can have any valid date and/or time format. If the year portion of *date-time-string* is omitted, the current year is used. Any time information in *date-time-string* shall be ignored.

Return Type and Value: number – The serial value of the date and/or time represented by the string *date-time-string*.

However, if

- *date-time-string* is out of range for the current date base value, #VALUE! is returned.
- *date-time-string* does not represent a date, #VALUE! is returned.

[*Example*: When the current year is 2006,

```
DATEVALUE("2/1/2006")
DATEVALUE("01-Feb-2006 10:06 AM")
DATEVALUE("2006/2/1")
DATEVALUE("2006-2-1")
DATEVALUE("1-Feb")
```

all result in 38749 for the 1900 date base system, or 37287 for the 1904 date base system. *end example*]

3.17.7.77 DAVERAGE

Syntax:

DAVERAGE (*database* , *field* , *criteria*)

Description: Averages the values in a column of a list or database that match the specified criteria.

[*Note*: In order to perform an operation on an entire column in a database, a blank line must be entered below the column labels in the criteria range. *end note*]

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database, which shall be a list of related data in which rows of related information are records, and columns of data are fields. The first row of the list shall contain labels for each column.
field	text, number	Indicates the column to which *criteria* shall be applied. It can either be a string containing the column's label, or the column's position number, where columns are numbered starting at 1. [*Example*: If column 3's label is "Age" then either 3 or "Age" can be used. *end example*]
criteria	reference	The range of cells that contains the specified conditions. Each cell in that range that contains a condition shall have a value that is the form of a number, an expression, a cell reference, or text that defines which cells will be selected. In the case of text, a condition can consist of any comparison operator followed by the operand against which each cell's value is to be compared. If the text form is used and the text does not begin with a comparison operator, the criteria matches any string starting with that text. [*Example*: A criteria of "Pea" can result in Pea, Pear, and Peach's being matched, whereas a criteria of "=Pea" will only match Pea. *end example*] *criteria* can include one or more wildcard characters, question mark (`?`) and asterisk (*). A question mark matches any single character; an asterisk matches any sequence of characters. To search for a question mark, asterisk, or tilde character, prefix that character with a tilde (~). The range shall include at least one column label and at least one cell below the column label in which a condition for the column is specified. [*Example*: If the range `G1:G2` contains the column label `Income` in `G1` and the amount 10,000 in `G2`, one could define the range as `MatchIncome` and use that name as *criteria*. *end example*] The value of *criteria* shall not overlap the range specified by *database*. To find rows that meet multiple criteria for a single column, all of the criteria shall be specified directly below one another in separate rows of the *criteria* range. To find rows that meet multiple criteria for multiple columns, all of the criteria shall be specified in the same row of the *criteria* range. To find rows that meet multiple criteria for multiple columns, where any criteria can be true, each of the criteria shall be specified in a different row of the *criteria*

Name	Type	Description
		range. To find rows that meet multiple sets of criteria, where each set includes criteria for multiple columns, each set of criteria shall be specified in a separate row of the *criteria* range. To find rows that meet multiple sets of criteria, where each set includes criteria for one column, multiple columns with the same column heading shall be included in the *criteria* range.

Return Type and Value: number – The average of the values of the cells that correspond to the specified criteria.

[*Example*: Given the following data:

	A	B	C	D	E	F
1	Tree	Height	Age	Yield	Profit	Height
2	=Apple	>10				<16
3	=Pear					
4	Tree	Height	Age	Yield	Profit	
5	Apple	18	20	14	105.00	
6	Pear	12	12	10	96.00	
7	Cherry	13	14	9	105.00	
8	Apple	14	15	10	75.00	
9	Pear	9	8	8	76.80	
10	Apple	8	9	6	45.00	

the average yield of apple trees over 10 feet in height is computed by `DAVERAGE(A4:E10,"Yield",A1:B2)`, which results in 12

The average age of all trees is computed by `DAVERAGE(A4:E10,3,A4:E10)`, which results in 13

end example]

3.17.7.78 DAY

Syntax:

DAY (*date-value*)

Description: Computes the numeric Gregorian day for the date and/or time having the given *date-value*, taking into account the current date base value.

Arguments:

Name	Type	Description
date-value	number, text	The date and/or time whose day is to be computed. That date and/or time shall be expressed either as a serial value, in which case, its fractional part is ignored, or as a *string-constant* having any valid date and/or time format, in which case, any time information shall be ignored.

Return Type and Value: number – The Gregorian day for the date and/or time having the given *date-value*. The returned value shall be in the range 1–31.

However, if *date-value* is out of range for the current date base value, `#NUM!` is returned.

[*Example*:

`DAY(DATE(2006,1,2))` results in 2
`DAY(DATE(2006,0,2))` results in 31
`DAY("2006/1/2 10:45 AM")` results in 2
`DAY(30000)` results in 18 for the 1900 date base system, or 19 for the 1904 date base system

end example]

3.17.7.79 DAYS360

Syntax:

`DAYS360 ( start-date , end-date [ , method-flag ] )`

Description: Computes the signed number of days between two dates based on a 360-day year (twelve 30-day months).

Arguments:

Name	Type	Description
start-date	number	*start-date* and *end-date* are the dates for which the difference is to be computed. *start-date* can be earlier than, the same as, or later than *end-date*.
start-date	number	
method-flag	logical	Specifies whether to use the U.S. or European method in the calculation, as follows:

Value	Meaning

Name	Type	Description	
		FALSE or omitted	U.S. (NASD) method: If the *start-date* is the 31st day of a month, it is changed to the 30th day of that same month. If the *end-date* is the 31st day of a month and the *start-date* is earlier than the 30th day of a month, the *end-date* is changed to the 1st day of the following month; otherwise the *end-date* is changed to the 30th day of the same month.
		TRUE	European method: *start-date*s and *end-date*s that occur on the 31st day of a month are changed to the 30th day of the same month.

Return Type and Value: number – The signed number of days between two dates based on a 360-day year (12 30-day months). If *start-date* is later than *end-date*, the return value shall be negative, and the magnitude shall be the difference in days.

However, if *start-date* or *end-date* is out of range for the current date base value, `#NUM!` is returned.

[*Example*:

`DAYS360(DATE(2002,2,3),DATE(2005,5,31))` results in `1198`
`DAYS360(DATE(2005,5,31),DATE(2002,2,3))` results in `-1197`
`DAYS360(DATE(2002,2,3),DATE(2005,5,31),FALSE)` results in `1198`
`DAYS360(DATE(2002,2,3),DATE(2005,5,31),TRUE)` results in `1197`

|*end example*]

3.17.7.80 DB

Syntax:

`DB ( cost , salvage , life , period [ , [ month ] ] )`

Description: Computes the depreciation of an asset for a specified period using the fixed-declining balance method.

Mathematical Formula:

The fixed-declining balance method computes depreciation at a fixed rate. DB uses the following formulas to calculate depreciation for a period:

(cost - total depreciation from prior periods) * rate

where:

rate = 1 - ((salvage / cost) ^ (1 / life)), rounded to three decimal places

Depreciation for the first and last periods is a special case. For the first period, DB uses this formula:

cost * rate * month / 12

For the last period, DB uses this formula:

((cost - total depreciation from prior periods) * rate * (12 - month)) / 12

Arguments:

Name	Type	Description
cost	number	The initial cost of the asset.
salvage	number	The value at the end of the depreciation. (This is sometimes called the salvage value of the asset.)
life	number	The number of periods over which the asset is being depreciated. (This is sometimes called the useful life of the asset.)
period	number	The period for which the depreciation is to be calculated. (*period* shall use the same units as *life*.)
month	number	The number of months in the first year. If omitted, a value of 12 is used.

Return Type and Value: number – The depreciation of an asset for a specified period using the fixed-declining balance method.

However, if

- *cost*, *salvage*, *life*, or *period* < 0, `#NUM!` is returned.
- *month* is outside the range 1–12, `#NUM!` is returned.

[*Example*:

`DB(1000000,100000,6,1,7)` results in `186,083.33`
`DB(1000000,100000,6,2,7)` results in `259,639.42`
`DB(1000000,100000,6,3,7)` results in `176,814.44`
`DB(1000000,100000,6,4,7)` results in `120,410.64`
`DB(1000000,100000,6,5,7)` results in `81,999.64`
`DB(1000000,100000,6,6,7)` results in `55,841.76`
`DB(1000000,100000,6,7,7)` results in `15,845.10`

end example]

3.17.7.81 DCOUNT

Syntax:

DCOUNT (*database* , *field* , *criteria*)

Description: Counts the number of values in a column of a list or database that match the specified criteria. (See the DAVERAGE function §3.17.7.77.)

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database.
field	text, number	The column to which *criteria* shall be applied.
criteria	reference	The range of cells that contains the specified conditions.

For a detailed description of each argument, see the DAVERAGE function §3.17.7.77.

Return Type and Value: number – The count of the values of the cells that correspond to the specified criteria.

[*Example*: Using the data in the example in the DAVERAGE function §3.17.7.77:

For all the apple trees having a height between 10 and 16, the number of Age fields that contain numbers is computed by `DCOUNT(A4:E10,"Age",A1:F2)`, which results in 1.

end example]

3.17.7.82 DCOUNTA

Syntax:

DCOUNTA (*database* , *field* , *criteria*)

Description: Counts the number of non-blank cells in a column of a list or database that match the specified criteria. (See the DAVERAGE function §3.17.7.77.)

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database.
field	text, number	The column to which *criteria* shall be applied.
criteria	reference	The range of cells that contains the specified conditions.

For a detailed description of each argument, see the DAVERAGE function §3.17.7.77.)

Return Type and Value: number – The count of the non-blank cells that correspond to the specified criteria.

[*Example*: Using the data in the example in the DAVERAGE function §3.17.7.77:

For all the apple trees having a height between 10 and 16, the number of Profit fields that are not blank is computed by `DCOUNTA(A4:E10,"Profit",A1:F2)`, which results in `1`.

end example]

3.17.7.83 DDB

Syntax:

`DDB` (*cost* , *salvage* , *life* , *period* [, *factor*]])

Description: Computes the depreciation of an asset for a specified period using the double-declining balance or some other specified method. [*Note*: Use VDB (§3.17.7.338) for a straight-line depreciation method when depreciation is greater than the declining balance calculation. *end note*]

Mathematical Formula:

`MIN`((cost - total depreciation from prior periods) * (factor/life), (cost - salvage - total depreciation from prior periods))

Arguments:

Name	Type	Description
cost	number	The initial cost of the asset.
salvage	number	The value at the end of the depreciation. (This is sometimes called the salvage value of the asset.)
life	number	The number of periods over which the asset is being depreciated. (This is sometimes called the useful life of the asset.)
period	number	The period for which the depreciation is to be calculated. (*period* shall use the same units as *life*.)
factor	number	The rate at which the balance declines. If omitted, it is assumed to be 2 (the double-declining balance method).

Return Type and Value: number – The depreciation of an asset for a specified period.

However, if

- salvage < 0 #NUM! is returned.

- cost life <= 0, #NUM! is returned.
- life <= 0 #NUM! is returned.
- period <= 0, #NUM! is returned.
- factor <= 0, #NUM! is returned.

[*Example*:

`DDB(2400,300,10*365,1)` results in `1.32`
`DDB(2400,300,10*12,1,2)` results in `40.00`
`DDB(2400,300,10,1,2)` results in `480.00`
`DDB(2400,300,10,2,1.5)` results in `306.00`
`DDB(2400,300,10,10)` results in `22.12`

end example]

3.17.7.84 DEC2BIN

Syntax:

`DEC2BIN ( number [ , num-bin-digits ] )`

Description: Makes the binary equivalent of *number*, with the result having *num-bin-digits* digits.

Arguments:

Name	Type	Description
number	number	The decimal number that is to be converted to a binary string.
num-bin-digits	number	The number of digits in the result, with leading zeros added as necessary. However, if *number* is negative, *num-bin-digits* is ignored and the result has 10 digits. If omitted, the minimum number of digits is used in the result. *num-bin-digits* is truncated to an integer.

Return Type and Value: text – The binary equivalent of *number* using twos-complement representation with the left-most bit (10th bit from the right) representing the sign bit.

However, if

- *number* is outside the range `-512` (1000000000 binary) to `511` (0111111111 binary), inclusive, `#NUM!` is returned.
- *number* needs more digits that *num-bin-digits*, `#NUM!` is returned.
- *num-bin-digits* ≤ 0 or > 10, `#NUM!` is returned.

[*Example*:

DEC2BIN(23) results in 10111
DEC2BIN(-256) results in 1100000000
DEC2BIN(18,7) results in 0010010

end example]

3.17.7.85 DEC2HEX

Syntax:

DEC2HEX (*number* [, *num-hex-digits*])

Description: Makes the hexadecimal equivalent of *number*, with the result having *num-hex-digits* digits.

Arguments:

Name	Type	Description
number	number	The decimal number that is to be converted to a hexadecimal string.
num-bin-digits	number	*num-hex-digits* is the number of digits in the result, with leading zeros added as necessary. However, if *number* is negative, *num-hex-digits* is ignored and the result has 10 digits. If *num-hex-digits* is omitted, the minimum number of digits is used in the result. *num-hex-digits* is truncated to an integer.

Return Type and Value: text – The hexadecimal equivalent of *number* using twos-complement representation with the left-most bit (40th bit from the right) representing the sign bit.

However, if

- *number* is outside the range -549,755,813,888 (8000000000 hex) to 549,755,813,887 (7FFFFFFFFF hex), inclusive, #NUM! is returned.
- *number* needs more digits that *num-hex-digits*, #NUM! is returned.
- *num-hex-digits* ≤ 0 or > 10, #NUM! is returned.

[*Example*:

DEC2HEX(23) results in 17
DEC2HEX(-256) results in FFFFFFFF00
DEC2HEX(18,7) results in 0000012

end example]

3.17.7.86 DEC2OCT

Syntax:

DEC2OCT (*number* [, *num-oct-digits*])

Description: Makes the octal equivalent of *number*, with the result having *num-oct-digits* digits.

Arguments:

Name	Type	Description
number	number	The decimal number that is to be converted to an octal string.
num-bin-digits	number	The number of digits in the result, with leading zeros added as necessary. However, if *number* is negative, *num-oct-digits* is ignored and the result has 10 digits. If *num-oct-digits* is omitted, the minimum number of digits is used in the result. *num-oct-digits* is truncated to an integer.

Return Type and Value: text – The octal equivalent of *number* using twos-complement representation with the left-most bit (30th bit from the right) representing the sign bit.

However, if

- *number* is outside the range `-536,870,912` (4000000000 octal) to `536,870,911` (3777777777 octal), inclusive, `#NUM!` is returned.
- *number* needs more digits that *num-oct-digits*, `#NUM!` is returned.
- *num-oct-digits* ≤ 0 or > 10, `#NUM!` is returned.

[*Example*:

`DEC2OCT(23)` results in `27`
`DEC2OCT(-256)` results in `7777777400`
`DEC2OCT(18,7)` results in `0000022`

end example]

3.17.7.87 DEGREES

Syntax:

DEGREES (*angle*)

Description: Converts *angle* in radians into degrees.

Arguments:

Name	Type	Description
angle	number	The number of radians that is to be converted into degrees.

Return Type and Value: number – *angle* in degrees.

[*Example*:

`DEGREES(2 * PI())` results in `360`
`DEGREES(PI())` results in `180`
`DEGREES(PI()/2)` results in `90`
`DEGREES(8.5)` results in `487.0141259`

end example]

3.17.7.88 DELTA

Syntax:

DELTA (*number-1* [, *number-2*])

Description: Compares two numbers for equality.

Arguments:

Name	Type	Description
number-1	number	The numbers that are to be compared for equality. If *number-2* is omitted, it is assumed to be zero.
number-2	number	

Return Type and Value: number – 1 if *number-1* equals *number-2*; otherwise, 0.

[*Example*:

`DELTA(10.5,10.5)` results in `1`
`DELTA(10.5,10.6)` results in `0`
`DELTA(10.5)` results in `0`
`DELTA(0)` results in `1`

end example]

3.17.7.89 DEVSQ

Syntax:

DEVSQ (*argument-list*)

Description: Computes the sum of squares of deviations of data points from their sample mean.

Mathematical Formula:

$$\mathrm{DEVSQ} = \sum (x - \bar{x})^2$$

Arguments:

Name	Type	Description
argument-list	logical, number, name, array, or reference to a number. Argument list can be a single argument that is an array or a reference to an array.	The *argument*s in *argument-list* designate the values for which the sum of squared deviations is to be calculated. Logical values and text representations of numbers occurring directly in the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0value 0value 0 are included.

Return Type and Value: number – The sum of squares of deviations of data points from their sample mean.

[*Example*:

DEVSQ(5.6,8.2,9.2) results in 6.906666667
DEVSQ({5.6,8.2,9.2}) results in 6.906666667

end example]

3.17.7.90 DGET

Syntax:

DGET (*database* , *field* , *criteria*)

Description: Extracts a single value from a column of a list that matches the specified criteria. (See the DAVERAGE function §3.17.7.77.)

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database.
field	text, number	The column to which *criteria* shall be applied.
criteria	reference	The range of cells that contains the specified conditions.

For a detailed description of each argument, see the DAVERAGE function §3.17.7.77.

Return Type and Value: number – The value of the cell that corresponds to the specified criteria.

However, if

- No record matches the criteria, `#VALUE!` is returned.
- More than one record matches the criteria, `#NUM!` is returned.

[*Example*: Using the data in the example in the DAVERAGE function §3.17.7.77:

For all the apple trees having a height between 10 and 16, the number of Profit fields that are not blank is computed by `DGET(A4:E7,"Yield",A1:A2)`, which results in 14.

end example]

3.17.7.91 DISC

Syntax:

`DISC ( `*settlement*` , `*maturity*` , `*pr*` , `*redemption*` [ , [ `*basis*` ] ] )`

Description: Computes the discount rate for a security.

Mathematical Formula:

$$DISC = \frac{redemption - par}{par} \times \frac{B}{DSM}$$

where:

B = number of days in a year, depending on the year basis.
DSM = number of days between settlement and maturity.

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
pr	number	The security's price per $100 face value.
redemption	number	The security's redemption value per $100 face value.
basis	number	The truncated integer type of day count basis to use, as follows: **Value** / **Day Count Basis** `0` or omitted / US (NASD) 30/360

Name	Type	Description
		1 Actual/actual 2 Actual/360 3 Actual/365 4 European 30/360

Time information in the date arguments is ignored.

Return Type and Value: number – The discount rate for a security.

However, if

- *settlement* or *maturity* is out of range for the current date base value, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *pr* or *redemption* ≤ 0, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`DISC(DATE(2007,1,25),DATE(2007,6,15),97.975,100,1)` results in `5.2420%`

end example]

3.17.7.92 DMAX

Syntax:

`DMAX` (*database* , *field* , *criteria*)

Description: Computes the maximum value of the cells in a column of a list or database that match the specified criteria. (See the DAVERAGE function §3.17.7.77.)

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database.
field	text, number	The column to which *criteria* shall be applied.
criteria	reference	The range of cells that contains the specified conditions.

For a detailed description of each argument, see the DAVERAGE function §3.17.7.77.

Return Type and Value: number – The maximum of the values of the cells that correspond to the specified criteria.

[*Example*: Using the data in the example in the DAVERAGE function §3.17.7.77:

The maximum profit of apple and pear trees is computed by `DMAX(A4:E10,"Profit",A1:A3)`, which results in `105`.

end example]

3.17.7.93 DMIN

Syntax:

`DMIN ( database , field , criteria )`

Description: Computes the minimum value of the cells in a column of a list or database that match the specified criteria. (See the DAVERAGE function §3.17.7.77.)

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database.
field	text, number	The column to which *criteria* shall be applied.
criteria	reference	The range of cells that contains the specified conditions.

For a detailed description of each argument, see the DAVERAGE function §3.17.7.77.

Return Type and Value: number – The minimum of the values of the cells that correspond to the specified criteria.

[*Example*: Using the data in the example in the DAVERAGE function §3.17.7.77:

The minimum profit of apple trees over 10 in height is computed by `DMIN(A4:E10,"Profit",A1:B2)`, which results in `75`.

end example]

3.17.7.94 DOLLAR

Syntax:

`DOLLAR ( number [ , num-decimal ] )`

Description: Produces a string containing *number* rounded to *num-decimal* decimal places. The thousands separator, radix point, and currency symbol are locale-specific. The format used is $#,##0.00;($#,##0.00).

Arguments:

Name	Type	Description
number	number	The number that is to be formatted.
num-decimal	number	Designate the number of decimal places to be used in the resulting string; it is truncated to an integer. If *num-decimal* is negative, *number* is rounded to the left of the decimal point. If omitted, a value of 2 shall be assumed.

Return Type and Value: text – The string containing *number* rounded to *num-decimal* decimal places, and have a currency symbol and thousands separators.

[*Example*: In a US-English context:

`DOLLAR(1234.567)` results in `$1,234.57`
`DOLLAR(1234.567,-2)` results in `$1,200`
`DOLLAR(-1234.567,4)` results in `($1,234.5670)`

In a France-French context:

`DOLLAR(1234.567)` results in `1 234,57 €`
`DOLLAR(1234.567,-2)` results in `1 200 €`
`DOLLAR(-1234.567,4)` results in `-1 234,5670 €`

In a Swiss-French context:

`DOLLAR(1234.567)` results in `SFr. 1'234.57`
`DOLLAR(1234.567,-2)` results in `SFr. 1'200`
`DOLLAR(-1234.567,4)` results in `SFr. -1'234.5670`

In a Norway-Norwegian (Nynorsk) context:

`DOLLAR(1234.567)` results in `kr 1 234,57`
`DOLLAR(1234.567,-2)` results in `kr 1 200`
`DOLLAR(-1234.567,4)` results in `kr -1 234,5670`
end example]

3.17.7.95 DOLLARDE

Syntax:

DOLLARDE (*fractional-dollar* , *fraction*)

Description: Converts a dollar price expressed as a fraction into a dollar price expressed as a decimal number. [*Note*: This function is used to convert fractional dollar numbers, such as securities prices, to decimal numbers. *end note*]

Arguments:

Name	Type	Description
fractional-dollar	number	The number expressed as a fraction.
fraction	number	The integer to use in the denominator of the fraction. [*Example*: A *fractional-dollar* value of *m.n* means *m* + *n*/*fraction* dollars. *end example*]

Return Type and Value: number – The dollar price expressed as a decimal number.

However, if

- *fraction* < 0, `#NUM!` is returned.
- *fraction* = 0, `#DIV/0!` is returned.

[*Example*:

`DOLLARDE(1.02,16)` results in `1.125`
`DOLLARDE(1.1,32)` results in `1.3125`

end example]

3.17.7.96 DOLLARFR

Syntax:

`DOLLARFR` (*decimal-dollar* , *fraction*)

Description: Converts a dollar price expressed as a decimal into a dollar price expressed as a fraction. [*Note*: This function is used to convert decimal dollar numbers, such as securities prices, to fractional numbers. *end note*]

Arguments:

Name	Type	Description
decimal-dollar	number	The number expressed as a decimal.
fraction	number	The integer to use in the denominator of the fraction.

Return Type and Value: number – The dollar price expressed as a fractional number. [*Example*: A result of *m.n* means *m* + *n*/*fraction* dollars. *end example*]

However, if

- *fraction* < 0, `#NUM!` is returned.

- *fraction* = 0, `#DIV/0!` is returned.

[*Example*:

`DOLLARFR(1.125,16)` results in `1.02`
`DOLLARFR(1.125,32)` results in `1.04`

end example]

3.17.7.97 DPRODUCT

Syntax:

DPRODUCT (*database* , *field* , *criteria*)

Description: Computes the product of the values of the cells in a column of a list or database that match the specified criteria. (See the DAVERAGE function §3.17.7.77.)

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database.
field	text, number	The column to which *criteria* shall be applied.
criteria	reference	The range of cells that contains the specified conditions.

For a detailed description of each argument, see the DAVERAGE function §3.17.7.77.

Return Type and Value: number – The product of the values of the cells that correspond to the specified criteria.

[*Example*: Using the data in the example in the DAVERAGE function §3.17.7.77:

The product of the yields from apple trees with a height greater than 10 is computed by `DPRODUCT(A4:E10,"Yield",A1:B2)`, which results in `140`.

end example]

3.17.7.98 DSTDEV

Syntax:

DSTDEV (*database* , *field* , *criteria*)

Description: Estimates the standard deviation of a population based on a sample by using the numbers in a column of a list or database that match the specified criteria. (See the DAVERAGE function §3.17.7.77.)

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database.
field	text, number	The column to which *criteria* shall be applied.
criteria	reference	The range of cells that contains the specified conditions.

For a detailed description of each argument, see the DAVERAGE function §3.17.7.77.

Return Type and Value: number – An estimate of the standard deviation of a population based on the cells that correspond to the specified criteria.

[*Example*: Using the data in the example in the DAVERAGE function §3.17.7.77:

The estimated standard deviation in the yield of apple and pear trees if the data in the database is only a sample of the total orchard population is computed by DSTDEV(A4:E10,"Yield",A1:A3), which results in 2.97.

end example]

3.17.7.99 DSTDEVP

Syntax:

DSTDEVP (*database* , *field* , *criteria*)

Description: Computes the standard deviation of a population based on the entire population by using the numbers in a column of a list or database that match the specified criteria. (See the DAVERAGE function §3.17.7.77.)

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database.
field	text, number	The column to which *criteria* shall be applied.
criteria	reference	The range of cells that contains the specified conditions.

For a detailed description of each argument, see the DAVERAGE function §3.17.7.77.

Return Type and Value: number – The standard deviation of a population based on the cells that correspond to the specified criteria.

[*Example*: Using the data in the example in the DAVERAGE function §3.17.7.77:

The true standard deviation in the yield of apple and pear trees if the data in the database is the entire population is computed by DSTDEVP(A4:E10,"Yield",A1:A3), which results in 2.65.

end example]

3.17.7.100 DSUM

Syntax:

DSUM (*database* , *field* , *criteria*)

Description: Computes the sum of the values in a column of a list or database that match the specified criteria. (See the DAVERAGE function §3.17.7.77.)

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database.
field	text, number	The column to which *criteria* shall be applied.
criteria	reference	The range of cells that contains the specified conditions.

For a detailed description of each argument, see the DAVERAGE function §3.17.7.77.

Return Type and Value: number – The sum of the values of the cells that correspond to the specified criteria.

[*Example*: Using the data in the example in the DAVERAGE function §3.17.7.77:

The total profit from apple trees is computed by `DSUM(A4:E10,"Profit",A1:A2)`, which results in 225.

The total profit from apple trees with a height between 10 and 16 is computed by `DSUM(A4:E10,"Profit",A1:F2)`, which results in 75.

end example]

3.17.7.101 DURATION

Syntax:

DURATION (*settlement* , *maturity* , *coupon* , *yld* , *frequency* [, [*basis*]])

Description: Computes the Macauley duration for an assumed par value of $100. Duration is defined as the weighted average of the present value of the cash flows and is used as a measure of a bond price's response to changes in yield.

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.

Name	Type	Description
maturity	number	The security's maturity date.
coupon	number	The security's annual coupon rate.
yld	number	The security's annual yield.
frequency	number	The number of coupon payments per year. For annual payments, frequency is 1; for semiannual payments, frequency is 2; for quarterly payments, frequency is 4. *frequency* is truncated to an integer.
basis	number	The truncated integer type of day count basis to use, as follows: 0 or omitted: US (NASD) 30/360 1: Actual/actual 2: Actual/360 3: Actual/365 4: European 30/360

Time information in the date arguments is ignored.

Return Type and Value: number – The Macauley duration for an assumed par value of $100.

However, if

- *settlement* or *maturity* is out of range for the current date base value, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *coupon* or *yld* < 0, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`DURATION(DATE(2008,1,1),DATE(2016,1,1),0.08,0.09,2,1)` results in `5.993774956`

end example]

3.17.7.102 DVAR

Syntax:

DVAR (*database* , *field* , *criteria*)

Description: Estimates the variance of a population based on a sample by using the numbers in a column of a list or database that match the specified criteria. (See the DAVERAGE function §3.17.7.77.)

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database.
field	text, number	The column to which *criteria* shall be applied.
criteria	reference	The range of cells that contains the specified conditions.

For a detailed description of each argument, see the DAVERAGE function §3.17.7.77.

Return Type and Value: number – An estimate of the variance of a population based on the cells that correspond to the specified criteria.

[*Example*: Using the data in the example in the DAVERAGE function §3.17.7.77:

The estimated variance in the yield of apple and pear trees if the data in the database is only a sample of the total orchard population is computed by `DVAR(A4:E10,"Yield",A1:A3)`, which results in `8.8`.

end example]

3.17.7.103 DVARP

Syntax:

DVARP (*database* , *field* , *criteria*)

Description: Calculates the variance of a population based on the entire population by using the numbers in a column of a list or database that match the specified criteria. (See the DAVERAGE function §3.17.7.77.)

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database.
field	text, number	The column to which *criteria* shall be applied.
criteria	reference	The range of cells that contains the specified conditions.

For a detailed description of each argument, see the DAVERAGE function §3.17.7.77.

Return Type and Value: number – The variance of a population based on the entire population using the cells that correspond to the specified criteria.

[*Example*: Using the data in the example in the DAVERAGE function §3.17.7.77:

The true variance in the yield of apple and pear trees if the data in the database is the entire orchard population is computed by DVARP(A4:E10,"Yield",A1:A3), which results in 7.04.

end example]

3.17.7.104 EDATE

Syntax:

EDATE (*start-date* , *month-offset*)

Description: Computes the serial value of the date that is *month-offset* months from the date specified by the date *date-string*, taking into account the current date base value.

Arguments:

Name	Type	Description
start-date	number	The start date.
month-offset	number	The number of months before or after *start-date*, truncated to integer. A positive value yields a future date; a negative value yields a past date; a zero value yields the date *start-date*.

Return Type and Value: number – The serial value of the date that is *month-offset* months from the date specified by the date *date-string*, as a whole number.

However, if

- *start-value* is out of range for the current date base value, #NUM! is returned.
- *start-value* plus *month-offset* is out of range for the current date base value, #NUM! is returned.

[*Example*: For the 1900 date base system:

EDATE(DATE(2006,1,31),5) results in a serial value of 38898
EDATE(DATE(2004,2,29),12) results in a serial value of 38411
EDATE(DATE(2004,2,28),12) results in a serial value of 38411
EDATE(DATE(2004,1,15),-23) results in a serial value of 37302

For the 1904 date base system:

EDATE(DATE(2006,1,31),5) results in a serial value of 37436
EDATE(DATE(2004,2,29),12) results in a serial value of 36949
EDATE(DATE(2004,2,28),12) results in a serial value of 36949

`EDATE(DATE(2004,1,15),-23)` results in a serial value of `35840`

end example]

3.17.7.105 EFFECT

Syntax:

`EFFECT` (*nominal-rate* , *npery*)

Description: Computes the effective annual interest rate, given the nominal annual interest rate and the number of compounding periods per year.

Mathematical Formula:

$$EFFECT = \left(1 + \frac{Nominal\_rate}{Npery}\right)^{Npery} - 1$$

Arguments:

Name	Type	Description
nominal-rate	number	The nominal interest rate.
npery	number	The number of compounding periods per year, truncated to integer.

Return Type and Value: number – The effective annual interest rate.

However, if

- *nominal-rate* ≤ 0, `#NUM!` is returned.
- *npery* < 1, `#NUM!` is returned.

[*Example*:

`EFFECT(0.0525,4)` results in `5.3543%`

end example]

3.17.7.106 EOMONTH

Syntax:

`EOMONTH` (*start-date* , *month-offset*)

Description: Computes the serial value of the last day of the month for the date that is *month-offset* months from the date specified by the date *start-date*, taking into account the current date base value.

Arguments:

Name	Type	Description
start-date	number	The start date.
month-offset	number	The number of months before or after *start-date*, truncated to integer. A positive value yields a future date; a negative value yields a past date; a zero value yields the date *start-date*.

Return Type and Value: number – The serial value of the last day of the month for the date that is *month-offset* months from the date specified by the date *start-date*, as a whole number.

However, if

- *start-date* is not a valid date, `#NUM!` is returned.
- *start-date* plus *month-offset* yields an invalid date, `#NUM!` is returned.

[*Example*: For the 1900 date base system:

`EOMONTH(DATE(2006,1,31),5)` results in a serial value of 38898
`EOMONTH(DATE(2004,2,29),12)` results in a serial value of 38411
`EOMONTH(DATE(2004,2,28),12)` results in a serial value of 38411
`EOMONTH(DATE(2004,1,15),-23)` results in a serial value of 37315

For the 1904 date base system:

`EOMONTH(DATE(2006,1,31),5)` results in a serial value of 37436
`EOMONTH(DATE(2004,2,29),12)` results in a serial value of 36949
`EOMONTH(DATE(2004,2,28),12)` results in a serial value of 36949
`EOMONTH(DATE(2004,1,15),-23)` results in a serial value of 35853

end example]

3.17.7.107 ERF

Syntax:

`ERF ( lower-bound [ , upper-bound ] )`

Description: Computes the error function integrated between *lower-bound* and *upper-bound*.

Mathematical Formula:

If *upper-bound* is omitted:

$$\mathrm{ERF}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2}\,dt$$

If *upper-bound* is present:

$$\mathrm{ERF}(a,b) = \frac{2}{\sqrt{\pi}} \int_a^b e^{-t^2}\,dt = \mathrm{ERF}(b) - \mathrm{ERF}(a)$$

Arguments:

Name	Type	Description
lower-bound	number	The lower bound for integrating ERF.
upper-bound	number	The upper bound for integrating ERF. If omitted, the value of the upper bound is *lower-bound*, and the lower bound becomes zero.

Return Type and Value: number – The error function integrated between *lower-bound* and *upper-bound*.

However, if

- *lower-bound* is negative, `#NUM!` is returned.
- *upper-bound* is negative, `#NUM!` is returned.

[*Example*:

`ERF(1.234,4.5432)` results in `0.08096060`
`ERF(0,1.345)` results in `0.94284416`
`ERF(0,1.345)` results in `0.94284416`

end example]

3.17.7.108 ERFC

Syntax:

ERFC (*lower-bound*)

Description: Computes the complementary error function integrated between *lower-bound* and ∞.

Mathematical Formula:

$$\mathrm{ERFC}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2}\,dt = 1 - \mathrm{ERF}(x)$$

Arguments:

Name	Type	Description
lower-bound	number	The lower bound for integrating ERFC.

Return Type and Value: number – The complementary error function integrated between *lower-bound* and ∞.

However, if *lower-bound* is negative

lower-bound or *upper-bound* is negative, `#NUM!` is returned.

[*Example*:

`ERFC(1.234)` results in `0.08096060`
`ERFC(0)` results in `1.00000000`

end example]

3.17.7.109 ERROR.TYPE

Syntax:

`ERROR.TYPE ( value )`

Description: Determines the kind of the error value designated by *value*.

Arguments:

Name	Type	Description
value	any	A value whose type is to be determined. No conversion shall take place on the argument passed to this function.

Return Type and Value: number – The kind of the error value designated by *value*, as follows:

value	Return Value
`#NULL!`	1
`#DIV/0!`	2
`#VALUE!`	3
`#REF!`	4
`#NAME?`	5
`#NUM!`	6
`#N/A`	7
Anything else	#N/A

[*Example*:

`ERROR.TYPE(A1)` results in 2 if A1 evaluates to `#DIV/0!`
`ERROR.TYPE(A1)` results in 4 if A1 evaluates to `#REF/0!`
`ERROR.TYPE(A1)` results in 7 if A1 evaluates to `#N/A`
`ERROR.TYPE(A1)` results in `#N/A` if A1 evaluates to a non-error value, such as a number or text

end example]

3.17.7.110 EVEN

Syntax:

EVEN (*x*)

Description: Computes *x* rounded to the nearest even integer, away from zero. Regardless of the sign of *x*, a value is rounded up when adjusted away from zero.

Arguments:

Name	Type	Description
x	number	The value to be rounded.

Return Type and Value: number – The rounded value of *x*. If *x* is zero, the result is zero.

[*Example*:

`EVEN(1.5)` rounds 1.5 up to the nearest even integer; that is, to 2.
`EVEN(3)` rounds 3 up to the nearest even integer; that is, to 4.
`EVEN(2)` rounds 2 up to the nearest even integer; that is, to 2.
`EVEN(-1)` rounds -1 up to the nearest even integer; that is, to `-2`.

end example]

3.17.7.111 EXACT

Syntax:

EXACT (*string-1* , *string-2*)

Description: Performs a case-sensitive comparison of *string-1* and *string-2*.

Arguments:

Name	Type	Description
string-1	text	The two strings to be compared.

Name	Type	Description
string-2	text	

Return Type and Value: logical – TRUE if *string-1* and *string-2* have the exact same length and contents; otherwise, FALSE.

[*Example*:

EXACT("ABC","ABC") results in TRUE
EXACT("ABC","ABCD") results in FALSE
EXACT("Abc","aBC") results in FALSE
EXACT("","") results in TRUE

end example]

3.17.7.112 EXP

Syntax:

```
EXP ( x )
```

Description: Computes e^x, where the constant *e* is the base of the natural logarithm.

Arguments:

Name	Type	Description
x	number	The exponent to which *e* is to be raised.

Return Type and Value: number – e^x.

However, if *x* is too large for the result to be representable, #NUM! is returned.

[*Example*:

EXP(-1) results in 0.367879441
EXP(0) results in 1
EXP(1) results in 2.718281828
EXP(2) results in 7.389056099

end example]

3.17.7.113 EXPONDIST

Syntax:

EXPONDIST (*x* , *lambda* , *cumulative-flag*)

Description: Computes the exponential distribution.

Mathematical Formula:

The equation for the probability density function is:

$$f(x;\lambda) = \lambda e^{-\lambda x}$$

The equation for the cumulative distribution function is:

$$F(x;\lambda) = 1 - e^{-\lambda x}$$

Arguments:

Name	Type	Description
x	number	The value of the function.
lambda	number	The parameter value.
cumulative-flag	logical	Determines the form of the function. If TRUE, EXPONDIST returns the cumulative distribution function; if FALSE, EXPONDIST returns the probability density function.

Return Type and Value: number – The exponential distribution.

However, if

- *x* < 0, #NUM! is returned.
- *lambda* ≤ 0, #NUM! is returned.

[*Example*:

EXPONDIST(0.2,10,FALSE) results in 1.353352832
EXPONDIST(2.3,1.5,TRUE) results in 0.968254364

end example]

3.17.7.114 FACT

Syntax:

FACT (*x*)

Description: Computes the factorial of *x*.

Arguments:

Name	Type	Description
x	number	The non-negative value whose factorial is to be computed. *x* is truncated to an integer.

Return Type and Value: number – The factorial of *x*.

However, if

- *x* is negative, `#NUM!` is returned.
- *x* is too large for the result to be representable, `#NUM!` is returned.

[*Example*:

`FACT(5)` results in `120`
`FACT(3.5)` results in `6`
`FACT(0)` results in `1`

end example]

3.17.7.115 FACTDOUBLE

Syntax:

FACTDOUBLE (*n*)

Description: Computes the double factorial of *n*.

Mathematical Formula:

If *n* is even:

$$n!! = n(n-2)(n-4)\ldots(4)(2)$$

If *n* is odd:

$$n!! = n(n-2)(n-4)\ldots(3)(1)$$

Arguments:

Name	Type	Description
n	number	The non-negative value whose double factorial is to be computed. *n* is truncated to an integer.

Return Type and Value: number – The double factorial of *n*.

However, if

- *n* is negative, #NUM! is returned.
- *n* is too large for the result to be representable, #NUM! is returned.

[*Example*:

FACTDOUBLE(5) results in 15
FACTDOUBLE (3.5) results in 3
FACTDOUBLE (0) results in 1

end example]

3.17.7.116 FALSE

Syntax:

FALSE ( )

Description: Computes the value FALSE. (A call to function FALSE is equivalent to using the *logical-constant* FALSE.)

Arguments: None.

Return Type and Value: logical – The value FALSE.

[*Example*:

FALSE() results in FALSE

end example]

3.17.7.117 FDIST

Syntax:

FDIST (*x* , *degrees-freedom-1* , *degrees-freedom-2*)

Description: Computes the F probability distribution.

Mathematical Formula:

FDIST=P(F>x), where F is a random variable that has an F distribution with *degrees-freedom-1* and *degrees-freedom-2* degrees of freedom.

Arguments:

Name	Type	Description
x	number	The value at which the function is to be evaluated.
degrees-	number	The number of degrees of freedom for the numerator,

Name	Type	Description
freedom-1		truncated to an integer.
degrees-freedom-2	number	The number of degrees of freedom for the denominator, truncated to an integer.

Return Type and Value: number – The F probability distribution.

However, if

- *x* is negative, `#NUM!` is returned.
- *degrees-freedom-1* < 1 or *degrees-freedom-1* ≥ 10^{10}, `#NUM!` is returned.
- *degrees-freedom-2* < 1 or *degrees-freedom-2* ≥ 10^{10}, `#NUM!` is returned.

[*Example*:

`FDIST(12.345,3,4)` results in `0.017226183`

end example]

3.17.7.118 FIND

Syntax:

`FIND ( string-1 , string-2 [ , start-pos ] )`

Description: Performs a case-sensitive search for the first occurrence of *string-1* in *string-2*, starting at character position *start-pos* within *string-2*. (`FIND` is intended for use with languages that use the single-byte character set (SBCS), whereas `FINDB` (§3.17.7.119) is intended for use with languages that use the double-byte character set (DBCS).)

Arguments:

Name	Type	Description
string-1	text	Designate the string to be searched for within the string designated by *string-2*.
string-2	text	
start-pos	number	The number of the start position within *string-2* for which *string-1* is to be searched. The start position of the first character is 1. If omitted, a position of 1 shall be assumed. *start-pos* shall be at least 0.

Return Type and Value: number – The start position of the first occurrence of *string-1* in *string-2*, starting at character position *start-pos* within *string-2*. If *string-1* is an empty string, it shall always be found in any *string-2* at position *start-pos*, or at position 1 if *start-pos* is omitted.

However, if

- *string-1* is not found within *string-2*, #VALUE! is returned.
- *start-pos* designates a position outside *string-2*, #VALUE! is returned.

[*Example*:

`FIND("de","abcdef")` results in 4
`FIND(A10,B10)` results in 4, when A10 contains `de`, and B10 contains `abcdef`

end example]

3.17.7.119 FINDB

Syntax:

`FINDB ( string-1 , string-2 , [ start-pos ] )`

Description: Performs a case-sensitive search for the first occurrence of *string-1* in *string-2*, starting at byte position *start-pos* within *string-2*. (`FINDB` is intended for use with languages that use the double-byte character set (DBCS), whereas `FIND` (§3.17.7.118) is intended for use with languages that use the single-byte character set (SBCS).)

Arguments:

Name	Type	Description
string-1	text	Designate the string to be searched for within the string designated by *string-2*.
string-2	text	
start-pos	number	The number of the start position within *string-2* for which *string-1* is to be searched. The start position of the first byte is 1. If omitted, a position of 1 shall be assumed. *start-pos* shall be at least 0.

Return Type and Value: number – The start position of the first occurrence of *string-1* in *string-2*, starting at character position *start-pos* within *string-2*. If *string-1* is an empty string, it shall always be found in any *string-2* at position *start-pos*, or at position 1 if *start-pos* is omitted.

However, if

- *string-1* is not found within *string-2*, #VALUE! is returned.
- *start-pos* designates a position outside *string-2*, #VALUE! is returned.

[*Example*: Assuming 1-byte characters

`FINDB("de","abcdef")` results in 4

FINDB(A10,B10) results in 4, when A10 contains de, and B10 contains abcdef

end example]

3.17.7.120 FINV

Syntax:

FINV (*probability* , *degrees-freedom-1* , *degrees-freedom-2*)

Description: Computes the inverse of the F probability distribution. Given a value for *probability*, FINV seeks that value *x* such that FDIST(*x*, *degrees-freedom1*, *degrees-freedom2*) = *probability*. Thus, precision of FINV depends on precision of FDIST. FINV uses an iterative search technique.

Arguments:

Name	Type	Description
probability	number	A probability associated with the F cumulative distribution.
degrees-freedom-1	number	The number of degrees of freedom for the numerator, truncated to an integer.
degrees-freedom-2	number	The number of degrees of freedom for the denominator, truncated to an integer.

Return Type and Value: number – The inverse of the F probability distribution.

However, if

- *probability* < 0 or *probability* > 1, #NUM! is returned.
- *degrees-freedom-1* < 1 or *degrees-freedom-1* ≥ 10^{10}, #NUM! is returned.
- *degrees-freedom-2* < 1 or *degrees-freedom-2* ≥ 10^{10}, #NUM! is returned.
- The search has not converged after some implementation-defined number of iterations, #N/A is returned

[*Example*:

FINV(0.5,3,4) results in 0.940534076

end example]

3.17.7.121 FISHER

Syntax:

FISHER (*x*)

Description: Computes the Fisher transformation at *x*.

Mathematical Formula:

$$z' = \frac{1}{2}\ln\left(\frac{1+x}{1-x}\right)$$

Arguments:

Name	Type	Description
x	number	The number for which the transformation is wanted.

Return Type and Value: number – The Fisher transformation at *x*.

However, if

- *x* ≤ -1, `#NUM!` is returned.
- *x* ≥ 1, `#NUM!` is returned.

[*Example*:

`FISHER(-0.43)` results in `-0.459896681`
`FISHER(0.578)` results in `0.659454094`

end example]

3.17.7.122 FISHERINV

Syntax:

```
FISHERINV ( y )
```

Description: Computes the inverse of the Fisher transformation.

Mathematical Formula:

$$x = \frac{e^{2y} - 1}{e^{2y} + 1}$$

Arguments:

Name	Type	Description
y	number	The number for which the inverse of the transformation is wanted.

Return Type and Value: number – The inverse of the Fisher transformation.

[*Example*:

`FISHERINV(-0.43)` results in `0.405321309`
`FISHERINV(0.578)` results in `0.521210269`

end example]

3.17.7.123 FIXED

Syntax:

`FIXED ( number [ , [ num-decimal ] [ , suppress-commas-flag ] ] )`

Description: Produces a string containing *number* rounded to *num-decimal* decimal places. Thousands separator commas are included as determined by *suppress-commas-flag*.

Arguments:

Name	Type	Description
number	number	Designate the number that is to be formatted, truncated to integer.
num-decimal	number	Designate the number of decimal places to be used in the resulting string. If negative, *number* is rounded to the left of the decimal point. If omitted, a value of 2 shall be assumed.
suppress-commas-flag	logical	If TRUE, commas are not included; if FALSE or omitted, commas are included.

Return Type and Value: text – The string containing *number* rounded to *num-decimal* decimal places.

[*Example*:

`FIXED(1234567)` results in `1,234,567.00`
`FIXED(1234567.555555,4,TRUE)` results in `1234567.5556`
`FIXED(.555555,10)` results in `0.5555550000`
`FIXED(1234567,-3)` results in `1,235,000`

end example]

3.17.7.124 FLOOR

Syntax:

`FLOOR ( x , significance )`

Description: Computes *x* rounded down, toward zero, to the nearest multiple of *significance*. Regardless of the sign of *x*, a value is rounded down when adjusted away from zero.

Arguments:

Name	Type	Description
x	number	The value to be rounded,
significance	number	The multiple to which *x* is to be rounded.

Return Type and Value: number – The rounded-down value of *x*.

However, if *x* and *significance* have different signs, `#NUM!` is returned.

[*Example*:

`FLOOR(2.5,1)` rounds 2.5 down to nearest multiple of 1; that is, to 2
`FLOOR(-2.5,-2)` rounds -2.5 down to nearest multiple of -2; that is, to `-2`
`FLOOR(1.5,0.1)` rounds 1.5 down to the nearest multiple of 0.1; that is, to `1.5`
`FLOOR(0.234,0.01)` rounds 0.234 down to the nearest multiple of 0.01; that is, to 0.23

end example]

3.17.7.125 FORECAST

Syntax:

FORECAST (*x* , *known-ys* , *known-xs*)

Description: Calculates, or predicts, a future value by using existing values. The predicted value is a y-value for a given x-value. The known values are existing x-values and y-values, and the new value is predicted by using linear regression.

Mathematical Formula:

FORECAST=a+bx, where:

$$a = \bar{y} - b\bar{x}$$

and:

$$b = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sum (x - \bar{x})^2}$$

and where x and y are the sample means `AVERAGE(known-xs)` and `AVERAGE(known-ys)`.

Arguments:

Name	Type	Description
x	number	The data point for which a value is to be predicted.
known-xs	array, reference	The independent data.
known-ys	array, reference	The dependent data.

Return Type and Value: number – The future value.

However, if

- *known-xs* and *known-ys* are empty or contain a different number of data points, the return value is unspecified.
- The variance of *known-xs* equals zero, the return value is unspecified.

[*Example*:

`FORECAST(30,{6,7,9,15,21},{20,28,31,38,40})` results in `10.60725309`

end example]

3.17.7.126 FREQUENCY

Syntax:

FREQUENCY (*data-array* , *bins-array*)

Description: Calculates how often values occur within a range of values. A call to FREQUENCY shall be entered as an array formula.

Arguments:

Name	Type	Description
data-array	array, reference to number	Set of values for which frequencies are to be computed. If *data-array* contains no values, FREQUENCY returns an array of zeros. Cells containing text or that are empty are ignored.
bins-array	array, reference	Set of intervals into which the values in *data-array* are to be grouped. If *bins-array* contains no values, FREQUENCY returns the number of elements in *data-array*.

Return Type and Value: vertical array of numbers – The frequency at which values occur within a range of values. The number of elements in the returned array is one more than the number of elements in *bins-array*. The extra element contains the count of any values above the highest interval.

[*Example*:

If the cells A2:A10 contain 79, 85, 78, 85, 50, 81, 95, 88, and 97, and the cells B2:B4 contain 70, 79, and 89, `FREQUENCY(A2:A10,B2:B4)` results in a vertical array containing 1 (50), 2 (79, 78), 4 (85, 85, 81, 88), and 2 (95, 97).

end example]

3.17.7.127 FTEST

Syntax:

FTEST (*array-1* , *array-2*)

Description: Computes the result of an F-test.

Arguments:

Name	Type	Description
array-1	number, name, array, reference to number	If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.
array-2		

Return Type and Value: number – The two-tailed probability that the variances in *array-1* and *array-2* are not significantly different.

However, if

- The number of data points in *array-1* or *array-2* is less than 2, the return value is unspecified.
- The variance of *array-1* or *array-2* is zero, the return value is unspecified.

[*Example*:

If the cells D6:D10 contain 6, 7, 9, 15, and 21, and the cells E6:E10 contain 20, 28, 31, 38, and 40, `FTEST(D6:D10,E6:E10)` results in `0.648317847`

end example]

3.17.7.128 FV

Syntax:

FV (*rate* , *nper* , *pmt* [, [*pv*] [, [*type*]]])

Description: Computes the future value of an investment based on periodic, constant payments and a constant interest rate.

Arguments:

Name	Type	Description
rate	number	The interest rate.
nper	number	The total number of payment periods, truncated to integer.
pmt	number	The payment made each period; it cannot change over the life of the annuity. [*Note*: Typically, *pmt* contains principal and interest, but no other fees or taxes. *end note*] If omitted, *pv* shall be provided.
pv	number	The the present value, or the lump-sum amount that a series of future payments is worth right now. If omitted, it is assumed to be 0, and *pmt* shall be provided.
type	number	The timing of the payment, truncated to integer, as follows: **Value** / **Timing** `0` – Payment at the end of the period `1` – Payment at the beginning of the period

Arguments representing cash paid by investor shall be expressed as negative numbers; arguments representing cash received by the investor shall be expressed as positive numbers.

Return Type and Value: number – The future value of an investment based on periodic, constant payments and a constant interest rate.

However, if *type* is any number other than 0 or 1, `#NUM!` is returned.

[*Example*:

`FV(0.06/12,10,-200,-500,1)` `2,581.40`
`FV(0.12/12,12,-1000)` results in `12,682.50`
`FV(0.11/12,35,-2000,,1)` results in `82,846.25`
`FV(0.06/12,12,-100,-1000,1)` results in `2,301.40`

end example]

3.17.7.129 FVSCHEDULE

Syntax:

FVSCHEDULE (*principal* , *schedule*)

Description: Computes the future value of an initial principal after applying a series of compound interest rates. [*Note*: This function can be used to calculate the future value of an investment with a variable or adjustable rate. *end note*]

Arguments:

Name	Type	Description
principal	number	The present value.
schedule	array	Set of interest rates to apply. The values in this array can be numbers or blank cells. Blank cells are taken as zeros (i.e., no interest).

Return Type and Value: number – The future value of an initial principal after applying a series of compound interest rates.

However, if any element of the array *schedule* is not a number and not blank, #VALUE ! is returned.

[*Example*:

FVSCHEDULE(1,{0.09,0.11,0.1}) results in 1.33089

end example]

3.17.7.130 GAMMADIST

Syntax:

GAMMADIST (*x* , *alpha* , *beta* , *cumulative-flag*)

Description: Computes the gamma distribution.

Mathematical Formula:

The equation for the gamma probability density function is:

$$f(x;\alpha,\beta) = \frac{1}{\beta^{\alpha}\Gamma(\alpha)} x^{\alpha-1} e^{-\frac{x}{\beta}}$$

The standard gamma probability density function is:

$$f(x;\alpha) = \frac{x^{\alpha-1} e^{-x}}{\Gamma(\alpha)}$$

When *alpha* = 1, GAMMADIST returns the exponential distribution with:

$$\lambda = \frac{1}{\beta}$$

For a positive integer n, when *alpha* = n/2, *beta* = 2, and cumulative = TRUE, GAMMADIST returns (1-CHIDIST(x)) with n degrees of freedom.

When *alpha* is a positive integer, GAMMADIST is also known as the Erlang distribution.

Arguments:

Name	Type	Description
x	number	The value at which the distribution is to be evaluated.
alpha	number	A parameter of the distribution.
beta	number	A parameter of the distribution. If *beta* = 1, GAMMADIST returns the standard gamma distribution.
cumulative-flag	logical	Determines the form of the function. If TRUE, GAMMADIST returns the cumulative distribution function; if FALSE, it returns the probability density function.

Return Type and Value: number – The gamma distribution.

However, if

- *x* < 0, #NUM! is returned.
- *alpha* ≤ 0 or *beta* ≤ 0, #NUM! is returned.

[*Example*:

GAMMADIST(10,9,2,FALSE) results in 0.03263902
GAMMADIST(10,9,2,TRUE) results in 0.068093631

end example]

3.17.7.131 GAMMAINV

Syntax:

GAMMAINV (*probability* , *alpha* , *beta*)

Description: Computes the inverse of the gamma distribution. Given a value for *probability*, GAMMAINV seeks that value *x* such that GAMMADIST(*x*, *alpha*, *beta*, TRUE) = *probability*. Thus, the precision of GAMMAINV depends on the precision of GAMMADIST. GAMMAINV uses an iterative search technique.

Arguments:

Name	Type	Description

Name	Type	Description
probability	number	The probability associated with the gamma distribution.
alpha	number	A parameter of the distribution.
beta	number	A parameter of the distribution. If *beta* = 1, GAMMAINV returns the standard gamma distribution.

Return Type and Value: number – The inverse of the gamma distribution.

However, if

- *probability* < 0 or *probability* > 1, #NUM! is returned.
- *alpha* ≤ 0 or *beta* ≤ 0, #NUM! is returned.
- The search has not converged after some implementation-defined number of iterations, #N/A is returned.

[*Example*:

GAMMAINV(0.068,9,2) results in 9.997130086

end example]

3.17.7.132 GAMMALN

Syntax:

GAMMALN (*x*)

Description: Computes the natural logarithm of the gamma function.

Mathematical Formula:

$$GAMMALN = LN(\Gamma(x))$$

where:

$$\Gamma(x) = \int_0^{\infty} e^{-u} u^{x-1} du$$

Arguments:

Name	Type	Description
x	number	The value for which the gamma function is to be calculated.

Return Type and Value: number – The natural logarithm of the gamma function.

However, if $x \leq 0$, `#NUM!` is returned.

[*Example*:

`GAMMALN(4.5)` results in `2.453736571`

end example]

3.17.7.133 GCD

Syntax:

GCD (*argument-list*)

Description: Computes the greatest common divisor of the one or more numbers, designated by *argument*s in *argument-list*.

Arguments:

Name	Type	Description
argument-list	numbers	The *argument*s in *argument-list* designate the values. Each argument is truncated to an integer.

Return Type and Value: number – The greatest common divisor of one or more numbers.

However, if any *argument* is negative, `#NUM!` is returned.

[*Example*:

`GCD(5)` results in `5`
`GCD(5,2)` results in `1`
`GCD(100,50,28)` results in `2`
`GCD(24.5,36.3)` results in `12`
`GCD(7,1)` results in `1`
`GCD(5,0)` results in `5`

end example]

3.17.7.134 GEOMEAN

Syntax:

GEOMEAN (*argument-list*)

Description: Computes the geometric mean of an array or range of positive data.

Mathematical Formula:

$$GM_{\bar{y}} = \sqrt[n]{y_1 y_2 y_3 \cdots y_n}$$

Arguments:

Name	Type	Description
argument-list	logical, number, name, array, or reference to number.	The *argument*s in *argument-list* designate the values to be averaged. Logical values and text representations of numbers that entered directly into the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The geometric mean of an array or range of positive data.

However, if the value of any data point ≤ 0, `#NUM!` is returned.

[*Example*:

`GEOMEAN(10.5,5.3,2.9)` results in `5.444454702`
`GEOMEAN(10.5,{5.3,2.9},"12")` results in `6.633780588`

end example]

3.17.7.135 GESTEP

Syntax:

`GESTEP ( number [ , step ] )`

Description: Tests if the value of *number* is greater than or equal to that of *step*.

Arguments:

Name	Type	Description
number	number	*number* is the value to test against *step*. If *step* is omitted, zero is used.
step	number	

Return Type and Value: number – 1 if *number* ≥ *step*; otherwise, 0.

[*Example*:

`GESTEP(5.6,-4.3)` results in `1`
`GESTEP(5.6,5.6)` results in `1`

`GESTEP(-5.6)` results in `0`

end example]

3.17.7.136 GETPIVOTDATA

Syntax:

```
GETPIVOTDATA ( data-field , pivot-table , field-1 , item-1
    [ , field-2 , item-2 [ , … ] ] )
```

Description: Retrieves data stored in a PivotTable report. Calculated fields or items and custom calculations are included in `GETPIVOTDATA` calculations.

Arguments:

Name	Type	Description
data-field	text	The name of the data field that contains the data to be retrieved.
pivot-table	reference to any cell, range of cells, or named range of cells in a PivotTable report	This information is used to determine which PivotTable report contains the data to be retrieved. If *pivot-table* is a range that includes two or more PivotTable reports, data shall be retrieved from whichever report was created most recently in the range.
field-1 through *field-n*	text	Argument pairs *field-1* and *item-1*, *field-2* and *item-2* through *field-n* and *item-n* are field names and item names that describe the data to be retrieved. The pairs can be in any order. Field names and names for items other than dates/times (which shall be expressed as numbers) and numbers shall be enclosed in quotation marks. For OLAP PivotTable reports, items can contain the source name of the dimension as well as the source name of the item. [*Example*: A field and item pair for an OLAP PivotTable might look like this: `"[Product]","[Product].[All Products].[Foods].[Baked Goods]"` *end example*] If the field and item arguments describe a single cell, the value of that cell is returned regardless of its type or value.
item-1 through *item-n*	text	

Return Type and Value: any – The data stored in a PivotTable report.

However, if

- *pivot-table* is not a range in which a PivotTable report is found, the return value is unspecified.
- The arguments do not describe a visible field, the return value is unspecified.
- The arguments include a page field that is not displayed, the return value is unspecified.

[*Example*: Given the following data:

	A	B	C	D	E
2	Region	North			
3					
4	Sum of Sales		Product		
5	Month	Salesperson	Beverages	Produce	Grand Total
6	March	Buchanan	$ 3,522	$ 10,201	$ 13,723
7		Davolio	$ 8,725	$ 7,889	$ 16,614
8	March Total		$ 12,247	$ 18,090	$ 30,337
9	April	Buchanan	$ 5,594	$ 7,265	$ 12,859
10		Davolio	$ 5,461	$ 668	$ 6,129
11	April Total		$ 11,055	$ 7,933	$ 18,988
12	Grand Total		$ 23,302	$ 26,023	$ 49,325

`GETPIVOTDATA("Sales",$A$4)` returns the grand total of the Sales field, `$49,325`.

`GETPIVOTDATA("Sum of Sales",$A$4)` also returns the grand total of the Sales field, `$49,325`; the field name can be entered exactly as it looks on the sheet, or as its root (without "Sum of," "Count of," and so forth).

`GETPIVOTDATA("Sales",$A$4,"Month","March")` returns the grand total for March, `$30,337`.

`GETPIVOTDATA("Sales",$A$4,"Month","March","Product","Produce","Salesperson","Buchanan")` returns `$10,201`.

`GETPIVOTDATA("Sales",$A$4,"Region","South")` is unspecified because the South region data is not visible.

`GETPIVOTDATA("Sales",$A$4,"Product","Beverages","Salesperson","Davolio")` is unspecified because there is no total value of beverage sales for Davolio.

end example]

3.17.7.137 GROWTH

Syntax:

`GROWTH ( known-ys [ , [ known-xs ] [ , [ new-xs ] [ , const-flag ] ] )`

Description: Computes predicted exponential growth by using existing data. GROWTH can also fit an exponential curve to existing x-values and y-values.

Arguments:

Name	Type	Description
known-ys	array	Set of y-values already known in the relationship $y=b*m^x$. If the array *known-ys* is a single column, then each column of *known-xs* is interpreted as a separate variable. If the array *known-ys* is a single row, then each row of *known-xs* is interpreted as a separate variable.
known-xs	array	Set of x-values that might already be know in the

Name	Type	Description
		relationship y=b*m^x. The array *known-xs* can include one or more sets of variables. If only one variable is used, *known-ys* and *known-xs* can be ranges of any shape, as long as they have equal dimensions. If more than one variable is used, *known-ys* must be a vector (that is, a *known-ys* with a height of one row or a width of one column). If *known-xs* is omitted, it is assumed to be the array {1,2,3,...} that is the same size as *known-ys*.
new-xs	array	A set of new x-values for which GROWTH is to return corresponding y-values. *new-xs* shall include a column (or row) for each independent variable, just as *known-xs* does. So, if *known-ys* is in a single column, *known-xs* and *new-xs* shall have the same number of columns. If *known-ys* is in a single row, *known-xs* and *new-xs* shall have the same number of rows. If *new-xs* are omitted, it is assumed to be the array {1,2,3,...} that is the same size as *known-ys*.
const-flag	logical	Specifies whether to force the constant b to equal 1. If TRUE or omitted, *b* is calculated normally. If FALSE, *b* is set equal to 1 and the m-values are adjusted so that y= m^x.

Return Type and Value: array – The y-values for a series of new x-values.

However, if any of the numbers in *known-ys* are zero or negative, `#NUM!` is returned.

[*Example*: Given the following data:

	A	B	C
1	Month	Units	Formula (corresponding units)
2	11	33,100	32618.20377
3	12	47,300	47729.42261
4	13	69,000	69841.30086
5	14	102,000	102197.0734
6	15	150,000	149542.4867
7	16	220,000	218821.8762
8	Month	Formula (Predicted Units)	
9	17	320,196.72	
10	18	468,536.05	

When `GROWTH(A2:B4,A6:B8)` is array-entered into cells C2:C7, those cells take on the results shown.

When GROWTH(A2:B4,A6:B8,A9:A10) is array-entered into cells B9:B10, those cells take on the results shown.

end example]

3.17.7.138 HARMEAN

Syntax:

HARMEAN (*argument-list*)

Description: Computes the harmonic mean of a data set.

Mathematical Formula:

$$\frac{1}{H_g} = \frac{1}{n}\sum\frac{1}{Y_i}$$

Arguments:

Name	Type	Description
argument-list	logical, text, number, name, array, or reference to number.	The *argument*s in *argument-list* designate the values to be averaged. Argument values can be numbers, or names, arrays, or references that contain numbers. Logical values and text representations of numbers entered directly into the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The harmonic mean of a data set.

However, if the value of any data point ≤ 0, #NUM! is returned.

[*Example*:

HARMEAN(4.6,5.8,8.3,7) results in 6.124222
HARMMEAN(10.5,{5.3,2.9},"12") results in 5.617360

end example]

3.17.7.139 HEX2BIN

Syntax:

HEX2BIN (*number* [, *num-bin-digits*])

Description: Makes the binary equivalent of *number*, with the result having *num-bin-digits* digits.

Arguments:

Name	Type	Description
number	text	A 10-digit hexadecimal number in a string that is to be converted to a binary string. *number* is not case-sensitive. If *number* has less than 10 digits, leading zero digits are implied until it has exactly 10 digits. The 10 digits use twos-complement representation with the left-most bit (40th bit from the right) representing the sign bit.
num-bin-digits	number	The number of digits in the result, with leading zeros added as necessary. However, if *number* is negative, *num-bin-digits* is ignored and the result has 10 digits. If *num-bin-digits* is omitted, the minimum number of digits is used in the result. *num-bin-digits* is truncated to an integer.

Return Type and Value: text – The binary equivalent of *number*.

However, if

- *number* is outside the range `"FFFFFFFE00"` (1111111111111111111111111111111000000000 binary, -512 decimal) to `"1FF"` (0000000000000000000000000000000111111111 binary, 511 decimal), inclusive, `#NUM!` is returned.
- *number* contains one or more non-hexadecimal digits, `#NUM!` is returned.
- *number* contains more than 10 hexadecimal digits, `#NUM!` is returned.
- *number* needs more digits that *num-bin-digits*, `#NUM!` is returned.
- *num-bin-digits* is negative or > 10, `#NUM!` is returned.

[*Example*:

`HEX2BIN("fE")` results in `11111110`
`HEX2BIN("FFFFFFFFFE")` results in `1111111110`
`HEX2BIN("2")` results in `10`
`HEX2BIN("F",6)` results in `001111`

end example]

3.17.7.140 HEX2DEC

Syntax:

HEX2DEC (*number*)

Description: Makes the decimal equivalent of *number*.

Arguments:

Name	Type	Description
number	string	A 10-digit hexadecimal number in a string that is to be converted to a decimal number. *number* is not case-sensitive. If *number* has less than 10 digits, leading zero digits are implied until it has exactly 10 digits. The 10 digits use twos-complement representation with the left-most bit (40th bit from the right) representing the sign bit.

Return Type and Value: number – The decimal equivalent of *number*.

However, if

- *number* contains one or more non-hexadecimal digits, `#NUM!` is returned.
- *number* contains more than 10 hexadecimal digits; that is, *number* is outside the range `"8000000000"` (-548,755,813,888 decimal) to `"7FFFFFFFFF"` (548,755,813,887 decimal), inclusive, `#NUM!` is returned.

[*Example*:

`HEX2DEC("fE")` results in `254`
`HEX2DEC("FFFFFFFFFE")` results in `-2`
`HEX2DEC("F000000000")` results in `-68719476736`

end example]

3.17.7.141 HEX2OCT

Syntax:

HEX2OCT (*number* [, *num-oct-digits*])

Description: Makes the octal equivalent of *number*, with the result having *num-oct-digits* digits.

Arguments:

Name	Type	Description
number	text	A 10-digit hexadecimal number in a string that is to be converted to an octal string. *number* is not case-sensitive. If *number* has less than 10 digits, leading zero digits are implied until it has exactly 10 digits. The 10 digits use twos-complement representation with the left-most bit (40th bit from the right) representing the sign bit.
num-oct-	number	The number of digits in the result, with leading zeros

Name	Type	Description
digits		added as necessary. However, if *number* is negative, *num-oct-digits* is ignored and the result has 10 digits. If *num-oct-digits* is omitted, the minimum number of digits is used in the result. *num-oct-digits* is truncated to an integer.

Return Type and Value: text – The octal equivalent of *number*.

However, if

- *number* is outside the range `"FFE0000000"` (17774000000000 octal, -536,870,912 decimal) to `"1FFFFFFF"` (00003777777777 octal, 536,870,911 decimal), inclusive, `#NUM!` is returned.
- *number* contains one or more non-hexadecimal digits, `#NUM!` is returned.
- *number* contains more than 10 hexadecimal digits, `#NUM!` is returned.
- *number* needs more digits that *num-oct-digits*, `#NUM!` is returned.
- *num-oct-digits* is negative or > 10, `#NUM!` is returned.

[*Example*:

`HEX2OCT("fE")` results in `376`
`HEX2OCT("FFFFFFFFFE")` results in `7777777776`
`HEX2OCT("2")` results in `2`
`HEX2OCT("F",6)` results in `000017`

end example]

3.17.7.142 HLOOKUP

Syntax:

HLOOKUP (*lookup-value* , *table-array* , *row-index-num* [, [*range-lookup-flag*]])

Description: Performs a horizontal search for a value in the top row of a table or an array, noting the column in which the matching value is found. From that column, the value from a given row is returned.

Arguments:

Name	Type	Description
lookup-value	value of any type or a reference to a value of any type.	The value to be located in the first row of the table. If *range-lookup* is FALSE and *lookup-value* is a string, the wildcard characters, question mark (?) and asterisk (*), can be included in *lookup-value*. A question mark matches any single character; an asterisk matches any sequence of characters. To find a question mark or

Name	Type	Description
		asterisk, type a tilde (~) before the character.
table-array	array, reference, name	Designates the table of information to be searched. The values in the first row of *table-array* can be text, numbers, or logical values. If *range-lookup-flag* is TRUE, the values in the first row of *table-array* shall be placed in "ascending order", as follows: ..., -2, -1, 0, 1, 2, ..., A–Z, FALSE, TRUE. If *range-lookup-flag* is FALSE, *table-array*'s values need not be sorted. Uppercase and lowercase text is treated as equivalent.
row-index-num	number	The row number in *table-array* from which the matching value is to be returned. (A *row-index-num* of 1 returns the first row value in *table-array*, a *row-index-num* of 2 returns the second row value in *table-array*, and so on.)
range-lookup-flag	logical	Specifies whether HLOOKUP is to find an exact or approximate match. If TRUE or omitted, an approximate match is returned. That is, if an exact match is not found, the next largest value that is less than *lookup-value* is returned. If FALSE, an exact match is performed.

Return Type and Value: any – The value from a given row number, where the column is determined by a search of the top row looking for a match with a given value.

However, if

- An exact match is performed, but no match is found, #N/A is returned.
- *row-index-num* is less than 1, #VALUE! is returned.
- *row-index-num* is greater than the number of rows in *table-array*, #REF! is returned.
- *lookup-value* is smaller than the smallest value in the first row of *table-array*, #N/A is returned.

[*Example*: Given the following data:

	A	B	C
1	Axles	Bearings	Bolts
2	4	6	9
3	5	7	10
4	6	8	11

`HLOOKUP("Axles",A1:C4,2,TRUE)` results in 4
`HLOOKUP("Bearings",A1:C4,3,FALSE)` results in 7
`HLOOKUP("B",A1:C4,3,TRUE)` results in 5
`HLOOKUP("Bolts",A1:C4,4)` results in 11

HLOOKUP(3,{1,2,3;"a","b","c";"d","e","f"},2,TRUE) results in c

end example]

3.17.7.143 HOUR

Syntax:

HOUR (*time-value*)

Description: Computes the hour for the date and/or time having the given *time-value*.

Arguments:

Name	Type	Description
time-value	number	The date and/or time whose hour is to be computed. That date and/or time shall be expressed either as a serial value, in which case, its integer part is ignored, or as a *string-constant* having any valid date and/or time format, in which case, any date information shall be ignored.

Return Type and Value: number – The hour for the date and/or time having the given *time-value*. The returned value shall be in the range 0–59.

However, if *time-value* is out of range for the current date base value, #NUM! is returned.

[*Example*:

HOUR(DATE(2006,2,26)+TIME(2,10,20)) results in 2
HOUR(TIME(22,56,34)) results in 22
HOUR(0) results in 0, since serial value 0 represents 00:00:00
HOUR(10.5) results in 12, since serial value .5 represents 12:00:00
HOUR("22-Oct-2001 10:53:12") results in 10
HOUR("10:53:12 pm") results in 22
HOUR("22:53:12") results in 22

end example]

3.17.7.144 HYPERLINK

Syntax:

HYPERLINK (*link-location* [, [*friendly-name*]])

Description: Creates a shortcut that opens a document stored on a network server, an intranet, or the Internet. When the cell that contains the HYPERLINK function call is clicked, the file stored at *link-location* is opened.

Arguments:

Name	Type	Description
link-location	text	The path and file name to the document to be opened as text. *link-location* can refer to a place in a document—such as a specific cell or named range in a SpreadsheetML worksheet or workbook, or to a bookmark in a WordprocessingML document. The path can be to a file stored on a hard disk drive, or the path can be a universal naming convention (UNC) path on a server or a Uniform Resource Locator (URL) path on the Internet or an intranet. If the path specified in *link-location* does not exist or cannot be navigated, an unspecified error is produced when the cell is clicked. *link-location* can be a string or a reference to a cell containing a string.
friendly-name	text, number, name	The value that is displayed in the cell. If omitted, the cell displays *link-location*. *friendly-name* can be a value, a text string, a name, or a cell that contains the jump text or value. If the evaluation of *friendly-name* results in an error value, the cell displays that error value rather than the jump text.

Return Type and Value: text – The value of *friendly-name*, if it is specified; otherwise, the value of *link-location*.

[*Example*:

`HYPERLINK("http://example.openxmlformats.org/report/budget report.xls","Click for report")`, which opens a worksheet named "budget report.xls" that is stored on the Internet at the location example.openxmlformats.org/report, and displays the text "Click for report".

`HYPERLINK("D:\FINANCE\1stqtr.xls",H10)`, which opens the file 1stqtr.xls that is stored in a directory named Finance on drive D, and displays the numeric value stored in cell `H10`.

end example]

3.17.7.145 HYPGEOMDIST

Syntax:

`HYPGEOMDIST` (*sample-successes* , *number-sample* , *population-successes* , *number-population*)

Description: Computes the hypergeometric distribution; that is, the probability of a given number of sample successes, given the sample size, population successes, and population size.

Mathematical Formula:

$$P(X = x) = h(x; n, M, N) = \frac{\binom{M}{x}\binom{N-M}{n-x}}{\binom{N}{n}}$$

where:

x = *sample-successes*
n = number-sample
M = *population-successes*
N = *number-population*

Arguments:

Name	Type	Description
sample-successes	number	The number of successes in the sample, truncated to integer.
number-sample	number	The size of the sample, truncated to integer.
population-successes	number	The number of successes in the population, truncated to integer.
number-population	number	The population size, truncated to integer.

Return Type and Value: number – The hypergeometric distribution.

However, if

- *sample-successes* < 0 or *sample-successes* is greater than the lesser of *number-sample* and *population-successes*, `#NUM!` is returned.
- *sample-successes* is less than the larger of 0 or (*number-sample* - *number-population* + *population-successes*), `#NUM!` is returned.
- *number-sample* ≤ 0 or *number-sample* > *number-population*, `#NUM!` is returned.
- *population-successes* ≤ 0 or *population-successes* > *number-population*, `#NUM!` is returned.
- *number-population* ≤ 0, `#NUM!` is returned.

[*Example*:

`HYPGEOMDIST(1,4,8,20)` results in `0.363261`

end example]

3.17.7.146 IF

Syntax:

IF (*logical-value* , [*value-if-true*][, [*value-if-false*]])

Description: Tests *logical-value*, and if it is TRUE, *value-if-true* is evaluated and returned; otherwise, *value-if-false* is evaluated and returned.

Arguments:

Name	Type	Description
logical-value	logical	The value to be tested.
value-if-true	any	The value returned if *logical-value* is TRUE. If *logical-value* is TRUE and *value-if-true* is omitted, this argument evaluates to 0. *value-if-true* can contain up to seven levels of nested IF function calls. *value-if-true* and *value-if-false* need not evaluate to results of the same type.
value-if-false	any	The value returned if *logical-value* is FALSE. If *logical-value* is FALSE and *value-if-false* and its preceding comma is omitted, this argument evaluates to FALSE. If *logical-value* is FALSE and *value-if-false* is omitted, but its preceding comma is present, this argument evaluates to 0. *value-if-false* can contain at least seven levels of nested IF function calls. *value-if-true* and *value-if-false* need not evaluate to results of the same type.

If any argument is an array, every element of that array shall be evaluated when that argument is evaluated.

Return Type and Value: any – *value-if-true*, if *logical-value* is TRUE; otherwise, *value-if-false*.

[*Example*:

IF(10>5,"Yes","No") results in Yes
IF(10>5,"Yes") results in Yes
IF(10>5,"Yes",) results in Yes
IF(10<5,"Yes") results in FALSE
IF(10<5,"Yes",) results in 0

IF(10>5,,"No") results in 0
IF(10>5,,) results in 0
IF(10>5,"Yes",20) results in Yes
IF(10<5,"Yes",20) results in 20

end example]

3.17.7.147 IFERROR

Syntax:

`IFERROR ( value , value-if-error )`

Description: Provides a simpler and more efficient way of trapping and handling errors. It allows the generation of user-defined error text for a function call that can result in an error.

Arguments:

Name	Type	Description
value	any	The value that is checked for an error (i.e., any of the following: #N/A, #VALUE!, #REF!, #DIV/0!, #NUM!, #NAME?, or #NULL!). If *value* is an empty cell, it is treated as an empty string.
value-if-error	any	The value to return if *value* evaluates to an error. If *value-if-error* is an empty cell, it is treated as an empty string.

Return Type and Value: any – *value*, if *value* is not an error; otherwise, *value-if-error*. If *value* is an array formula, an array of results for each cell in the range specified in value, is returned.

[*Example*: Consider the case in which A3 contains 55, and B3 contains 0:

A3/B3 results in #DIV/0

IFERROR(A3/B3,"Error in calculation") results in Error in calculation

end example]

3.17.7.148 IMABS

Syntax:

`IMABS ( complex-number )`

Description: Computes the absolute value of *complex-number*.

Mathematical Formula:

$$\mathrm{IMABS}(z) = |z| = \sqrt{x^2 + y^2}$$

where z = x + yi

Arguments:

Name	Type	Description

Name	Type	Description
complex-number	text	The complex number for which the absolute value is being computed. *complex-number* shall be in *x + yi* or *x + yj* text format.

Return Type and Value: number – The absolute value of *complex-number*.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMABS("3+4i")` results in `5`
`IMABS("-2.5-34.6j")` results in `34.69020035`

end example]

3.17.7.149 IMAGINARY

Syntax:

`IMAGINARY ( complex-number )`

Description: Computes the imaginary coefficient of *complex-number*.

Arguments:

Name	Type	Description
complex-number	text	The complex number for which the imaginary coefficient is being computed. *complex-number* shall be in *x + yi* or *x + yj* text format.

Return Type and Value: number – The imaginary coefficient of *complex-number*.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMAGINARY("3+4i")` results in `4`
`IMAGINARY("-2.5-34.6j")` results in `34.6`

end example]

3.17.7.150 IMARGUMENT

Syntax:

IMARGUMENT (*complex-number*)

Description: Computes the argument θ, an angle expressed in radians, such that for a complex number *complex-number* having the form *x+yi*:

$$x + yi = |x + yi| \times e^{\theta} = |x + yi|(\cos\theta + i\sin\theta)$$

Mathematical Formula:

$$\mathrm{IMARGUMENT}(z) = \tan^{-1}\left(\frac{y}{x}\right) = \theta$$

where:

$$\theta \in (-\pi; \pi]$$

and z = x + yi

Arguments:

Name	Type	Description
complex-number	text	The complex number in *x + yi* or *x + yj* text format.

Return Type and Value: number – The angle θ, expressed in radians.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMARGUMENT("13+4i")` results in `0.298498932`
`IMARGUMENT("-2.5-5j")` results in `-2.034443936`

end example]

3.17.7.151 IMCONJUGATE

Syntax:

IMCONJUGATE (*complex-number*)

Description: Computes the complex conjugate of the complex number *complex-number*.

Mathematical Formula:

$$\mathrm{IMCONJUGATE}(x + yi) = \bar{z} = (x - yi)$$

Arguments:

Name	Type	Description
complex-number	text	The complex number for which the complex conjugate is being computed. *complex-number* shall be in *x + yi* or *x + yj* text format.

Return Type and Value: text – A string containing the complex conjugate of *complex-number*, in *x+yi* or *x+yj* text format.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMCONJUGATE("2.3+4.5i")` results in `2.3-4.5i`
`IMCONJUGATE("-1-4j")` results in `-1+4j`

end example]

3.17.7.152 IMCOS

Syntax:

`IMCOS ( complex-number )`

Description: Computes the cosine of the complex number *complex-number*.

Mathematical Formula:

$$\cos(x + yi) = \cos(x)\cosh(y) - \sin(x)\sinh(y)i$$

Arguments:

Name	Type	Description
complex-number	text	The complex number for which the cosine is being computed. *complex-number* shall be in *x + yi* or *x + yj* text format.

Return Type and Value: text – A string containing the cosine of *complex-number*, in *x+yi* or *x+yj* text format.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMCOS("2.3+4.5i")` results in `-29.9918288739746-33.5589799796873i`
`IMCOS("-1-4j")` results in `14.7547011704838-22.963673499193j`

end example]

3.17.7.153 IMDIV

Syntax:

`IMDIV ( complex-number-1 , complex-number-2 )`

Description: Computes the quotient from dividing two complex numbers.

Mathematical Formula:

$$\mathrm{IMDIV}(z_1, z_2) = \frac{(a+bi)}{(c+di)} = \frac{(ac+bd)+(bc-ad)i}{c^2+d^2}$$

Arguments:

Name	Type	Description
complex-number-1	text	Complex numbers in *x + yi* or *x + yj* text format; they designate the dividend and divisor, respectively.
complex-number-2	text	

Return Type and Value: text – A string containing the quotient from *number-1 / number-2*, in *x+yi* or *x+yj* text format.

However, if *complex-number-1* or *complex-number-2* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMDIV("13+4i","5+3i")` results in `2.26470588235294-0.558823529411765i`
`IMDIV("-3-3.5i","5+3i")` results in `-0.75-0.25i`

end example]

3.17.7.154 IMEXP

Syntax:

`IMEXP ( complex-number )`

Description: Computes the exponential of the complex number *complex-number*.

Mathematical Formula:

$$\mathrm{IMEXP}(z) = e^{(x+yi)} = e^x e^{yi} = e^x(\cos y + i\sin y)$$

Arguments:

Name	Type	Description
complex-number	text	The complex number for which the exponential is being computed. *complex-number* shall be in *x + yi* or *x + yj* text format.

Return Type and Value: text – A string containing the exponential of *complex-number*, in *x+yi* or *x+yj* text format.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMEXP("2.3+4.5i")` results in `-2.10251576423113-9.75006374866818i`
`IMEXP("-1-4j")` results in `-0.240462049968584+0.278412079051034j`

end example]

3.17.7.155 IMLN

Syntax:

`IMLN ( complex-number )`

Description: Computes the natural logarithm of *complex-number*.

Mathematical Formula:

$$\ln(x + yi) = \ln\sqrt{x^2 + y^2} + i\tan^{-1}\left(\frac{y}{x}\right)$$

Arguments:

Name	Type	Description
complex-number	text	The complex number for which the natural logarithm is being computed. *complex-number* shall be in *x + yi* or *x + yj* text format.

Return Type and Value: text – The natural logarithm of *complex-number*, in *x+yi* or *x+yj* text format.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMLN("3+4i")` results in `1.6094379124341+0.927295218001612i`
`IMLN("-2.5-34.6j")` results in `3.54645723627033-1.64292531532225j`

end example]

3.17.7.156 IMLOG10

Syntax:

`IMLOG10 ( complex-number )`

Description: Computes the base-10 logarithm of *complex-number*.

Mathematical Formula:

The common logarithm of a complex number can be calculated from the natural logarithm as follows:

$$\log_{10}(x+yi) = (\log_{10} e)\ln(x+yi)$$

Arguments:

Name	Type	Description
complex-number	text	The complex number for which the base-10 logarithm is being computed. *complex-number* shall be in *x + yi* or *x + yj* text format.

Return Type and Value: text – The base-10 logarithm of *complex-number*, in *x+yi* or *x+yj* text format.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMLOG10("3+4i")` results in `10.698970004336019+0.402719196273373i`
`IMLOG10("-2.5-34.6j")` results in `11.54020680801806-0.713513398623614j`

end example]

3.17.7.157 IMLOG2

Syntax:

`IMLOG2 ( complex-number )`

Description: Computes the base-2 logarithm of *complex-number*.

Mathematical Formula:

The base-2 logarithm of a complex number can be calculated from the natural logarithm as follows:

$$\log_{2}(x+yi) = (\log_{2} e)\ln(x+yi)$$

Arguments:

Name	Type	Description
complex-number	text	The complex number for which the base-2 logarithm is being computed. *complex-number* shall be in *x + yi* or *x + yj* text format.

Return Type and Value: text – The base-2 logarithm of *complex-number*, in *x+yi* or *x+yj* text format.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

IMLOG2("3+4i") results in `2.32192809506607+1.33780421255394i`
IMLOG2("-2.5-34.6j") results in `5.11645626788577-2.37024020514877j`

end example]

3.17.7.158 IMPOWER

Syntax:

```
IMPOWER ( complex-number , y )
```

Description: Computes the complex number *complex-number* raised to the power *y*.

Mathematical Formula:

$$(x+yi)^n = r^n e^{i\theta n} = r^n \cos n\theta + ir^n \sin n\theta$$

where:

$$r = \sqrt{x^2 + y^2}$$

and:

$$\theta = \tan^{-1}\left(\frac{y}{x}\right)$$

and:

$$\theta \in (-\pi; \pi]$$

Arguments:

Name	Type	Description
complex-	text	The complex number in *x + yi* or *x + yj* text format.

Name	Type	Description
number		
y	number	The exponent to which *complex-number* is to be raised.

Return Type and Value: text – A string containing *complex-number*^{y}, in *x+yi* or *x+yj* text format.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMPOWER("2.3+4.5i",2.5)` results in `-52.9752689709953+22.138528463954i`
`IMPOWER("-1-4j",-3.56)` results in `6.34818926783845E-003+1.16156377299512E-003j`

end example]

3.17.7.159 IMPRODUCT

Syntax:

`IMPRODUCT ( argument-list )`

Description: Multiplies the values of its complex number arguments.

Mathematical Formula:

$$(a + bi)(c + di) = (ac - bd) + (ad + bc)i$$

Arguments:

Name	Type	Description
argument-list	text	Each *argument* in *argument-list* is a complex number string in *x + yi* or *x + yj* text format.

Return Type and Value: text – A string containing the product of the values of its arguments, in *x+yi* or *x+yj* text format.

However, if any *argument* in *argument-list* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMPRODUCT("13+4i")` results in `13+4i`
`IMPRODUCT("-3-3.5i","5+3i")` results in `-4.5-26.5i`
`IMPRODUCT("1.3-2j","-3.4+3j","2.3-6j")` results in `67.834+15.13j`

end example]

3.17.7.160 IMREAL

Syntax:

`IMREAL ( complex-number )`

Description: Computes the real coefficient of *complex-number*.

Arguments:

Name	Type	Description
complex-number	text	The complex number for which the real coefficient is being computed. *complex-number* shall be in *x + yi* or *x + yj* text format.

Return Type and Value: number – The real coefficient of *complex-number*.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMREAL("3+4i")` results in `3`
`IMREAL("-2.5-34.6j")` results in `-2.5`

end example]

3.17.7.161 IMSIN

Syntax:

`IMSIN ( complex-number )`

Description: Computes the sine of the complex number *complex-number*.

Mathematical Formula:

$$\sin(x + yi) = \sin(x)\cosh(y) - \cos(x)\sinh(y)i$$

Arguments:

Name	Type	Description
complex-number	text	The complex number for which the sine is being computed. *complex-number* shall be in *x + yi* or *x + yj* text format.

Return Type and Value: text – A string containing the sine of *complex-number*, in *x+yi* or *x+yj* text format.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMSIN("2.3+4.5i")` results in `33.567264016308-29.9844272159606i`
`IMSIN("-1-4j")` results in `-22.9790855778861-14.7448051885587j`

end example]

3.17.7.162 IMSQRT

Syntax:

`IMSQRT ( complex-number )`

Description: Computes the square root of the complex number *complex-number*.

Mathematical Formula:

$$\sqrt{x+yi} = \sqrt{r}\cos\left(\frac{\theta}{2}\right) + i\sqrt{r}\sin\left(\frac{\theta}{2}\right)$$

where:

$$r = \sqrt{x^2 + y^2}$$

and:

$$\theta = \tan^{-1}\left(\frac{y}{x}\right)$$

and:

$$\theta \in (-\pi; \pi]$$

Arguments:

Name	Type	Description
complex-number	text	The complex number for which the square root is being computed. *complex-number* shall be in *x + yi* or *x + yj* text format.

Return Type and Value: text – A string containing the square root of *complex-number*, in *x+yi* or *x+yj* text format.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMSQRT("2.3+4.5i")` results in `1.91751290835255+1.17339496918073i`
`IMSQRT("-1-4j")` results in `1.24962106768765-1.60048518044024j`

end example]

3.17.7.163 IMSUB

Syntax:

`IMSUB ( complex-number-1 , complex-number-2 )`

Description: Computes the difference of two complex numbers.

Mathematical Formula:

$$(a + bi) - (c + di) = (a - c) + (b - d)i$$

Arguments:

Name	Type	Description
complex-number-1	text	Complex numbers in *x + yi* or *x + yj* text format; they designate the minuend and subtrahend, respectively.
complex-number-2	text	

Return Type and Value: text – A string containing *number-1* - *number-2*, in *x+yi* or *x+yj* text format.

However, if *complex-number-1* or *complex-number-2* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMSUB("13+4i","5+3i")` results in `8+i`
`IMSUB("-3-3.5i","5+3i")` results in `-8-6.5i`

end example]

3.17.7.164 IMSUM

Syntax:

`IMSUM ( argument-list )`

Description: Adds the values of its arguments.

Mathematical Formula:

$(a+bi)+(c+di)=(a+c)+(b+d)i$

Arguments:

Name	Type	Description
argument-list	text	Each *argument* in *argument-list* is a complex number string in *x + yi* or *x + yj* text format.

Return Type and Value: text – The sum of the values of its arguments, in *x+yi* or *x+yj* text format.

However, if any *argument* in *argument-list* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMSUM("3+4i")` results in `3+4i`
`IMSUM("3+4i","5-3i")` results in `8+i`

end example]

3.17.7.165 INDEX

Syntax:

array form: `INDEX ( array , [ row-number ][ , [ column-number ]] )`

reference form: `INDEX ( reference [ , [ row-number ][ , [ column-number ]`
`[ , [ area-number ]]]] )`

Description: Locates a value or the reference to a value from within a table or range. There are two forms of the `INDEX` function: the array form and the reference form.

Arguments:

Name	Type	Description
array	array, reference	Table or range to be searched. If *array* contains only one row, the corresponding *row-number* argument is optional. If *array* contains only one column, the corresponding *column-number* argument is optional.
reference	reference	A reference to one or more cell ranges. If each area (§3.17.2.3) in *reference* contains only one row, *row-number* is optional. If each area contains only one column, *column-number* is optional.
row-number	number	*row-number* indicates the row in *array* (or *reference*) from which to return a value (or reference). If *row-number* is omitted, *column-number* shall be present. *column-number* indicates the column in *array* (or
column-number	number	

Name	Type	Description
		reference) from which to return a value (or *reference*). If *column-number* is omitted, *row-number* shall be present. If both the *row-number* and *column-number* arguments are used, `INDEX` returns the value (or *reference*) in the cell at the intersection of *row-number* and *column-number*. If *array* has more than one row and more than one column, and only *row-number* or *column-number* is used, `INDEX` returns an array of the entire row or column in array. If *row-number* or *column-number*, but not both, is 0, `INDEX` returns the array of values for the entire column or row, respectively. In the reference form, if *row-number* and *column-number* are both omitted, `INDEX` returns the area in reference specified by *area-number*.
area-number	number	Indicates a range in *reference* from which to return the intersection of *row-number* and *column-number*. The first area selected or entered is numbered 1, the second 2, and so on. If *area-number* is omitted, 1 is assumed. [*Example*: If *reference* describes the cells `(A1:B4, D1:E4, G1:H4)`, then area-number 1 is the range `A1:B4`, *area-number* 2 is the range `D1:E4`, and *area-number* 3 is the range `G1:H4`. *end example*]

Return Type and Value: various – For the array form, returns a single value, a whole row, or a whole column from a table or an array, depending on the presence and values of the row and column number indexes.

For the reference form, returns a single reference, a whole row, or a whole column from a reference, depending on the presence and values of the row and column number indexes, and the area number.

However, for the array form

- *row-number* is outside the bounds of *array*, `#REF!` is returned.
- *column-number* is outside the bounds of *array*, `#REF!` is returned.

For the reference form

- *row-number* is outside the bounds of *reference*, `#REF!` is returned.
- *column-number* is outside the bounds of *reference*, `#REF!` is returned.
- *area-number* is outside the bounds of *reference*, `#REF!` is returned.

[*Example*:

`INDEX({"Apples","Lemons";"Bananas","Pears"},2,2)` results in `Pears`
`INDEX({"Apples","Lemons";"Bananas","Pears"},2,1)` results in `Bananas`

`INDEX({"Apples","Lemons"},,2)` results in `Lemons`
`INDEX({"Apples";"Bananas"},1)` results in `Apples`

Given the following data:

	A	B	C
1	Fruit	Price	Count
2	Apples	0.69	40
3	Bananas	0.34	38
4	Lemons	0.55	15
5	Oranges	0.25	25
6	Pears	0.59	40
7	Almonds	2.8	10

`INDEX(A2:C7,2,3)` results in `38`
`INDEX((A2:C4,A6:C7),2,2,2)` results in `2.8`
`INDEX((A2:C4,A6:C7),2,2,1)` results in `0.34`

end example]

3.17.7.166 INDIRECT

Syntax:

`INDIRECT ( ` *ref-text* ` [ , [ ` *A1-ref-style-flag* ` ] ] )`

Description: Locates the reference specified by *ref-text* and evaluates that reference to get to its underlying value. [*Note*: This function should be used when the reference to a cell within a formula is to be changed without changing the formula itself. *end note*]

Arguments:

Name	Type	Description
ref-text	An A1-style reference, an R1C1-style reference, a name defined as a reference, or a reference to a cell as a string.	If *ref-text* refers to another workbook (i.e., it's an external reference), that other workbook shall be open.
A1-ref-style-flag	logical	Specifies the kind of reference that is contained in the cell *ref-text*. If TRUE or omitted, *ref-text* is interpreted as an A1-style reference (§3.17.2.3.1); otherwise, *ref-text* is

Name	Type	Description
		interpreted as an R1C1-style reference (§3.17.2.3.2).

Return Type and Value: any – The underlying value of the location referred to by *ref-text*.

However, if

- *ref-text* is not a valid cell reference, `#REF!` is returned.
- *ref-text* refers to another workbook yet that other workbook is not currently open, the return value is unspecified.

[*Example*:

Given the following data:

	A	B
1	Data	Data
2	B2	1.333
3	B3	45
4	George	10
5	5	62

where A2 contains a reference to B2, A3 contains a reference to B3, A4 contains the defined name George that refers to B4, and A5 contains the row number of B5:

`INDIRECT($A$2)` results in `1.333`
`INDIRECT($A$3)` results in `45`
`INDIRECT($A$4)` results in `10`
`INDIRECT("B"&$A$5)` results in `62`

`INDIRECT("R[-1]C",FALSE)` uses the cell in the previous row and current column.
end example]

3.17.7.167 INFO

Syntax:

```
INFO ( category )
```

Description: Retrieves the operating environment value that corresponds to *category*.

Arguments:

Name	Type	Description
category	text	The string designated by *category* is not case-sensitive. The valid strings are shown in the table below.

category	Meaning	Result Type
"directory"	Path of the current directory or folder.	text
"memavail"	Amount of memory available, in bytes.	number
"memused"	Amount of memory being used for data.	number
"numfile"	Number of active worksheets in the open workbooks.	number
"origin"	The absolute cell reference of the top and leftmost cell visible in the window, based on the current scrolling position, prefixed with "$A:". [*Example*: Using cell D9 as an example, the return value would be $A:$D$9. *end example*]	text
"osversion"	Current operating system version.	text
"recalc"	Current recalculation mode: "Automatic" or "Manual"	text
"release"	Version of the implementation.	text
"system"	Name of the operating environment.	text
"totmem"	Total memory available, including memory already in use, in bytes.	number

Return Type and Value: text – The operating environment value that corresponds to *category*.

However, if *category* is not one of the values defined above, #VALUE! is returned.

[*Example*:

INFO("directory") might result in e:\My Documents\
INFO(A10) might result in e:\My Documents\, where A10 contains directory
INFO("memavail") might result in 1048576

INFO("memused") might result in 1474464
INFO("numfile") might result in 5
INFO("origin") might result in $A:$C$536

INFO("osversion") might result in Windows (32-bit) NT 5.01
INFO("recalc") might result in Automatic
INFO("release") might result in 11.0

INFO("system") might result in pcdos
INFO("totmem") might result in 2523040

end example]

3.17.7.168 INT

Syntax:

```
INT ( x )
```

Description: Computes *x* rounded down to an integer.

Arguments:

Name	Type	Description
x	number	The value to be rounded down.

Return Type and Value: number – The rounded-down value of *x*.

[*Example*:

`INT(8.9)` results in `8`
`INT(-8.9)` results in `-9`

end example]

3.17.7.169 INTERCEPT

Syntax:

```
INTERCEPT ( known-ys , known-xs )
```

Description: Computes the point at which a line will intersect the y-axis by using existing x-values and y-values. The intercept point is based on a best-fit regression line plotted through the known x-values and known y-values.

Mathematical Formula:

The equation for the intercept of the regression line, a, is:

$$a = \bar{y} - b\bar{x}$$

where the slope, b, is calculated as:

$$b = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sum (x - \bar{x})^2}$$

and where x and y are the sample means AVERAGE(*known-xs*) and AVERAGE(*known-ys*).

Arguments:

Name	Type	Description
known-ys	number, name, array, reference to number	The dependent set of observations or data. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.
known-xs	number, name, array, reference to number	The independent set of observations or data. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The point at which a line will intersect the y-axis by using existing x-values and y-values.

However, if

- *known-ys* and *known-xs* contain a different number of data points, the return value is unspecified.
- *known-ys* or *known-xs* contain no data points, the return value is unspecified.

[*Example*:

`INTERCEPT({2,3,9,1,8},{6,5,11,7,5})` results in `0.048387097`

end example]

3.17.7.170 INTRATE

Syntax:

`INTRATE ( `*settlement*` , `*maturity*` , `*investment*` , `*redemption*` [ , [ `*basis*` ] ] )`

Description: Computes the interest rate for a fully invested security.

Mathematical Formula:

$$INTRATE = \frac{redemption - investment}{investment} \times \frac{B}{DIM}$$

where:

B = number of days in a year, depending on the year basis.
DIM = number of days from settlement to maturity.

Arguments:

Name	Type	Description

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
investment	number	The amount invested in the security.
redemption	number	The amount to be received at maturity.he security's annual yield.
basis	number	The truncated integer type of day count basis to use, as follows:

Value	Day Count Basis
0 or omitted	US (NASD) 30/360
1	Actual/actual
2	Actual/360
3	Actual/365
4	European 30/360

Time information in the date arguments is ignored.

Return Type and Value: number – The interest rate for a fully invested security.

However, if

- *settlement* or *maturity* is out of range for the current date base value, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *investment* or *redemption* ≤ 0, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`INTRATE(DATE(2008,2,15),DATE(2008,5,15),1000000,1014420,2)` results in `5.7680%`

end example]

3.17.7.171 IPMT

Syntax:

`IPMT ( rate , per, nper , pv , [ fv ] [ , [ type ]] )`

Description: Computes the interest payment for a given period for an investment based on periodic, constant payments and a constant interest rate.

Arguments:

Name	Type	Description
rate	number	The interest rate.
per	number	The period for which the interest is to be found, and shall be in the range 1–*nper*.
nper	number	The total number of payment periods in an annuity.
pv	number	The present value, or the lump-sum amount that a series of future payments is worth right now.
fv	number	The future value, or a cash balance to be attained after the last payment is made. If omitted, it is assumed to be 0 (i.e., the future value of a loan, for example, is 0).
type	number	The timing of the payment, truncated to integer, as follows: **Value** / **Timing** `0` – Payment at the end of the period `1` – Payment at the beginning of the period

Arguments representing cash paid by investor shall be expressed as negative numbers; arguments representing cash received by the investor shall be expressed as positive numbers.

Return Type and Value: number – The interest payment for a given period for an investment based on periodic, constant payments and a constant interest rate.

However, if *type* is any number other than 0 or 1, `#NUM!` is returned.

[*Example*:

`IPMT(0.1/12,1*3,3,8000)` results in `-22.41`
`IPMT(0.1,3,3,8000)` results in `-292.45`

end example]

3.17.7.172 IRR

`IRR ( values [ , [ guess ] ] )`

Description: Computes the internal rate of return for a series of cash flows represented by the numbers in *values*. (These cash flows do not have to be even, as they would be for an annuity. However, the cash flows shall occur at regular intervals, such as monthly or annually. The internal rate of return is the interest rate received for an investment consisting of payments (negative values) and income (positive values) that occur at regular periods.) `IRR` uses an iterative calculation technique.

Arguments:

Name	Type	Description
values	array, reference, text, logical	The set of numbers for which the internal rate of return is to be calculated. *values* shall contain at least one positive value and one negative value to calculate the internal rate of return. The order of numbers in *values* is significant, so be sure payment and income numbers are in the desired sequence. If *values* contains elements that are text, logical values, or empty cells, those elements are ignored.
guess	number	An estimate of the result of IRR. If omitted, it is assumed to be 0.1 (i.e., 10 percent).

Return Type and Value: number – The internal rate of return for a series of cash flows.

However, if the calculation has not converged after an implementation-defined number of iterations, #NUM! is returned.

[*Example*:

IRR({-70000,12000,15000,18000,21000}) results in -2.1245%
IRR({-70000,12000,15000,18000,21000,26000}) results in 8.6631%
IRR({-70000,12000,15000},-0.1) results in -44.3507%

end example]

3.17.7.173 ISBLANK

Syntax:

ISBLANK (*value*)

Description: Determines if *value* refers to an empty cell.

Arguments:

Name	Type	Description
value	any	The value to be tested. No conversion shall take place on an argument passed to this function.

Return Type and Value: logical – TRUE if *value* refers to an empty cell; otherwise, FALSE.

[*Example*:

`ISBLANK(A10)` results in `TRUE`, when A10 is empty
`ISBLANK(A10)` results in `FALSE`, when A10 contains 123

end example]

3.17.7.174 ISERR

Syntax:

`ISERR ( value )`

Description: Determines if *value* is any of the error values other than `#N/A`.

Arguments:

Name	Type	Description
value	any	The value to be tested. No conversion shall take place on an argument passed to this function.

Return Type and Value: logical – `TRUE` if *value* is one of the error values, excluding `#N/A`; otherwise, `FALSE`.

[*Example*:

`ISERR(A1)` results in `TRUE` if A1 evaluates to `#DIV/0!`, for example
`ISERR(B1)` results in `FALSE` if B1 evaluates to `#N/A`

end example]

3.17.7.175 ISERROR

Syntax:

`ISERROR ( value )`

Description: Determines if *value* is any of the error values.

Arguments:

Name	Type	Description
value	any	The value to be tested. No conversion shall take place on an argument passed to this function.

Return Type and Value: logical – `TRUE` if *value* is one of the error values; otherwise, `FALSE`.

[*Example*:

`ISERROR(A1)` results in TRUE if A1 evaluates to `#DIV/0!`, for example

end example]

3.17.7.176 ISEVEN

Syntax:

`ISEVEN ( value )`

Description: Determines if *value* is an even number or refers to a cell containing an even number.

Arguments:

Name	Type	Description
value	number	The value to be tested. It is truncated to an integer.

Return Type and Value: logical – TRUE if *value* is an even number or refers to a cell containing an even number; otherwise, FALSE.

[*Example*:

`ISEVEN(12.456)` results in TRUE
`ISEVEN(A10)` results in FALSE, when A10 contains -15

end example]

3.17.7.177 ISLOGICAL

Syntax:

`ISLOGICAL ( value )`

Description: Determines if *value* contains a logical value or refers to a cell containing a logical value.

Arguments:

Name	Type	Description
value	any	The value to be tested. No conversion shall take place on an argument passed to this function.

Return Type and Value: logical – TRUE if *value* contains a logical value or refers to a cell containing a logical value; otherwise, FALSE.

[*Example*:

ISLOGICAL(TRUE) results in TRUE
ISLOGICAL(A10) results in FALSE, when A10 contains 123
ISLOGICAL({TRUE,2}) results in TRUE
ISLOGICAL({2,TRUE}) results in FALSE

end example]

3.17.7.178 ISNA

Syntax:

ISNA (*value*)

Description: Determines if *value* is the error value #N/A.

Arguments:

Name	Type	Description
value	any	The value to be tested. No conversion shall take place on an argument passed to this function.

Return Type and Value: logical – TRUE if *value* is #N/A; otherwise, FALSE.

[*Example*:

ISERR(A1) results in TRUE if A1 evaluates to #N/A
ISERR(B1) results in TRUE if B1 evaluates to #DIV/0!, for example

end example]

3.17.7.179 ISNONTEXT

Syntax:

ISNONTEXT (*value*)

Description: Determines if *value* does not contain text or does not refer to a cell containing text. An empty cell is not text.

Arguments:

Name	Type	Description
value	any	The value to be tested. No conversion shall take place on an argument passed to this function.

Return Type and Value: logical – TRUE if *value* does not contain text or does not refer to a cell containing text; otherwise, FALSE.

[*Example*:

ISNONTEXT("ABC") results in FALSE
ISNONTEXT(A10) results in TRUE, when A10 contains 123
ISNONTEXT({1,"ABC"}) results in TRUE
ISNONTEXT({"ABC",1}) results in FALSE

end example]

3.17.7.180 ISNUMBER

Syntax:

ISNUMBER (*value*)

Description: Determines if *value* contains a number or refers to a cell that contains a number.

Arguments:

Name	Type	Description
value	any	The value to be tested. No conversion shall take place on an argument passed to this function.

Return Type and Value: logical – TRUE if *value* contains a number or refers to a cell that contains a number; otherwise, FALSE.

[*Example*:

ISNUMBER(10.56) results in TRUE
ISNUMBER(A10) results in FALSE, when A10 contains ABC
ISNUMBER({1,"ABC"}) results in TRUE
ISNUMBER({"ABC",1}) results in FALSE

end example]

3.17.7.181 ISODD

Syntax:

ISODD (*value*)

Description: Determines if *value* is an odd number or refers to a cell containing an odd number.

Arguments:

Name	Type	Description
value	number	The value to be tested. It is truncated to an integer.

Return Type and Value: logical – TRUE if *value* is an odd number or refers to a cell containing an odd number; otherwise, FALSE.

[*Example*:

`ISODD(12.456)` results in FALSE
`ISODD(A10)` results in TRUE, when A10 contains `-15`

end example]

3.17.7.182 ISPMT

Syntax:

`ISPMT ( rate , per, nper , pv )`

Description: Computes the interest paid during a specific period of an investment.

Arguments:

Name	Type	Description
rate	number	The interest rate for the investment.
per	number	The period for which the interest is to be found, and shall be in the range 1–*nper*.
nper	number	The total number of payment periods for the investment.
pv	number	The present value the investment.

Arguments representing cash paid by investor shall be expressed as negative numbers; arguments representing cash received by the investor shall be expressed as positive numbers.

Return Type and Value: number – The interest paid during a specific period of an investment.

[*Example*:

`ISPMT(0.1/12,1,3*12,8000000)` results in `-64814.81`
`ISPMT(0.1,1,3,8000000)` results in `-533333.33`

end example]

3.17.7.183 ISREF

Syntax:

`ISREF ( value )`

Description: Determines if *value* is a cell reference.

Arguments:

Name	Type	Description
value	any	The value to be tested. No conversion shall take place on an argument passed to this function.

Return Type and Value: logical – TRUE if *value* is a cell reference; otherwise, FALSE.

[*Example*:

`ISREF("ABC")` results in FALSE
`ISREF(A10)` results in TRUE

end example]

3.17.7.184 ISTEXT

Syntax:

`ISTEXT ( value )`

Description: Determines if *value* contains text or refers to a cell containing text.

Arguments:

Name	Type	Description
value	any	The value to be tested. No conversion shall take place on an argument passed to this function.

Return Type and Value: logical – TRUE if *value* contains text or refers to a cell containing text; otherwise, FALSE.

[*Example*:

`ISTEXT("ABC")` results in TRUE
`ISTEXT(A10)` results in FALSE, when A10 contains 123
`ISTEXT({1,"ABC"})` results in FALSE
`ISTEXT({"ABC",1})` results in TRUE

end example]

3.17.7.185 JIS

Syntax:

JIS (*string*)

Description: Creates a string that is the conversion of half-width (single-byte) letters within *string* to full-width (double-byte) characters.

Arguments:

Name	Type	Description
string	text	Designates the string to be converted. If *string* does not contain any half-width English letters or katakana, nothing in *string* is converted.

Return Type and Value: text – The string resulting from the conversion.

[*Example*:

JIS("ABC") results in ABC
JIS("ｴｸｾﾙ") results in エクセル

end example]

3.17.7.186 KURT

Syntax:

KURT (*argument-list*)

Description: Computes the kurtosis of a data set. Kurtosis characterizes the relative peakedness or flatness of a distribution compared with the normal distribution. Positive kurtosis indicates a relatively peaked distribution. Negative kurtosis indicates a relatively flat distribution.

Mathematical Formula:

Kurtosis is defined as:

$$\left\{\frac{n(n+1)}{(n-1)(n-2)(n-3)}\sum\left(\frac{x_j-\bar{x}}{s}\right)^4\right\} - \frac{3(n-1)^2}{(n-2)(n-3)}$$

where s is the sample standard deviation.

Arguments:

Name	Type	Description
argument-list	array reference to an array, number, name, or reference to number.	The *argument*s in *argument-list* are the values for which kurtosis is to be calculated. Any *argument* in *argument-list* can be an array or a reference to an array. Logical values and text representations of numbers that are directly entered into the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The kurtosis of a data set.

However, if

- There are fewer than four data points, the return value is unspecified.
- The standard deviation of the sample equals zero, the return value is unspecified.

[*Example*:

KURT(10.5,12.4,19.4,23.2) results in -3.644621343
KURT(10.5,{12.4,19.4},23.2) results in -3.644621343

end example]

3.17.7.187 LARGE

Syntax:

LARGE (*array* , *k*)

Description: Computes the k^{th} largest value in a data set.

Arguments:

Name	Type	Description
array	array, reference	The set of numbers from which the k^{th}-largest value is to be determined.
k	number	The position (from the largest) in the array or cell range of data to return.

Return Type and Value: number – The k^{th} largest value in a data set.

However, if

- *array* is empty, the return value is unspecified.
- $k \leq 0$, `#NUM!` is returned.
- *k* is greater than the number of data points, `#NUM!` is returned.

[*Example*:

`LARGE({3,5,3,5,4;4,2,4,6,7},3)` results in `5`
`LARGE({3,5,3,5,4;4,2,4,6,7},7)` results in `4`

end example]

3.17.7.188 LCM

Syntax:

`LCM` (*argument-list*)

Description: Computes the least common multiple of the one or more *argument*s in *argument-list*.

Arguments:

Name	Type	Description
argument-list	number	*argument-list* specifies the *argument*s. Each argument is truncated to an integer.

Return Type and Value: number – The least common multiple of one or more numbers.

However, if any *argument* is negative, `#NUM!` is returned.

[*Example*:

`LCM(5)` results in `5`
`LCM(5,2)` results in `10`
`LCM(24.99,36.45)` results in `72`
`LCM(24,36,15)` results in `360`

end example]

3.17.7.189 LEFT

Syntax:

`LEFT` (*string* [, *number-chars*])

Description: Extracts the left-most *number-chars* characters from *string*. (LEFT is intended for use with languages that use the single-byte character set (SBCS), whereas LEFTB (§3.17.7.190) is intended for use with languages that use the double-byte character set (DBCS).)

Arguments:

Name	Type	Description
string	text	Designate the string from which a substring is to be extracted.
number-chars	number	The number of characters to be extracted. If omitted, a count of 1 shall be assumed. *number-chars* shall be at least 0. If *number-chars* exceeds the length of *string*, the whole of *string* shall be extracted.

Return Type and Value: text – A string containing the left-most *number-chars* characters from *string*.

However, if *number-chars* is negative, #VALUE! is returned.

[*Example*:

`LEFT("abcdef",2)` results in `ab`
`LEFT(A10,4)` results in `xyz1`, when A10 contains `xyz123`

end example]

3.17.7.190 LEFTB

Syntax:

`LEFTB ( string [ , number-bytes ] )`

Description: Extracts the left-most *number-bytes*-worth of characters from *string*. (LEFTB is intended for use with languages that use the double-byte character set (DBCS), whereas LEFT (§3.17.7.190) is intended for use with languages that use the single-byte character set (SBCS).)

Arguments:

Name	Type	Description
string	text	Designate the string from which a substring is to be extracted.
number-bytes	number	The number of bytes to be extracted. If omitted, a count of 1 shall be assumed. *number-bytes* shall be at least 0. If *number-bytes* exceeds the length of *string*, the whole of *string* shall be extracted.

Return Type and Value: text – A string containing the left-most *number-bytes*-worth of characters from *string*.

However, if *number-bytes* is negative, #VALUE! is returned.

[*Example*: Assuming 1-byte characters:

`LEFTB("abcdef",2)` results in `ab`
`LEFTB(A10,4)` results in `xyz1`, when A10 contains `xyz123`

end example]

3.17.7.191 LEN

Syntax:

`LEN ( string )`

Description: Determines the number of characters in *string*. (`LEN` is intended for use with languages that use the single-byte character set (SBCS), whereas `LENB` (§3.17.7.192) is intended for use with languages that use the double-byte character set (DBCS).)

Arguments:

Name	Type	Description
string	text	Designates the string whose length is to be found.

Return Type and Value: number – The number of characters in *string*.

[*Example*:

`LEN("abc")` results in `3`
`LEN(A10)` results in `3`, when A1 contains `abc`

end example]

3.17.7.192 LENB

Syntax:

`LENB ( string )`

Description: Determines the number of bytes in *string*. (`LENB` is intended for use with languages that use the double-byte character set (DBCS), whereas `LEN` (§3.17.7.191) is intended for use with languages that use the single-byte character set (SBCS).)

Arguments:

Name	Type	Description
string	text	Designates the string whose length is to be found.

Return Type and Value: number – The number of bytes in *string*.

[*Example*: Assuming 1-byte characters:

`LENB("abc")` results in 3
`LENB(A10)` results in 3, when A1 contains `abc`

end example]

3.17.7.193 LINEST

Syntax:

`LINEST ( known-ys [ , [ known-xs ] [ , [ const-flag ] [ , stats-flag ] ] )`

Description: Calculates the statistics for a line by using the "least squares" method to calculate a straight line that best fits the data, and returns an array that describes the line.

Mathematical Formula:

The equation for the line is:

y = mx + b

or

y = m1x1 + m2x2 + ... + b (if there are multiple ranges of x-values)

where the dependent y-value is a function of the independent x-values. The m-values are coefficients corresponding to each x-value, and b is a constant value. y, x, and m can be vectors.

When there is only one independent x-variable, the slope and y-intercept values can be obtained directly by using the following formulas:

Slope: `INDEX(LINEST(`*known-ys*`,`*known-xs*`),1)`

Y-intercept: `INDEX(LINEST(`*known-ys*`,`*known-xs*`),2)`

The accuracy of the line calculated by `LINEST` depends on the degree of scatter in the data. The more linear the data, the more accurate the `LINEST` model. `LINEST` uses the method of least squares for determining the best fit for the data. When there is only one independent x-variable, the calculations for m and b are based on the following formulas:

$$m = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sum (x - \bar{x})^2}$$

$$b = \bar{y} - m\bar{x}$$

where x and y are sample means, i.e., x = AVERAGE(*known-xs*) and y = AVERAGE(*known-ys*).

Arguments:

Name	Type	Description
known-ys	array	The set of y-values already known in the relationship y=mx+b. If the array *known-ys* is a single column, then each column of *known-xs* is interpreted as a separate variable. If the array *known-ys* is a single row, then each row of *known-xs* is interpreted as a separate variable.
known-xs	array	An optional set of x-values that might already be known in the relationship y=mx+b. The array *known-xs* can include one or more sets of variables. If only one variable is used, *known-ys* and *known-xs* can be ranges of any shape, as long as they have equal dimensions. If more than one variable is used, *known-ys* shall be a vector (that is, a range with a height of one row or a width of one column). If *known-xs* is omitted, it is assumed to be the array {1,2,3,...} that is the same size as *known-ys*.
const-flag	logical	Specifies whether to force the constant *b* to be zero. If TRUE or omitted, *b* is calculated normally. If FALSE, *b* is set to zero, and the m-values are adjusted to fit y=mx.
stats-flag	logical	Specifies whether to return additional regression statistics. If TRUE, LINEST returns the additional regression statistics (see table below), so the returned array is {mn, mn-1, ..., m1, b; sen, sen-1, ..., se1, seb; r2, sey; F, df; ssreg, ssresid}. If FALSE or omitted, LINEST returns only the m-coefficients and the constant b.

The additional regression statistics are as follows:

Statistic	Description
se1, se2, ..., se*n*	The standard error values for the coefficients m1, m2, ..., m*n*.
seb	The standard error value for the constant b.
r2	The coefficient of determination.
sey	The standard error for the y estimate.
F	The F statistic, or the F-observed value.

Statistic	Description
df	The degrees of freedom.
ssreg	The regression sum of squares.
ssresid	The residual sum of squares.

Return Type and Value: array – The array that describes the line, in the form {mn, mn-1, ..., m1, b}. The following illustration shows the order in which the additional regression statistics are returned.

[*Example*:

`LINEST({1,9,5,7},{0,4,2,3},,FALSE)` results in a slope of 2 and a y-intercept of 1

end example]

3.17.7.194 LN

Syntax:

```
LN ( x )
```

Description: Computes the natural logarithm of *x*.

Arguments:

Name	Type	Description
x	number	The positive real number for which the natural logarithm is being computed.

Return Type and Value: number – The natural logarithm of *x*.

However, if *x* is zero or negative, `#NUM!` is returned.

[*Example*:

`LN(86)` results in `4.454347296`
`LN(2.7182818)` results in `0.99999999`
`LN(EXP(3))` results in 3

end example]

3.17.7.195 LOG

Syntax:

```
LOG ( x [ , base ] )
```

Description: Computes the logarithm of *x* to the base *base*.

Arguments:

Name	Type	Description
x	number	The positive real number for which the logarithm is being computed.
base	number	The base of the logarithm. If omitted, base 10 is assumed.

Return Type and Value: number – The logarithm of *x*.

However, if

- *x* is zero or negative, `#NUM!` is returned.
- *base* is zero or negative, `#NUM!` is returned.

[*Example*:

`LOG(10)` results in `1`
`LOG(8,2)` results in 3
`LOG(86,2.7182818)` results in `4.454347343`

end example]

3.17.7.196 LOG10

Syntax:

```
LOG10 ( x )
```

Description: Computes the base-10 logarithm of *x*.

Arguments:

Name	Type	Description
x	number	The positive real number for which the logarithm is being computed.

Return Type and Value: number – The base-10 logarithm of *x*.

However, if *x* is zero or negative, `#NUM!` is returned.

[*Example*:

`LOG10(86)` results in `1.934498451`

`LOG10(10)` results in 1
`LOG10(1E5)` results in 5
`LOG10(10^5)` results in 5

end example]

3.17.7.197 LOGEST

Syntax:

`LOGEST ( known-ys [ , [ known-xs ] [ , [ const-flag ] [ , stats-flag ] ] )`

Description: Calculates an exponential curve that fits the data, and returns an array of values that describes the curve.

Mathematical Formula:

The equation for the curve is:

y = b*mx

or

y = (b*(m1x1)*(m2x2)*…) (if there are multiple x-values)

where the dependent y-value is a function of the independent x-values. The m-values are bases corresponding to each exponent x-value, and b is a constant value. Note that y, x, and m can be vectors.

When there is only one independent x-variable, the y-intercept (b) values can be obtained directly by using the following formula:

Y-intercept (b): `INDEX(LOGEST(`*known-ys*`,`*known-xs*`),2)`

Arguments:

Name	Type	Description
known-ys	array	The set of y-values already known in the relationship y=b*m^x. If the array *known-ys* is a single column, then each column of *known-xs* is interpreted as a separate variable. If the array *known-ys* is a single row, then each row of *known-xs* is interpreted as a separate variable.
known-xs	array	An optional set of x-values that might already be known in the relationship y=b*m^x. The array *known-xs* can include one or more sets of variables. If only one variable is used, *known-ys* and *known-xs* can be ranges of any shape, as long as they have equal dimensions. If more than one variable is used, *known-ys* shall be a vector (that

Name	Type	Description
		is, a range with a height of one row or a width of one column). If *known-xs* is omitted, it is assumed to be the array {1,2,3,...} that is the same size as *known-ys*.
const-flag	logical	Specifies whether to force the constant *b* to be 1. If TRUE or omitted, *b* is calculated normally. If FALSE, *b* is set to 1, and the m-values are adjusted to fit $y=m^x$.
stats-flag	logical	Specifies whether to return additional regression statistics. If TRUE, LOGEST returns the additional regression statistics, so the returned array is {mn, mn-1, ..., m1, b; sen, sen-1, ..., se1, seb; r2, sey; F, df; ssreg, ssresid}. If FALSE or omitted, LOGEST returns only the m-coefficients and the constant b.

The additional regression statistics are described in §3.17.7.193.

Return Type and Value: array – The array that describes the line, in the form {mn, mn-1, ..., m1, b}. The order in which the additional regression statistics are returned is described in §3.17.7.193.

[*Example*: Given the following data:

	A	B
1	Month	Units
2	11	33,100
3	12	47,300
4	13	69,000
5	14	102,000
6	15	150,000
7	16	220,000
8	Formula	
9	1.463275628	495.3047702

When `LOGEST(B2:B7,A2:A7,TRUE,FALSE)` is array-entered into cells A9:B9, those cells take on the results shown.

end example]

3.17.7.198 LOGINV

Syntax:

`LOGINV ( probability , mean , standard-dev )`

Description: Calculates the inverse of the lognormal cumulative distribution function of x, where ln(x) is normally distributed with parameters *mean* and *standard-dev*.

Mathematical Formula:

$$LOGINV(p,\mu,\sigma) = e^{[\mu+\sigma\times(NORMSINV(p))]}$$

Arguments:

Name	Type	Description
probability	number	A probability associated with the lognormal distribution.
mean	number	The mean of ln(x).
standard-dev	number	The standard deviation of ln(x).

Return Type and Value: number – The inverse of the lognormal cumulative distribution function of x.

However, if

- *probability* < 0 or *probability* > 1, `#NUM!` is returned.
- *standard-dev* ≤ 0, `#NUM!` is returned.

[*Example*:

`LOGINV(0.039084,3.5,1.2)` results in `4.000025219`

end example]

3.17.7.199 LOGNORMDIST

Syntax:

`LOGNORMDIST ( x , mean , standard-dev )`

Description: Calculates the cumulative lognormal distribution of x, where ln(x) is normally distributed with parameters *mean* and *standard-dev*.

Mathematical Formula:

$$LOGNORMDIST(x,\mu,\sigma) = NORMSDIST\left(\frac{\ln(x)-\mu}{\sigma}\right)$$

Arguments:

Name	Type	Description
x	number	The value at which to evaluate the function.

Name	Type	Description
mean	number	The mean of ln(x).
standard-dev	number	The standard deviation of ln(x).

Return Type and Value: number – The inverse of the lognormal cumulative distribution function of x.

However, if

- *x* ≤ 0, `#NUM!` is returned.
- *standard-dev* ≤ 0, `#NUM!` is returned.

[*Example*:

`LOGNORMDIST(4,3.5,1.2)` results in `0.039083556`

end example]

3.17.7.200 LOOKUP

Syntax:

vector form: `LOOKUP` (*lookup-value* , *lookup-vector* , *result-vector*)

array form: `LOOKUP` (*lookup-value* , *array*)

Description: The vector form looks in a vector for a value, and returns a value from the same position in a second vector. The array form looks in the first row or column of an array for the specified value and returns a value from the same position in the last row or column of that array.

Arguments:

Name	Type	Description
lookup-value	number, string, logical, name, reference	The value to search for in *lookup-vector* (or *array*).
lookup-vector	reference	A range that contains only one row or one column. The values in *lookup-vector* can be strings, numbers, or logical values. These values shall be placed in "ascending" order, as follows: …, -2, -1, 0, 1, 2, …, A–Z, FALSE, TRUE. Upper- and lowercase strings are equivalent. If LOOKUP can't find the *lookup-value*, it matches the largest value in *lookup-vector* (or *array*) that is less than or equal to *lookup-value*.
result-vector	reference	A range that contains only one row or column. It shall be the same size as *lookup-vector*.

Name	Type	Description
array	text, number, logical	A range of cells whose values are to be compared with *lookup-value*. These values shall be placed in "ascending" order, as follows: …, -2, -1, 0, 1, 2, …, A–Z, FALSE, TRUE. Upper- and lowercase strings are equivalent. If *array* covers an area that has more columns than rows, *lookup-value* is searched for in the first row. If *array* is square or has more rows than columns, *lookup-value* is searched for in the first column.

Return Type and Value: any – The vector form looks in a vector for a value, and returns a value from the same position in a second vector. The array form looks in the first row or column of an array for the specified value and returns a value from the same position in the last row or column of that array.

However, if

- *lookup-value* is smaller than the smallest value in *lookup-vector* (or the first row or column of *array*), the return value is unspecified.
- The size of the range specified by *result-vector* is not the same as that specified by *lookup-vector*, the return value is unspecified.
- The values in *lookup-vector* (or *array*) are not in "ascending" order, the return value is unspecified.

[*Example*: Given the following data:

	A	B
1	Frequency	Color
2	4.14	red
3	4.19	orange
4	5.17	yellow
5	5.77	green
6	6.39	blue

`LOOKUP(4.19,A2:A6,B2:B6)` results in `orange`
`LOOKUP(5,A2:A6,B2:B6)` results in `orange`
`LOOKUP(7.66,A2:A6,B2:B6)` results in `blue`

`LOOKUP("C",{"a","b","c","d";1,2,3,4})` results in 3
`LOOKUP("bump",{"a",1;"b",2;"c",3})` results in 2

end example]

3.17.7.201 LOWER

Syntax:

LOWER (*string*)

Description: Makes a lowercase version of *string*.

Arguments:

Name	Type	Description
string	text	Designates the string to be converted.

Return Type and Value: text – The lowercase version of *string*.

[*Example*:

LOWER("AbCd123#$%^") results in abcd123#$%^
LOWER(A10) results in 234frtqwc$#%, when A10 contains 234FRTqwc$#%

end example]

3.17.7.202 MATCH

Syntax:

MATCH (*lookup-value* , *lookup-array* [, [*match-type*]])

Description: Locates the relative position of an array item that matches a specified value in a specified order. MATCH shall not distinguish between uppercase and lowercase letters when matching strings.

Arguments:

Name	Type	Description
lookup-value	number, string, logical, name, reference	The value to search for in *lookup-array*. If *match-type* is 0 and *lookup-value* is a string, the wildcard characters, question mark (?) and asterisk (*), can be used in *lookup-value*. A question mark matches any single character; an asterisk matches any sequence of characters. To locate a question mark or asterisk, precede that character with a tilde (~).
lookup-array	array, reference	A contiguous range of cells containing possible lookup values.
match-type	number	Specifies how *lookup-value* is matched with values in *lookup-array*, as follows:

Name	Type	Description
		<table><tr><th>Value</th><th>Meaning</th></tr><tr><td>-1</td><td>Finds the smallest value that is greater than or equal to *lookup-value*. The values in *lookup-array* shall be placed in "descending" order: TRUE, FALSE, Z–A, ..., 2, 1, 0, -1, -2, ...</td></tr><tr><td>0</td><td>Finds the first value that is exactly equal to *lookup-value*. The values in *lookup-array* can be in any order.</td></tr><tr><td>1 or omitted</td><td>Finds the largest value that is less than or equal to *lookup-value*. The values in *lookup-array* shall be placed in "ascending" order: ..., -2, -1, 0, 1, 2, ..., A–Z, FALSE, TRUE.</td></tr></table>

Return Type and Value: number – The relative position of an array item that matches a specified value in a specified order.

However, if

- No match is found, `#NUM!` is returned.
- *match-type*'s value is out-of-bounds, `#NUM!` is returned.

[*Example*:

`MATCH(39,{25,38,40,41},1)` results in 2
`MATCH(41,{25,38,40,41},0)` results in 4

end example]

3.17.7.203 MAX

Syntax:

`MAX` (*argument-list*)

Description: Computes the largest of a set of numbers.

Arguments:

Name	Type	Description

Name	Type	Description
argument-list	logical, number, name, arrays, reference to number. Any *argument* can be an array or a reference to an array.	The *argument*s in *argument-list* designate the values for which the largest value is to be computed. Logical values and text representations of numbers occurring directly in the list of arguments are included. However, logical values and numbers in strings and are ignored inside references. [*Note*: To include these, use MAXA (§3.17.7.204). *end note*] If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The largest of a set of numbers; however, if the arguments contain no numbers, zero is returned.

[*Example*:

`MAX(10.4,-3.5,12.6)` results in `12.6`
`MAX(10.4,{-3.5,12.6})` results in `12.6`
`MAX({"ABC",TRUE})` results in `0`

Consider the case in which cell `B3` contains `0`:
`MAX(-10,-12,-15,B3)` results in `-10`
`MAXA(-10,-12,-15,B3)` results in `0`

end example]

3.17.7.204 MAXA

Syntax:

MAXA (*argument-list*)

Description: Computes the largest of a set of numbers.

Arguments:

Name	Type	Description
argument-list	logical, number, name, arrays, reference to number. Any *argument* can be an array or a reference to an array.	The *argument*s in *argument-list* designate the values for which the largest value is to be computed. Logical values and text representations of numbers occurring directly in the list of arguments are included. Logical values and numbers in strings inside references are also included. [*Note*: To ignore these, use MAX (§3.17.7.203). *end note*] If an array or reference argument contains non-numeric text or empty cells, those values are ignored; however,

Name	Type	Description
		cells with the value 0 are included.

Return Type and Value: number – The largest of a set of numbers; however, if the arguments contain no numbers, zero is returned.

[*Example*:

`MAXA(10.4,-3.5,12.6)` results in `12.6`
`MAXA(10.4,{-3.5,12.6})` results in `12.6`
`MAXA({"ABC",TRUE})` results in `0`

Consider the case in which cell `B3` contains `0`:
`MAX(-10,-12,-15,B3)` results in `-10`
`MAXA(-10,-12,-15,B3)` results in `0`

end example]

3.17.7.205 MDETERM

Syntax:

```
MDETERM ( array )
```

Description: Computes the determinant of the square matrix of numbers designated by *array*. The determinant is calculated with an accuracy of at least 15 digits, which can lead to a small numeric error when the calculation is not complete. [*Example*: The determinant of a singular matrix can differ from zero by 1E-16. *end example*]

Arguments:

Name	Type	Description
array	array, reference	Designate a square matrix of numbers.

Return Type and Value: number – The determinant of *array*. Some square matrices cannot be inverted. The determinant of a non-invertible matrix is 0.

However, if

- Any cells in *array* are empty or contain text, the return value is unspecified.
- The matrix designated by *array* is not square, `#VALUE!` is returned.

[*Example*:

`MDETERM(A2:D5)` results in the determinant of the 4x4 array designated by the cell range

`MDETERM({3,6,1;1,1,0;3,10,2})` results in `1`
`MDETERM({3,6;1,1})` results in `-3`

end example]

3.17.7.206 MDURATION

Syntax:

`MDURATION ( `*settlement*` , `*maturity*` , `*coupon*` , `*yld*` , `*frequency*` [ , [ `*basis*` ] ] )`

Description: Computes the modified Macauley duration for a security with an assumed par value of $100.

Mathematical Formula:

$$\text{MDURATION} = \frac{\text{DURATION}}{1+\left(\frac{\text{Market yield}}{\text{Coupon payments per year}}\right)}$$

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
coupon	number	The security's annual coupon rate.
yld	number	The security's annual yield.
frequency	number	the number of coupon payments per year. (For annual payments, *frequency* is 1; for semiannual payments, *frequency* is 2; for quarterly payments, *frequency* is 4.) *frequency* is truncated to an integer.
basis	number	The truncated integer type of day count basis to use, as follows: **Value** / **Day Count Basis** `0` or omitted / US (NASD) 30/360 1 / Actual/actual 2 / Actual/360 3 / Actual/365 4 / European 30/360

Time information in the date arguments is ignored.

Return Type and Value: number – The modified Macauley duration for a security with an assumed par value of $100.

However, if

- *settlement* or *maturity* is out of range for the current date base value, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *coupon* or *yld* < 0, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`MDURATION(DATE(2008,1,1),DATE(2016,1,1),0.08,0.09,2,1)` results in `5.7357`

end example]

3.17.7.207 MEDIAN

Syntax:

`MEDIAN (` *argument-list* `)`

Description: Computes the median of the numeric values of its arguments.

Arguments:

Name	Type	Description
argument-list	logical, number, name, arrays, reference to number.	The *argument*s in *argument-list* designate the values whose median is to be computed. If there is an even number of numbers in the set, `MEDIAN` calculates the average of the two numbers in the middle. Logical values and text representations of numbers entered directly into the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The median of the values of its arguments.

[*Example*:

`MEDIAN(10,20)` results in `15`
`MEDIAN(-3.5,1.4,6.9,-4.5)` results in `-1.05`
`MEDIAN({-3.5,1.4,6.9},-4.5)` results in `-1.05`

end example]

3.17.7.208 MID

Syntax:

`MID ( string , start-pos , number-chars )`

Description: Extracts *number-chars* characters from *string*, starting at character position *start-pos*. (`MID` is intended for use with languages that use the single-byte character set (SBCS), whereas `MIDB` (§3.17.7.209) is intended for use with languages that use the double-byte character set (DBCS).)

Arguments:

Name	Type	Description
string	text	Designate the string from which a substring is to be extracted.
number-chars	number	The number of characters to be extracted. *number-chars* shall be at least 0.
start-pos	number	The starting position within *string*, where the first character is position 1. If *start-pos* is greater than the length of *string*, or if *start-pos* and *number-chars* combined exceeds the length of *string*, the whole of *string* shall be extracted.

Return Type and Value: text – A string containing *number-chars* characters from *string*, starting at character position *start-pos*.

However, if

- *start-pos* < 0, `#VALUE!` is returned.
- *number-chars* < 0, `#VALUE!` is returned.

[*Example*:

`MID("abcdef",3,2)` results in `cd`
`MID(A10,4,1)` results in `1`, when A10 contains `xyz123`
`MID("abcdef",4,5)` results in `def`

end example]

3.17.7.209 MIDB

Syntax:

MIDB (*string* , *start-pos* , *number-bytes*)

Description: Extracts *number-bytes*-worth of characters from *string*, starting at character position *start-pos*. (MIDB is intended for use with languages that use the double-byte character set (DBCS), whereas MID (§3.17.7.208) is intended for use with languages that use the single-byte character set (SBCS).)

Arguments:

Name	Type	Description
string	text	Designate the string from which a substring is to be extracted.
number-bytes	number	The number of characters to be extracted. *number- bytes* shall be at least 0.
start-pos	number	The starting position within *string*, where the first byte is position 1. If *start-pos* is greater than the length of *string*, or if *start-pos* and *number- bytes* combined exceeds the length of *string*, the whole of *string* shall be extracted.

Return Type and Value: text – A string containing *number-bytes*-worth of characters from *string*, starting at character position *start-pos*.

However, if

- *start-pos* < 0, #VALUE! is returned.
- *number-bytes* < 0, #VALUE! is returned.

[*Example*: Assuming 1-byte characters:

`MIDB("abcdef",3,2)` results in `cd`
`MIDB(A10,4,1)` results in `1`, when A10 contains `xyz123`
`MIDB("abcdef",4,5)` results in `def`

end example]

3.17.7.210 MIN

Syntax:

MIN (*argument-list*)

Description: Computes the smallest of a set of numbers.

Arguments:

Name	Type	Description
argument-list	logical, number,	The *argument*s in *argument-list* designate the values for

Name	Type	Description
	name, arrays, reference to number. Any *argument* can be an array or a reference to an array.	which the largest value is to be computed. Logical values and text representations of numbers occurring directly in the list of arguments are included. However, logical values and numbers in strings and are ignored inside references. [*Note*: To include these, use `MINA` (§3.17.7.211). *end note*] If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The smallest of a set of numbers; however, if the *argument*s contain no numbers, zero is returned.

[*Example*:

`MIN(10.4,-3.5,12.6)` results in `-3.5`
`MIN(10.4,{-3.5,12.6})` results in `-3.5`
`MIN({"ABC",TRUE})` results in `0`

Consider the case in which cell `B3` contains `0`:
`MIN(10,12,15,B3)` results in `10`
`MINA(10,12,15,B3)` results in `0`

end example]

3.17.7.211 MINA

Syntax:

`MINA` (*argument-list*)

Description: Computes the smallest of a set of numbers.

Arguments:

Name	Type	Description
argument-list	logical, number, name, arrays, reference to number. Any *argument* can be an array or a reference to an array.	The *argument*s in *argument-list* designate the values for which the largest value is to be computed. Logical values and text representations of numbers occurring directly in the list of arguments are included. Logical values and numbers in strings inside references are also included. [*Note*: To ignore these, use `MIN` (§3.17.7.210). *end note*] If an array or reference argument contains non-numeric text or empty cells, those values are ignored; however, cells with the value 0 are included.

Any *argument* in *argument-list* can be an array or a reference to an array.

Return Type and Value: number – The smallest of a set of numbers; however, if the *argument*s contain no numbers, zero is returned.

[*Example*:

`MINA(10.4,-3.5,12.6)` results in `-3.5`
`MINA(10.4,{-3.5,12.6})` results in `-3.5`
`MINA({"ABC",TRUE})` results in `0`

Consider the case in which cell `B3` contains `0`:
`MIN(10,12,15,B3)` results in `10`
`MINA(10,12,15,B3)` results in `0`

end example]

3.17.7.212 MINUTE

Syntax:

`MINUTE (` *time-value* `)`

Description: Computes the minute for the date and/or time having the given *time-value*.

Arguments:

Name	Type	Description
time-value	number	The date and/or time whose minute is to be computed. That date and/or time shall be expressed either as a serial value, in which case, its integer part is ignored, or as a *string-constant* having any valid date and/or time format, in which case, any date information shall be ignored.

Return Type and Value: number – The minute for the date and/or time having the given *time-value*. The returned value shall be in the range 0–59.

However, if *time-value* is out of range for the current date base value, `#NUM!` is returned.

[*Example*:

`MINUTE(DATE(2006,2,26)+TIME(2,10,20))` results in `10`
`MINUTE(TIME(22,56,34))` results in `56`

`MINUTE(0)` results in 0, since serial value 0 represents `00:00:00`
`MINUTE(10.5)` results in 0, since serial value .5 represents `12:00:00`
`MINUTE("22-Oct-2001 10:53:12")` results in `53`
`MINUTE("10:53:12 pm")` results in `53`
`MINUTE("22:53:12")` results in `53`

end example]

3.17.7.213 MINVERSE

Syntax:

`MINVERSE ( array )`

Description: Computes the inverse of the square matrix of numbers designated by *array*. The inverse matrix is calculated with an accuracy of at least 15 digits, which can lead to a small numeric error when the cancellation is not complete.

Arguments:

Name	Type	Description
array	array, reference	Designate a square matrix of numbers.

Return Type and Value: number – The inverse of the square matrix designated by *array*.

However, if

- Any cells in *array* are empty or contain text, the return value is unspecified.
- The matrix designated by *array* is not square, #VALUE! is returned.
- The matrix cannot be inverted, the return value is unspecified.

[*Example*:

`MINVERSE({3,6,1;1,1,0;3,10,2})` results in 2
`MINVERSE({3,6;1,1})` results in `-0.333333333`

end example]

3.17.7.214 MIRR

`MIRR ( values , finance-rate , reinvest-rate )`

Description: Computes the modified internal rate of return for a series of periodic cash flows. (Both the cost of the investment and the interest received on reinvestment of cash are considered.)

Mathematical Formula:

If n is the number of cash flows in *values*, frate is the *finance-rate*, and rrate is the *reinvest-rate*, then the formula for MIRR is:

$$\left(\frac{-\mathrm{NPV}(rrate, values[positive]) * (1+rrate)^{n}}{\mathrm{NPV}(frate, values[negative]) * (1+frate)}\right)^{\frac{1}{n-1}} - 1$$

Arguments:

Name	Type	Description
values	array, reference	Designates a set of numbers for which the rate of return is to be calculated. *values* shall contain at least one positive value and one negative value to calculate the internal rate of return. The order of numbers in *values* is significant, so be sure payment and income numbers are in the desired sequence. If *values* contains elements that are text, logical values, or empty cells, those elements are ignored.
finance-rate	number	The interest rate paid pay on the money used in the cash flows.
reinvest-rate	number	The interest rate received on the cash flows as they are reinvested.

Return Type and Value: number – The modified internal rate of return for a series of periodic cash flows.

However, if *values* does not contain at least one positive value and one negative value, `#DIV/0!` is returned.

[*Example*:

`MIRR({-120000,39000,30000,21000,37000,46000},0.1,0.12)` results in `12.6094%`
`MIRR({-120000,39000,30000,21000},0.1,0.12)` results in `-4.8045%`
`MIRR({-120000,39000,30000,21000,37000,46000},0.1,0.14)` results in `13.4759%`

end example]

3.17.7.215 MMULT

Syntax:

`MMULT ( array-1 , array-2 )`

Description: Computes the product of the matrices of numbers designated by *array-1* and *array-2*.

Mathematical Formula:

The matrix product array *a* of two arrays *b* and *c* is:

$$a_{ij} = \sum_{k=1}^{n} b_{ik} c_{kj}$$

where *i* is the row number, and *j* is the column number.

Arguments:

Name	Type	Description
array-1	array, reference, name	Designate the matrices of numbers to be multiplied.
array-2		

Return Type and Value: number – The product of the matrices of numbers designated by *array-1* and *array-2*.

However, if

- Any cells in *array-1* or *array-2* are empty or contain text, the return value is unspecified.
- The number of columns in *array-1* is different from the number of rows in *array-2*, #VALUE! is returned.

[*Example*:

`MMULT({3,6,1;1,1,0},{5,7;4,6;2,5})` results in `41`

end example]

3.17.7.216 MOD

Syntax:

`MOD ( x , y )`

Description: Computes the remainder when *x* is divided by *y*. The result has the same sign as *y*.

Arguments:

Name	Type	Description
x	number	The number for which the remainder is being sought.
y	number	The number by which *x* is to be divided.

Return Type and Value: number – The remainder when *x* is divided by *y*. The result has the same sign as *y*. If *y* is 0, the return value is unspecified.

[*Example*:

`MOD(3,2)` results in `1`

MOD(-3,2) results in 1
MOD(3,-2) results in -1
MOD(-3,-2) results in -1

end example]

3.17.7.217 MODE

Syntax:

MODE (*argument-list*)

Description: Computes the most frequently occurring of the numeric values of its arguments. If the set of values contains more than one most-frequent value, the first occurrence of any most-frequent value in the list is used as the result.

Arguments:

Name	Type	Description
argument-list	logical, number, name, arrays, reference to number. Any *argument* can be an array or a reference to an array.	The *argument*s in *argument-list* designate the values whose mode is to be computed. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The most frequently occurring of the values of its arguments.

However, if the data set contains no duplicate data points, #N/A is returned.

[*Example*:

MODE(9,1,5,1,9,5,6,6) results in 9
MODE(1,9,5,1,9,5,6,6) results in 1
MODE(5,1,9,5,1,9,6,6) results in 5

end example]

3.17.7.218 MONTH

Syntax:

MONTH (*date-value*)

Description: Computes the numeric Gregorian month for the date and/or time having the given *date-value*, taking into account the current date base value. That date and/or time shall be expressed either as a serial value, in which case, its fractional part is ignored, or as a *string-constant* having any valid date and/or time format, in which case, any time information shall be ignored.

Arguments:

Name	Type	Description
date-value	number, text	The date and/or time whose month is to be computed. That date and/or time shall be expressed either as a serial value, in which case, its fractional part is ignored, or as a *string-constant* having any valid date and/or time format, in which case, any time information shall be ignored.

Return Type and Value: number – The Gregorian month for the date and/or time having the given *date-value*, in the range 1900–9999.

However, if *date-value* is out of range for the current date base value, `#NUM!` is returned.

[*Example*:

`MONTH(DATE(2006,1,2))` results in `1`
`MONTH(DATE(2006,0,2))` results in `12`
`MONTH("2006/1/2 10:45 AM")` results in `1`
`MONTH(30000)` results in 2 for both the 1900 and 1904 date base systems

end example]

3.17.7.219 MROUND

Syntax:

`MROUND` (*x* , *multiple*)

Description: Computes *x* rounded to *multiple*, away from zero. It rounds up if the remainder of dividing *x* by *multiple* is greater than or equal to half the value of *multiple*; otherwise, it rounds down.

Arguments:

Name	Type	Description
x	number	The value to round.
multiple	number	The multiple to which *x* is to be rounded.

Return Type and Value: number – *x* rounded to *multiple*.

However, if *x* and *multiple* have different signs, `#NUM!` is returned.

[*Example*:

`MROUND(10,3)` rounds 10 to a nearest multiple of 3; that is, to 9
`MROUND(-10,-3)` rounds -10 to a nearest multiple of -3; that is, to `-9`
`MROUND(1.3,0.2)` rounds 1.3 to a nearest multiple of 0.2; that is, to `1.4`

end example]

3.17.7.220 MULTINOMIAL

Syntax:

`MULTINOMIAL` (*argument-list*)

Description: Computes the ratio of the factorial of the sum of the values in *argument-list* to the product of the factorials.

Mathematical Formula:

The multinomial is:

$$\mathrm{MULTINOMIAL}(a,b,c) = \frac{(a+b+c)!}{a!b!c!}$$

Arguments:

Name	Type	Description
argument-list	number	The *argument*s in *argument-list* designate the numerical values for which the multinomial is desired.

Return Type and Value: number – The ratio of the factorial of the sum of the values in *argument-list* to the product of the factorials.

However, if any *argument* is less than zero, `#NUM!` is returned.

[*Example*:

`MULTINOMIAL(2)` results in `1`
`MULTINOMIAL(2,3)` results in `10`
`MULTINOMIAL(2,3,4)` results in `1260`

end example]

3.17.7.221 N

Syntax:

N (*value*)

Description: Converts *value* to a number or, if *value* is a reference to a single cell, converts the value of that cell to a number.

Arguments:

Name	Type	Description
value	any	Value to be converted.

Return Type and Value: number or error – An integer that is the converted value of *value*, or, if *value* is a reference to a single cell, the converted value of that cell, as follows:

value	Value Returned
number	That number
TRUE	1
FALSE	0
error value	That error value
Anything else (including array and text)	0

[*Example*:

`N(10.5)` results in `10.5`
`N(A10)` results in `-1234`, when A10 contains the number `-1234`
`N("ABC")` results in `0`
`N(A10)` results in `0`, when A10 contains the string ABC
`N(TRUE)` results in `1`
`N(A10)` results in `0`, when A10 contains FALSE
`N(A10)` results in #N/A, when A10 contains #N/A
`N({12.5,13.6,56.9})` results in `12.50`
`N(A10:A11)` results in `0`, when A10 contains FALSE, and A11 contains 321

end example]

3.17.7.222 NA

Syntax:

NA ( )

Description: Gets the error value #N/A. (The error value #N/A can be used instead of a call to this function; the result is the same.)

Arguments: None.

Return Type and Value: error – The error value #N/A.

[*Example*:

NA() results in #N/A
IF(ISNA(NA()),"T","F") results in T

end example]

3.17.7.223 NEGBINOMDIST

Syntax:

NEGBINOMDIST (*number-failures* , *number-successes* , *success-probability*)

Description: Computes the negative binomial distribution. NEGBINOMDIST returns the probability that there will be *number-failures* failures before the *number-successes*th success, when the constant probability of a success is *success-probability*.

Mathematical Formula:

$$nb(x;r,p) = \binom{x+r-1}{r-1} p^r (1-p)^x$$

where:

x is *number-failures*, r is *number-successes*, and p is *success-probability*.

Arguments:

Name	Type	Description
number-failures	number	The number of failures, truncated to integer.
number-successes	number	The threshold number of successes, truncated to integer.
success-probability	number	The probability of a success.

Return Type and Value: number – The negative binomial distribution.

However, if

- *number-failures* < 0 or *number-successes* < 1, #NUM! is returned.
- *success-probability* < 0 or *success-probability* > 1, #NUM! is returned.

[*Example*:

NEGBINOMDIST(6,10,0.5) results in 0.076370239

end example]

3.17.7.224 NETWORKDAYS

Syntax:

NETWORKDAYS (*start-date* , *end-date* [, *holidays*])

Description: Computes the number of whole working days between *start-date* and *end-date*. Weekend days and any holidays specified by *holidays* are not considered as working days.

Arguments:

Name	Type	Description
start-date	number	The dates for which the difference is to be computed. *start-date* can be earlier than, the same as, or later than *end-date*.
end-date	number	
holidays	reference, array	An optional set of one or more dates that are to be excluded from the working day calendar. *holidays* shall be a range of cells that contain the dates, or an array constant of the serial values that represent those dates. The ordering of dates or serial values in *holidays* can be arbitrary.

Return Type and Value: number – The number of whole working days between *start-date* and *end-date*, excluding the specified holidays. If *start-date* is later than *end-date*, the return value shall be negative, and the magnitude shall be the number of whole working days.

However, if

- *start-date* is out of range for the current date base value, #NUM! is returned.
- *end-date* is out of range for the current date base value, #NUM! is returned.

[*Example*:

NETWORKDAYS(DATE(2006,1,1),DATE(2006,1,31)) results in 23
NETWORKDAYS(DATE(2006,1,31),DATE(2006,1,1)) results in -23

`NETWORKDAYS(DATE(2006,1,1),DATE(2006,2,1),{"2006/1/2","2006/1/16"})` results in 21

end example]

3.17.7.225 NOMINAL

`NOMINAL ( effect-rate , npery )`

Description: Computes the nominal annual interest rate, given the effective rate and the number of compounding periods per year.

Mathematical Formula:

NOMINAL is related to EFFECT:

$$EFFECT = \left(1 + \frac{Nominal\_rate}{Npery}\right)^{Npery} - 1$$

Arguments:

Name	Type	Description
effect-rate	number	The effective interest rate.
npery	number	The number of compounding periods per year, truncated to integer.

Return Type and Value: number – The nominal annual interest rate.

However, if

- *effect-rate* ≤ 0, `#NUM!` is returned.
- *npery* < 1, `#NUM!` is returned.

[*Example*:

`NOMINAL(0.053543,4)` results in `5.2500%`

end example]

3.17.7.226 NORMDIST

Syntax:

`NORMDIST ( x , mean , standard-deviation , cumulative-flag )`

Description: Computes the normal distribution for the specified mean and standard deviation.

Mathematical Formula:

The equation for the normal density function (*cumulative-flag* = FALSE) is:

$$f(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(\frac{(x-\mu)^2}{2\sigma^2}\right)}$$

When *cumulative-flag* = TRUE, the formula is the integral from negative infinity to x of the given formula.

Arguments:

Name	Type	Description
x	number	The value for which the distribution is to be computed.
mean	number	The arithmetic mean of the distribution.
standard-deviation	number	The standard deviation of the distribution.
cumulative-flag	logical	Determines the form of the function. If TRUE, then the cumulative distribution function is returned; if FALSE, the probability mass function is returned.

Return Type and Value: number – The normal distribution for the specified mean and standard deviation.

However, if *standard-deviation* ≤ 0, `#NUM!` is returned.

[*Example*:

`NORMDIST(42,40,1.5,TRUE)` results in `0.90878878`
`NORMDIST(42,40,1.5,FALSE)` results in `0.10934005`

end example]

3.17.7.227 NORMINV

Syntax:

`NORMINV` (*probability* , *mean* , *standard-deviation*)

Description: Computes the inverse of the normal distribution for the specified mean and standard deviation. `NORMINV` uses an iterative search technique.

Arguments:

Name	Type	Description
probability	number	The probability corresponding to the normal distribution.
mean	number	The arithmetic mean of the distribution.
standard-	number	The standard deviation of the distribution.

Name	Type	Description
deviation		

Return Type and Value: number – The inverse of the normal distribution for the specified mean and standard deviation.

However, if

- *probability* < 0 or if *probability* > 1, `#NUM!` is returned.
- *standard-deviation* ≤ 0, `#NUM!` is returned.
- The search has not converged after an implementation-defined number of iterations, `#N/A` is returned.

[*Example*:

`NORMINV(0.908789,40,1.5)` results in `42.00000201`

end example]

3.17.7.228 NORMSDIST

Syntax:

```
NORMSDIST ( z )
```

Description: Computes the standard normal distribution for the specified mean and standard deviation.

Mathematical Formula:

$$f(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}$$

Arguments:

Name	Type	Description
z	number	The value for which the distribution is to be computed.

Return Type and Value: number – The standard normal distribution for the specified mean and standard deviation.

[*Example*:

`NORMSDIST(1.333333)` results in `0.90878873`
`NORMSDIST(-1.5)` results in `0.06680720`

end example]

3.17.7.229 NORMSINV

Syntax:

NORMSINV (*probability*)

Description: Computes the inverse of the standard normal distribution. The distribution has a mean of zero and a standard deviation of 1. NORMSINV uses an iterative search technique.

Arguments:

Name	Type	Description
probability	number	The probability corresponding to the normal distribution.

Return Type and Value: number – The inverse of the standard normal distribution.

However, if

- *probability* < 0 or if *probability* > 1, #NUM! is returned.
- The search has not converged after an implementation-defined number of iterations, #N/A is returned.

[*Example*:

NORMSINV(0.945) results in 1.59819314
NORMSINV(0.13) results in -1.12639113

end example]

3.17.7.230 NOT

Syntax:

NOT (*logical-value*)

Description: Computes the logical negation of *logical-value*.

Arguments:

Name	Type	Description
logical-value	logical	The value to be negated.

Return Type and Value: logical – The logical negation of *logical-value*; that is, it returns TRUE if *logical-value* is FALSE, and FALSE if *logical-value* is TRUE.

[*Example*:

NOT(TRUE) results in FALSE
NOT(FALSE) results in TRUE
NOT(10>5) results in FALSE
NOT(16.567) results in FALSE

end example]

3.17.7.231 NOW

Syntax:

NOW ( )

Description: Computes the serial value of the current date and time, taking into account the current date base value.

Arguments: None.

Return Type and Value: number – The serial value of the current date and time.

[*Example*: On February 26, 2006, between 23:01 and 23:02, NOW() resulted in 38774.95958611110 for the 1900 date base system. On February 26, 2006, between 23:02 and 23:03, NOW() resulted in 37312.95982569440 for the 1904 date base system. *end example*]

3.17.7.232 NPER

Syntax:

NPER (*rate* , *pmt* , *pv* [, [*fv*] [, [*type*]]])

Description: Computes the number of periods for an investment based on periodic, constant payments and a constant interest rate.

Arguments:

Name	Type	Description
rate	number	The interest rate per period.
pmt	number	The payment made each period; it cannot change over the life of the annuity. Typically, *pmt* contains principal and interest but no other fees or taxes.
pv	number	The present value, or the lump-sum amount that a series of future payments is worth right now.
fv	number	The future value, or a cash balance to be attained after the last payment is made. If *fv* is omitted, it is assumed to be 0 (i.e., the future value of a loan, for example, is 0).

Name	Type	Description
type	number	The timing of the payment, truncated to integer, as follows: `0` – Payment at the end of the period `1` – Payment at the beginning of the period

Return Type and Value: number – The number of periods for an investment based on periodic, constant payments and a constant interest rate.

However, if *type* is any number other than 0 or 1, `#NUM!` is returned.

[*Example*:

`NPER(0.12/12,-100,-1000,10000,1)` results in `59.67`
`NPER(0.12/12,-100,-1000)` results in `-9.58`

end example]

3.17.7.233 NPV

Syntax:

NPV (*rate* , *argument-list*)

Description: Calculates the net present value of an investment by using a discount rate and a series of future payments and income.

The NPV investment begins one period before the date of the first *argument* cash flow and ends with the last cash flow in the list. The calculation is based on future cash flows. If the first cash flow occurs at the beginning of the first period, the first value shall be added to the NPV result, not included in *argument-list*.

Mathematical Formula:

If n is the number of cash flows in the list of values:

$$\mathrm{NPV} = \sum_{i=1}^{n} \frac{values_i}{(1+rate)^i}$$

Arguments:

Name	Type	Description

Name	Type	Description
rate	number	The rate of discount over the length of one period.
argument-list	number	The *argument*s in *argument-list* designate the series of future payments (negative values) and income (positive values). *argument*s shall be equally spaced in time and occur at the end of each period. The order of *argument*s is significant. *argument*s that are numbers, empty cells, logical values, or text representations of numbers are included; *argument*s that are error values or text that cannot be translated into numbers are ignored. If an *argument* is an array or reference, only numbers in that array or reference are included. Empty cells, logical values, text, or error values in the array or reference are ignored.

Return Type and Value: number – Net present value of an investment by using a discount rate and a series of future payments and income.

[*Example*:

`NPV(0.1,-10000,3000,4200,6800)` results in `1188.44`

end example]

3.17.7.234 OCT2BIN

Syntax:

`OCT2BIN ( number [ , num-bin-digits ] )`

Description: Makes the binary equivalent of *number*, with the result having *num-bin-digits* digits.

Arguments:

Name	Type	Description
number	text	A 10-digit octal number in a string that is to be converted to a binary string. If *number* has less than 10 digits, leading zero digits are implied until it has exactly 10 digits. The 10 digits use twos-complement representation with the left-most bit (30th bit from the right) representing the sign bit.
num-bin - digits	number	The number of digits in the result, with leading zeros added as necessary. However, if *number* is negative, *num-bin-digits* is ignored and the result has 10 digits. If *num-bin-digits* is omitted, the minimum number of digits is

Name	Type	Description
		used in the result. *num-bin-digits* is truncated to an integer.

Return Type and Value: text – The binary equivalent of *number*.

However, if

- *number* is outside the range "7777777000" (11111111111111111111000000000 binary, -512 decimal) to "777" (00000000000000000000111111111 binary, 511 decimal), inclusive, #NUM! is returned.
- *number* contains one or more non-octal digits, #NUM! is returned.
- *number* contains more than 10 octal digits, #NUM! is returned.
- *number* needs more digits that *num-bin-digits*, #NUM! is returned.
- *num-bin-digits* ≤ 0 or > 10, #NUM! is returned.

[*Example*:

OCT2BIN("67") results in 110111
OCT2BIN("7777777776") results in 1111111110
OCT2BIN("7",5) results in 00111

end example]

3.17.7.235 OCT2DEC

Syntax:

OCT2DEC (*number*)

Description: Makes the decimal equivalent of *number*.

Arguments:

Name	Type	Description
number	number	A 10-digit octal number in a string that is to be converted to a decimal number. If *number* has less than 10 digits, leading zero digits are implied until it has exactly 10 digits. The 10 digits use twos-complement representation with the left-most bit (30th bit from the right) representing the sign bit.

Return Type and Value: number – The decimal equivalent of *number*.

However, if

- *number* contains one or more non-octal digits, #NUM! is returned.
- *number* contains more than 10 octal digits; that is, *number* is outside the range "4000000000" (-536,870,912 decimal) to "3777777777" (536,870,911 decimal), inclusive, #NUM! is returned.

[*Example*:

OCT2DEC("67") results in 55
OCT2DEC("7777777776") results in -2
OCT2DEC("7000000000") results in -134217728

end example]

3.17.7.236 OCT2HEX

Syntax:

OCT2HEX (*number* [, *num-hex-digits*])

Description: Makes the hexadecimal equivalent of *number*, with the result having *num-hex-digits* digits.

Arguments:

Name	Type	Description
number	text	A 10-digit octal number in a string that is to be converted to a hexadecimal string. If *number* has less than 10 digits, leading zero digits are implied until it has exactly 10 digits. The 10 digits use twos-complement representation with the left-most bit (30th bit from the right) representing the sign bit.
num-hex-digits	number	*num-hex-digits* is the number of digits in the result, with leading zeros added as necessary. However, if *number* is negative, *num-hex-digits* is ignored and the result has 10 digits. If *num-hex-digits* is omitted, the minimum number of digits is used in the result. *num-hex-digits* is truncated to an integer.

Return Type and Value: text – The hexadecimal equivalent of *number*.

However, if

- *number* contains one or more non-octal digits, #NUM! is returned.
- *number* contains more than 10 octal digits; that is, *number* is outside the range "4000000000" (20000000 hex, -536,870,912 decimal) to "3777777777" (1FFFFFFF hex, 536,870,911 decimal), inclusive, #NUM! is returned.

- *number* needs more digits that *num-hex-digits*, `#NUM!` is returned.
- *num-hex-digits* ≤ 0or > 10, `#NUM!` is returned.

[*Example*:

`OCT2HEX("777")` results in `1FF`
`OCT2HEX("7777777776")` results in `FFFFFFFFFE`
`OCT2HEX("7",5)` results in `00007`

end example]

3.17.7.237 ODD

Syntax:

```
ODD ( x )
```

Description: Computes *x* rounded to the nearest odd integer, away from zero. Regardless of the sign of *x*, a value is rounded up when adjusted away from zero.

Arguments:

Name	Type	Description
x	number	The value to be rounded.

Return Type and Value: number – The rounded value of *x*.

[*Example*:

`ODD(1.5)` rounds 1.5 up to the nearest odd integer; that is, to 3
`ODD(3)` rounds 3 up to the nearest odd integer; that is, to 3
`ODD(2)` rounds 2 up to the nearest odd integer; that is, to 3
`ODD(-1)` rounds -1 up to the nearest odd integer; that is, to `-1`
`ODD(-2)` rounds -2 up to the nearest odd integer; that is, to `-3`

end example]

3.17.7.238 ODDFPRICE

Syntax:

```
ODDFPRICE ( settlement , maturity , issue , first-coupon , rate , yld , redemption ,
    frequency [ , [ basis ] ] )
```

Description: Computes the price per $100 face value of a security having an odd (short or long) first period.

Mathematical Formula:

Odd short first coupon:

$$ODDFPRICE = \left[\frac{redemption}{\left(1+\frac{yld}{frequency}\right)^{\left(N-1+\frac{DSC}{E}\right)}}\right] + \left[\frac{100 \times \frac{rate}{frequency} \times \frac{DFC}{E}}{\left(1+\frac{yld}{frequency}\right)^{\frac{DSC}{E}}}\right] + \left[\sum_{k=2}^{N} \frac{100 \times \frac{rate}{frequency}}{\left(1+\frac{yld}{frequency}\right)^{\left(k-1+\frac{DSC}{E}\right)}}\right] - \left[100 \times \frac{rate}{frequency} \times \frac{A}{E}\right]$$

where:

A = number of days from the beginning of the coupon period to the settlement date (accrued days).
DSC = number of days from the settlement to the next coupon date.
DFC = number of days from the beginning of the odd first coupon to the first coupon date.
E = number of days in the coupon period.
N = number of coupons payable between the settlement date and the redemption date. (If this number contains a fraction, it is raised to the next whole number.)

Odd long first coupon:

$$ODDFPRICE = \left[\frac{redemption}{\left(1+\frac{yld}{frequency}\right)^{\left(N+N_q+\frac{DSC}{E}\right)}}\right] + \left[\frac{100 \times \frac{rate}{frequency} \times \left[\sum_{i=1}^{NC}\frac{DC_i}{NL_i}\right]}{\left(1+\frac{yld}{frequency}\right)^{\left(N_q+\frac{DSC}{E}\right)}}\right] + \left[\sum_{k=1}^{N}\frac{100 \times \frac{rate}{frequency}}{\left(1+\frac{yld}{frequency}\right)^{\left(k-N_q+\frac{DSC}{E}\right)}}\right] - \left[100 \times \frac{rate}{frequency} \times \sum_{i=1}^{NC}\frac{A_i}{NL_i}\right]$$

where:

Ai = number of days from the beginning of the ith, or last, quasi-coupon period within odd period.
DCi = number of days from dated date (or issue date) to first quasi-coupon (i = 1) or number of days in quasi-coupon (i = 2,..., i = NC).
DSC = number of days from settlement to next coupon date.
E = number of days in coupon period.
N = number of coupons payable between the first real coupon date and redemption date. (If this number contains a fraction, it is raised to the next whole number.)
NC = number of quasi-coupon periods that fit in odd period. (If this number contains a fraction, it is raised to the next whole number.)
NLi = normal length in days of the full ith, or last, quasi-coupon period within odd period.
Nq = number of whole quasi-coupon periods between settlement date and first coupon.

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
issue	number	The security's issue date.
first-coupon	number	The security's first coupon date.
rate	number	The security's interest rate.

Name	Type	Description
yld	number	The security's annual yield.
redemption	number	The security's redemption value per $100 face value.
frequency	number	the number of coupon payments per year. (For annual payments, *frequency* is 1; for semiannual payments, *frequency* is 2; for quarterly payments, *frequency* is 4.)
basis	number	The truncated integer type of day count basis to use, as follows: **Value** / **Day Count Basis** 0 or omitted / US (NASD) 30/360 1 / Actual/actual 2 / Actual/360 3 / Actual/365 4 / European 30/360

Time information in the date arguments is ignored.

Return Type and Value: number – The price per $100 face value of a security having an odd (short or long) first period.

However, if

- *settlement*, *maturity*, *issue*, or *first-coupon* is out of range for the current date base value, `#NUM!` is returned.
- The following is not true: *maturity* is later than *first-coupon*, which is later than *settlement*, which is later than *issue*, so `#NUM!` is returned.
- *rate* or *yld* < 0, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

```
ODDFPRICE(DATE(2008,11,11),DATE(2021,3,1),DATE(2008,10,15),DATE(2009,3,1),
  0.0785,0.0625,100,2,1)
```
results in `113.5977`

end example]

3.17.7.239 ODDFYIELD

Syntax:

ODDFYIELD (*settlement* , *maturity* , *issue* , *first-coupon* , *rate* , *pr* , *redemption* , *frequency* [, [*basis*]])

Description: Computes the yield of a security that has an odd (short or long) first period.

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
issue	number	The security's issue date.
first-coupon	number	The security's first coupon date.
rate	number	The security's interest rate.
pr	number	The security's price.
redemption	number	The security's redemption value per $100 face value.
frequency	number	the number of coupon payments per year. (For annual payments, *frequency* is 1; for semiannual payments, *frequency* is 2; for quarterly payments, *frequency* is 4.)
basis	number	The truncated integer type of day count basis to use, as follows:

Value	Day Count Basis
0 or omitted	US (NASD) 30/360
1	Actual/actual
2	Actual/360
3	Actual/365
4	European 30/360

Time information in the date arguments is ignored.

Return Type and Value: number – The yield of a security that has an odd (short or long) first period.

However, if

- *settlement*, *maturity*, *issue*, or *first-coupon* is out of range for the current date base value, #NUM! is returned.
- The following is not true: *maturity* is later than *first-coupon*, which is later than *settlement*, which is later than *issue*, so #NUM! is returned.
- *rate* or *pr* < 0, #NUM! is returned.
- *frequency* is any number other than 1, 2, or 4, #NUM! is returned.
- *basis* < 0 or *basis* > 4, #NUM! is returned.

[*Example*:

```
ODDFYIELD(DATE(2008,11,11),DATE(2021,3,1),DATE(2008,10,15),DATE(2009,3,1),
  0.0575,84.5,100,2,0) results in 7.7246%
```

end example]

3.17.7.240 ODDLPRICE

Syntax:

```
ODDLPRICE ( settlement , maturity , last-interest , rate , yld , redemption ,
    frequency [ , [ basis ] ] )
```

Description: Computes the price per $100 face value of a security having an odd (short or long) last coupon period.

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
last-interest	number	The security's last coupon date.
rate	number	The security's interest rate.
yld	number	The security's annual yield.
redemption	number	The security's redemption value per $100 face value.
frequency	number	the number of coupon payments per year. (For annual payments, *frequency* is 1; for semiannual payments, *frequency* is 2; for quarterly payments, *frequency* is 4.)
basis	number	The truncated integer type of day count basis to use, as follows: **Value** / **Day Count Basis** 0 or omitted / US (NASD) 30/360 1 / Actual/actual 2 / Actual/360 3 / Actual/365 4 / European 30/360

Time information in the date arguments is ignored.

Return Type and Value: number – The price per $100 face value of a security having an odd (short or long) last coupon period.

However, if

- *settlement*, *maturity*, or *last-interest* is out of range for the current date base value, `#NUM!` is returned.
- The following is not true: *maturity* is later than *settlement*, which is later than *last-interest*, so `#NUM!` is returned.
- *rate* or *yld* < 0, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

```
ODDLPRICE(DATE(2008,11,11),DATE(2021,3,1),DATE(2008,10,15),
  0.0785,0.0625,100,2,1) results in 99.8783
```

end example]

3.17.7.241 ODDLYIELD

Syntax:

```
ODDLYIELD ( settlement , maturity , last-interest , rate , pr , redemption ,
    frequency [ , [ basis ] ] )
```

Description: Computes the yield of a security that has an odd (short or long) last period.

Mathematical Formula:

$$ODDLYIELD = \left[\frac{\left(redemption + \left(\left(\sum_{i=1}^{NC} DC_i / NL_i \right) \times \frac{100 \times rate}{frequency} \right) \right) - \left(par + \left(\left(\sum_{i=1}^{NC} A_i / NL_i \right) \times \frac{100 \times rate}{frequency} \right) \right)}{par + \left(\left(\sum_{i=1}^{NC} A_i / NL_i \right) \times \frac{100 \times rate}{frequency} \right)} \right] \times \left[\frac{frequency}{\left(\sum_{i=1}^{NC} DSC_i / NL_i \right)} \right]$$

where:

Ai = number of accrued days for the i^{th}, or last, quasi-coupon period within odd period counting forward from last interest date before redemption.
DCi = number of days counted in the i^{th}, or last, quasi-coupon period as delimited by the length of the actual

coupon period.
NC = number of quasi-coupon periods that fit in odd period; if this number contains a fraction it will be raised to the next whole number.
NLi = normal length in days of the i^{th}, or last, quasi-coupon period within odd coupon period.

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
last-interest	number	The security's last coupon date.
rate	number	The security's interest rate.
pr	number	The security's price.
redemption	number	The security's redemption value per $100 face value.
frequency	number	the number of coupon payments per year. (For annual payments, *frequency* is 1; for semiannual payments, *frequency* is 2; for quarterly payments, *frequency* is 4.)
basis	number	The truncated integer type of day count basis to use, as follows:

Value	Day Count Basis
0 or omitted	US (NASD) 30/360
1	Actual/actual
2	Actual/360
3	Actual/365
4	European 30/360

Time information in the date arguments is ignored.

Return Type and Value: number – The yield of a security that has an odd (short or long) last period.

However, if

- *settlement*, *maturity*, or *last-interest* is out of range for the current date base value, #NUM! is returned.
- The following is not true: *maturity* is later than *settlement*, which is later than *last-interest*, so #NUM! is returned.
- *rate* or *pr* < 0, #NUM! is returned.
- *frequency* is any number other than 1, 2, or 4, #NUM! is returned.
- *basis* < 0 or *basis* > 4, #NUM! is returned.

[*Example*:

```
ODDLYIELD(DATE(2008,11,11),DATE(2021,3,1),DATE(2008,10,15),
  0.0575,84.5,100,2,0) results in 4.5192%
```

end example]

3.17.7.242 OFFSET

Syntax:

OFFSET (*reference* , *rows* , *cols* [, [*height*] [, [*width*]]])

Description: Gets a reference to a range that is a specified number of rows and columns from a cell or range of cells. The reference that is returned can be a single cell or a range of cells. You can specify the number of rows and the number of columns to be returned.

Arguments:

Name	Type	Description
reference	reference	Designates the base. *reference* shall refer to a cell or range of adjacent cells.
rows	number	The number of rows, up or down, that indicates the upper-left cell of the result to refer to. A positive value means below the starting reference; a negative value means above the starting reference.
cols	number	The number of columns, to the left or right, that the upper-left cell of the result to refer to. A positive value means to the right of the starting reference; a negative value means to the left of the starting reference.
height	number	The height, in rows, of the set of cells referred to by the resulting reference. This height shall be positive. If omitted, it is the same as the height of *reference*.
width	number	The width, in columns, of the set of cells referred to by the resulting reference. The width shall be positive. If omitted, it is the same as the width of *reference*.

Return Type and Value: reference – A reference to a range that is a specified size and number of rows and columns from a cell or range of cells.

However, if

- *reference* does not refer to a cell or range of adjacent cells, #VALUE! is returned.
- The combination of *rows* and *cols* results outside the worksheet, #REF! is returned.

[*Example*:

`OFFSET(C3,2,3,1,1)` results in the value in cell `F5`
`SUM(OFFSET(C3:E5,-1,0,3,3))` results in the sum of the range `C2:E4`

end example]

3.17.7.243 OR

Syntax:

`OR` (*argument-list*)

Description: Tests if any one or more *argument*s in *argument-list* are TRUE.

Arguments:

Name	Type	Description
argument-list	logical, array, reference.	The *argument*s in *argument-list* designate the values to be tested. For an array or cell reference, a cell that contains text or is empty shall be ignored.

Return Type and Value: logical – TRUE if any one or more *argument*s in *argument-list* are TRUE; otherwise, `FALSE`.

However, if no logical values are found, the return value is unspecified.

[*Example*:

`OR(TRUE)` results in TRUE
`OR(FALSE,FALSE)` results in FALSE
`OR(10=5,3=1+2,0)` results in TRUE
`OR({10,5,6,7},TRUE,E6:F6)` results in TRUE, when E6 contains `FALSE` and F6 contains 0

end example]

3.17.7.244 PEARSON

Syntax:

`PEARSON` (*array-1* , *array-2*)

Description: Computes the Pearson product moment correlation coefficient, a dimensionless index that ranges from -1.0 to 1.0, inclusive, and reflects the extent of a linear relationship between two data sets.

Mathematical Formula:

The formula for the Pearson product moment correlation coefficient, r, is:

$$r = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sqrt{\sum (x - \bar{x})^2 \sum (y - \bar{y})^2}}$$

where x and y are the sample means AVERAGE(*array-1*) and AVERAGE(*array-2*).

Arguments:

Name	Type	Description
array-1	number, name, array, reference to number	The set of independent numerical values. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.
array-2	number, name, array, reference to number	The set of dependent numerical values. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The Pearson product moment correlation coefficient.

However, if

- *array-1* and *array-2* have a different number of data points, the return value is unspecified.
- *array-1* or *array-2* is empty, the return value is unspecified.

[*Example*:

`PEARSON({9,7,5,3,1},{10,6,1,5,3})` results in `0.699378606`

end example]

3.17.7.245 PERCENTILE

Syntax:

```
PERCENTILE ( array , k )
```

Description: Computes the k^{th} percentile of a set of values in a range.

Arguments:

Name	Type	Description
array	array, reference	The set of numerical data that defines relative standing.

Name	Type	Description
k	number	The percentile value in the range 0–1, inclusive. If *k* is not a multiple of 1/(n - 1), PERCENTILE interpolates to determine the value at the k^{th} percentile.

Return Type and Value: number – The k^{th} percentile of a set of values in a range.

However, if

- *array* is empty, the return value is unspecified.
- *k* is < 0 or *k* > 1, #NUM! is returned.

[*Example*:

PERCENTILE({1,3,2,4},0.3) results in 1.9
PERCENTILE({1,3,2,4},0.75) results in 3.25

end example]

3.17.7.246 PERCENTRANK

Syntax:

PERCENTRANK (*array* , *x* [, *significance*])

Description: Computes the rank of a value in a data set as a percentage of the data set.

Arguments:

Name	Type	Description
array	array, reference	*array* is the set of numerical data that defines relative standing.
x	number	The value for which the rank is to be computed. If *x* does not match one of the values in *array*, PERCENTRANK interpolates to return the correct percentage rank.
significance	number	The number of significant digits for the returned percentage value. If omitted, a value of 3 is used.

Return Type and Value: number – The rank of a value in a data set as a percentage of the data set.

However, if

- *array* is empty, the return value is unspecified.
- *significance* < 1, #NUM! is returned.

[*Example*:

`PERCENTRANK({12,6,7,9,3,8},4)` results in `0.066`
`PERCENTRANK({12,6,7,9,3,8},5)` results in `0.133`

end example]

3.17.7.247 PERMUT

Syntax:

`PERMUT ( number , number-chosen )`

Description: Computes the number of permutations for *number-chosen* objects that can be selected from *number* objects. [*Note*: A permutation is any set or subset of objects or events where internal order is significant. Permutations are different from combinations, for which the internal order is not significant. Use this function for lottery-style probability calculations. *end note*]

Mathematical Formula:

$$P_{k,n} = \frac{n!}{(n-k)!}$$

Arguments:

Name	Type	Description
number	number	The total number of items available, truncated to integer.
number-chosen	number	The number of items in each permutation, truncated to integer.

Return Type and Value: number – The number of different permutations of *number-chosen* in *number*.

However, if

- *number* < 0, `#NUM!` is returned.
- *number-chosen* < 0, `#NUM!` is returned.
- *number* < *number-chosen*, `#NUM!` is returned.

[*Example*:

`PERMUT(8,2)` results in `56`
`PERMUT(10,4)` results in `5040`
`PERMUT(6,5)` results in `720`

end example]

3.17.7.248 PHONETIC

Syntax:

```
PHONETIC ( string )
```

Description: Extracts the phonetic (furigana) characters from *string*. [*Note*: Furigana are aids used to indicate correct pronunciation of Japanese text. *end note*]

Arguments:

Name	Type	Description
string	text, reference	Designates a furigana string. If *string* is a cell range, the furigana string in the upper-left corner cell of that range is returned.

Return Type and Value: text – The phonetic (furigana) characters from *string*.

However, if *string* is a range of non-contiguous cells, `#N/A` is returned.

3.17.7.249 PI

Syntax:

```
PI ()
```

Description: Computes the value π to at least 15 significant figures.

Arguments: None.

Return Type and Value: number – The value π accurate to at least 15 significant digits.

[*Example*:

`PI()` results in `3.141592654`
`PI()/2` results in `1.570796327`
`PI()*(2.5^2)` results in `19.63495408`

end example]

3.17.7.250 PMT

Syntax:

```
PMT ( rate , nper , pv [ , [ fv ] [ , [ type ] ] ] )
```

Description: Computes the payment for a loan based on constant payments and a constant interest rate.

Arguments:

Name	Type	Description
rate	number	The interest rate for the loan.
nper	number	The total number of payment for the loan.
pv	number	The present value, or the total amount that a series of future payments is worth now; also known as the principal.
fv	number	The future value, or a cash balance to be attained after the last payment is made. If omitted, it is assumed to be 0 (i.e., the future value of a loan, for example, is 0).
type	number	The timing of the payment, truncated to integer, as follows: **Value** / **Timing** `0` – Payment at the end of the period `1` – Payment at the beginning of the period

Return Type and Value: number – The payment for a loan based on constant payments and a constant interest rate. (The payment returned by PMT includes principal and interest but no taxes, reserve payments, or fees sometimes associated with loans.)

Howver, if *type* is any number other than 0 or 1, `#NUM!` is returned.

[*Example*:

`PMT(0.08/12,10,10000)` results in `-1,037.03`
`PMT(0.08/12,10,10000,0,1)` results in `-1,030.16`

end example]

3.17.7.251 POISSON

Syntax:

`POISSON ( x , mean , cumulative-flag )`

Description: Computes the Poisson distribution.

Mathematical Formula:

For *cumulative-flag* = FALSE:

$$POISSON = \frac{e^{-\lambda}\ \lambda^{x}}{x!}$$

For *cumulative-flag* = TRUE:

$$CUMPOISSON = \sum_{k=0}^{x} \frac{e^{-\lambda}\ \lambda^{x}}{k!}$$

Arguments:

Name	Type	Description
x	number	The number of events, truncated to an integer.
mean	number	The expected numeric value.
cumulative-flag	logical	Determines the form of the function. If TRUE, POISSON returns the cumulative Poisson probability that the number of random events occurring will be between zero and *x*, inclusive; if FALSE, it returns the Poisson probability mass function that the number of events occurring will be exactly *x*.

Return Type and Value: number – The Poisson distribution.

However, if

- *x* < 0, #NUM! is returned.
- *mean* ≤ 0, #NUM! is returned.

[*Example*:

POISSON(2,5,TRUE) results in 0.124652019
POISSON(2,5,FALSE) results in 0.084224337

end example]

3.17.7.252 POWER

Syntax:

```
POWER ( x , y )
```

Description: Computes *x* raised to the power *y*.

Arguments:

Name	Type	Description

Name	Type	Description
x	number	The base and the number *y* is the exponent to which that base is raised.
y	number	The exponent to which the base is raised.

Return Type and Value: number – x^y.

However, if

- The value of *x* is negative and *y* is not a whole number, `#NUM!` is returned.
- *x* is zero and *y* is less than or equal to zero, `#DIV/0!` is returned.
- The result cannot be represented as a number, `#NUM!` is returned.

[*Example*:

`POWER(2,3)` results in `8`
`POWER(2,0.5)` results in `1.414213562`
`POWER(-1.234,5.0)` results in `-2.861381721`
`POWER(1.234,5.1)` results in `2.922182358`

end example]

3.17.7.253 PPMT

Syntax:

`PPMT ( rate , per , nper , pv [ , [ fv ][ , [ type ]]] )`

Description: Computes the payment on the principal for a given period for an investment based on periodic, constant payments and a constant interest rate.

Arguments:

Name	Type	Description
rate	number	The interest rate per period.
per	number	The the period and shall be in the range 1–*nper*.
nper	number	The total number of payment in an annuity.
pv	number	The present value, or the total amount that a series of future payments is worth now.
fv	number	The future value, or a cash balance to be attained after the last payment is made. If omitted, it is assumed to be 0 (i.e., the future value of a loan, for example, is 0).
type	number	The timing of the payment, truncated to integer, as

Name	Type	Description
		follows: **Value** / **Timing** `0` — Payment at the end of the period `1` — Payment at the beginning of the period

Return Type and Value: number – The payment on the principal for a given period for an investment based on periodic, constant payments and a constant interest rate.

However, if *type* is any number other than 0 or 1, `#NUM!` is returned.

[*Example*:

`PPMT(0.1/12,1,2*12,2000)` results in `-75.62`
`PPMT(0.08,10,10,200000)` results in `-27,598.05`

end example]

3.17.7.254 PRICE

Syntax:

`PRICE ( `*settlement*` , `*maturity*` , `*rate*` , `*yld*` , `*redemption*` , `*frequency*` [ , [ `*basis*` ] ] )`

Description: Computes the price per $100 face value of a security that pays periodic interest.

Mathematical Formula:

$$PRICE = \left[\frac{redemption}{\left(1+\frac{yld}{frequency}\right)^{\left(N-1+\frac{DSC}{E}\right)}}\right] + \left[\sum_{k=1}^{N}\frac{100\times\frac{rate}{frequency}}{\left(1+\frac{yld}{frequency}\right)^{\left(k-1+\frac{DSC}{E}\right)}}\right] - \left(100\times\frac{rate}{frequency}\times\frac{A}{E}\right)$$

where:

DSC = number of days from settlement to next coupon date.
E = number of days in coupon period in which the settlement date falls.

N = number of coupons payable between settlement date and redemption date.
A = number of days from beginning of coupon period to settlement date.

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
rate	number	The security's interest rate.
yield	number	The security's annual yield.
redemption	number	The security's redemption value per $100 face value.
frequency	number	the number of coupon payments per year. (For annual payments, *frequency* is 1; for semiannual payments, *frequency* is 2; for quarterly payments, *frequency* is 4.) *frequency* is truncated to an integer.
basis	number	The truncated integer type of day count basis to use, as follows:

Value	Day Count Basis
0 or omitted	US (NASD) 30/360
1	Actual/actual
2	Actual/360
3	Actual/365
4	European 30/360

Time information in the date arguments is ignored.

Return Type and Value: number – The price per $100 face value of a security that pays periodic interest.

However, if

- *settlement* or *maturity* is out of range for the current date base value, #NUM! is returned.
- *settlement* ≥ *maturity*, #NUM! is returned.
- *rate* or *yld* < 0, #NUM! is returned.
- *redemption* ≤ 0, #NUM! is returned.
- *frequency* is any number other than 1, 2, or 4, #NUM! is returned.
- *basis* < 0 or *basis* > 4, #NUM! is returned.

[*Example*:

`PRICE(DATE(2008,2,15),DATE(2017,11,15),0.0575,0.065,100,2,0)` results in `94.6344`

end example]

3.17.7.255 PRICEDISC

Syntax:

`PRICEDISC ( `*settlement*` , `*maturity*` , `*discount*` , `*redemption*` [ , [ `*basis*` ] ] )`

Description: Computes the price per $100 face value of a discounted security.

Mathematical Formula:

$$PRICEDISC = redemption - discount \times redemption \times \frac{DSM}{B}$$

where:

B = number of days in year, depending on year basis.
DSM = number of days from settlement to maturity.

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
discount	number	The security's discount rate.
redemption	number	The security's redemption value per $100 face value.
basis	number	The truncated integer type of day count basis to use, as follows: 0 or omitted – US (NASD) 30/360 1 – Actual/actual 2 – Actual/360 3 – Actual/365 4 – European 30/360

Time information in the date arguments is ignored.

Return Type and Value: number – The price per $100 face value of a discounted security.

However, if

- *settlement* or *maturity* is out of range for the current date base value, `#NUM!` is returned.

- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *discount* or *redemption* ≤ 0, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`PRICEDISC(DATE(2008,2,16),DATE(2008,3,1),0.0525,100,2)` results in `99.7958`

end example]

3.17.7.256 PRICEMAT

Syntax:

`PRICEMAT ( settlement , maturity , issue , rate , yld [ , [ basis ] ] )`

Description: Computes the price per $100 face value of a security that pays interest at maturity.

Mathematical Formula:

$$PRICEMAT = \frac{100 + (\frac{DIM}{B} \times rate \times 100)}{1 + (\frac{DSM}{B} \times yld)} - \left(\frac{A}{B} \times rate \times 100\right)$$

where:

B = number of days in year, depending on year basis.
DSM = number of days from settlement to maturity.
DIM = number of days from issue to maturity.
A = number of days from issue to settlement.

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
issue	number	The security's issue date.
rate	number	The security's interest rate.
yld	number	The security's annual yield.
basis	number	The truncated integer type of day count basis to use, as follows: **Value** / **Day Count Basis** `0` or omitted / US (NASD) 30/360

Name	Type	Description
		1 Actual/actual 2 Actual/360 3 Actual/365 4 European 30/360

Time information in the date arguments is ignored.

Return Type and Value: number – The price per $100 face value of a security that pays interest at maturity.

However, if

- *settlement*, *maturity*, or *issue* is out of range for the current date base value, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *rate* or *yld* < 0, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

```
PRICEMAT(DATE(2008,2,15),DATE(2008,4,13),DATE(2007,11,11),0.061,0.061,0)
```

results in `99.9845`

end example]

3.17.7.257 PROB

Syntax:

PROB (*x-range* , *probability-range* , *lower-limit* [, *upper-limit*])

Description: Computes the probability that values in a range are between two limits.

Arguments:

Name	Type	Description
x-range	array, reference	The set of numeric values of *x* with which there are associated probabilities.
probability-range	array, reference	A set of numeric probabilities associated with the values in *x-range*.
lower-limit	number	The lower bound on the value for which the probability is to be computed.
upper-limit	number	The upper bound on the value for which the probability is to be computed. If omitted, the probability that values in

Name	Type	Description
		x-range are equal to *lower-limit* is returned.

Return Type and Value: number – The probability that values in a range are between two limits.

However, if

- Any value in *probability-range* ≤ 0 or any value in *probability-range* > 1, `#NUM!` is returned.
- The sum of the values in *probability-range* < 1, `#NUM!` is returned.
- *x-range* and *probability-range* contain a different number of data points, the return value is unspecified.

[*Example*:

`PROB({0,1,2,3},{0.2,0.3,0.1,0.4},2)` results in `0.1`
`PROB({0,1,2,3},{0.2,0.3,0.1,0.4},1,4)` results in `0.8`

end example]

3.17.7.258 PRODUCT

Syntax:

PRODUCT (*argument-list*)

Description: Multiplies the numeric values of *argument*s in *argument-list*.

Arguments:

Name	Type	Description
argument-list	logical, number, text, array, reference	The *argument*s in *argument-list* designate the numbers to be multiplied. Arguments that are numbers, logical values, or text representations of numbers shall be counted. If an argument is an array or reference, only numbers in that array or reference shall be counted. Empty cells, logical values, and text in the array or reference shall be ignored.

Return Type and Value: number – The product of the values of its arguments.

[*Example*:

`PRODUCT(1)` results in `1`
`PRODUCT(1,2,3,4,5)` results in `120`
`PRODUCT({1,2;3,4})` results in `24`

PRODUCT({2,3},4,"5") results in 120

end example]

3.17.7.259 PROPER

Syntax:

PROPER (*string*)

Description: Makes a lowercase version of *string* except that the first letter in *string* and any other letters in *string* that immediately follow a character that is not a letter, are converted to uppercase.

Arguments:

Name	Type	Description
string	text, reference	Designates the string to be converted.

Return Type and Value: text – A version of *string* such that the first letter in *string* and any other letters in *string* that immediately follow a character that is not a letter, are converted to uppercase. All other letters are converted to lowercase, and all other non-letters are unchanged.

[*Example*:

PROPER("12aBC d123aD#$%sd^") results in 12Abc D123Ad#$%Sd^
PROPER(A10) results in 12Abc D123Ad#$%Sd^, when A10 contains 12aBC d123aD#$%sd^

end example]

3.17.7.260 PV

Syntax:

PV (*rate* , *nper* , *pmt* [, [*fv*][, [*type*]]])

Description: Computes the present value of an investment. (The present value is the total amount that a series of future payments is worth now.)

Mathematical Formula:

If *rate* is not 0, then:

$$pv * (1 + rate)^{nper} + pmt(1 + rate * type) * \left(\frac{(1 + rate)^{nper} - 1}{rate}\right) + fv = 0$$

If *rate* is 0, then:

(pmt * nper) + pv + fv = 0

Arguments:

Name	Type	Description
rate	number	The interest rate per period.
nper	number	The total number of payment in an annuity.
pmt	number	The payment made each period and cannot change over the life of the annuity. If is omitted, *fv* shall be provided. [*Note*: Typically, *pmt* includes principal and interest but no other fees or taxes. *end note*]
fv	number	The future value, or a cash balance to be attained after the last payment is made. If omitted, *pmt* shall be provided.
type	number	The timing of the payment, truncated to integer, as follows:

Value	Timing
`0`	Payment at the end of the period
`1`	Payment at the beginning of the period

Return Type and Value: number – The present value of an investment.

However, if *type* is any number other than 0 or 1, `#NUM!` is returned.

[*Example*:

`PV(0.08/12,12*20,500,,0)` results in `-59,777.15`

end example]

3.17.7.261 QUARTILE

Syntax:

`QUARTILE` (*array* , *result-category*)

Description: Computes the quartile of a data set.

Arguments:

Name	Type	Description

Name	Type	Description
array	array, reference	The set of numeric values for which the quartile value is to be computed.
result-category	number	When truncated to an integer, specifies which value is to be returned, as follows: **Value** / **Value Returned** 0 – Minimum value 1 – First quartile (25th percentile) 2 – Median value (50th percentile) 3 – Third quartile (75th percentile) 4 – Maximum value

Return Type and Value: number – The quartile of a data set.

However, if

- *array* is empty, the return value is unspecified.
- *result-category* < 0 or *result-category* > 4, `#NUM!` is returned.

[*Example*:

`QUARTILE({1,2,4,7,8,9,10,12},1)` results in `3.5`

end example]

3.17.7.262 QUOTIENT

Syntax:

`QUOTIENT ( dividend , divisor )`

Description: Computes the integer portion of the division of *dividend* by *divisor*.

Arguments:

Name	Type	Description
dividend	number	The dividend
divisor	number	The divisor.

Return Type and Value: number – The integer portion of the division of *dividend* by *divisor*.

[*Example*:

`QUOTIENT(5,2)` results in 2
`QUOTIENT(4.5,3.1)` results in 1
`QUOTIENT(-10,3)` results in -3

end example]

3.17.7.263 RADIANS

Syntax:

```
RADIANS ( angle )
```

Description: Converts *angle* in degrees into radians.

Arguments:

Name	Type	Description
angle	number	The angle expressed in degrees that is to be converted into radians.

Return Type and Value: number – *angle* in radians.

[*Example*:

`RADIANS(360)` results in `6.283185307`
`RADIANS(270)` results in `4.71238898`
`RADIANS(45)` results in `0.785398163`
`RADIANS(8.5)` results in `0.148352986`

end example]

3.17.7.264 RAND

Syntax:

```
RAND ( )
```

Description: Computes an evenly distributed random real number greater than or equal to 0 and less than 1. A new random real number is returned every time the cell's value is calculated.

Arguments: None.

Return Type and Value: number – An evenly distributed random real number greater than or equal to 0 and less than 1.

[*Example*:

`RAND()` results in `0.437337454`
`INT(RAND()*(6-1)+1)` might result in `3`

end example]

3.17.7.265 RANDBETWEEN

Syntax:

`RANDBETWEEN ( lower-bound , upper-bound )`

Description: Computes a random integer number in the range *lower-bound*–*upper-bound*. A new random integer number is returned every time the cell's value is calculated.

Arguments:

Name	Type	Description
lower-bound	number	The smallest integer that will be returned.
upper-bound	number	The largest integer that will be returned.

Return Type and Value: number – A random integer number in the range specified.

However, if *lower-bound* is greater than *upper-bound*, `#NUM!` is returned.

[*Example*:

`RANDBETWEEN(1,6)` results in an integer between `1` and `6`, inclusive
`RANDBETWEEN(-10,10)` results in an integer between `-10` and `10`, inclusive

end example]

3.17.7.266 RANK

Syntax:

`RANK ( number , number-list [ , order ] )`

Description: Computes the rank of a number in a list of numbers. RANK gives duplicate numbers the same rank. However, the presence of duplicate numbers affects the ranks of subsequent numbers.

Arguments:

Name	Type	Description
number	number	The number whose rank is to be found.

Name	Type	Description
number-list	reference	Designates the list of numbers. Non-numeric values in this list are ignored.
order	number	Specifies how *number* is to be ranked. If zero or omitted, *number* is ranked as if the list were sorted in descending order. If *order* is any non-zero value, *number* is ranked as if the list were sorted in ascending order.

Return Type and Value: number – The rank of a number in a list of numbers.

[*Example*:

When the cells `E1:I1` contain 7, 3.5, 3.5, 1, and 2

`RANK(E2,E1:I1,1)` results in 3
`RANK(E2,E1:I1,1,0)` results in 2
`RANK(E2,E1:I1,1)` results in 2
`RANK(E1,E1:I1,1)` results in 5, as the two 3.5 values both have a rank of 3; no value has rank 4.

end example]

3.17.7.267 RATE

Syntax:

RATE (*nper* , *pmt* , *pv* [, [[*fv*] [, [[*type*][, [*guess*]]]]]])

Description: Computes the interest rate per period of an annuity, using iteration, which can result in zero or more solutions.

Arguments:

Name	Type	Description
nper	number	The total number of payment periods.
pmt	number	The payment made each period and cannot change over the life of the annuity. (Typically, *pmt* includes principal and interest but no other fees or taxes.) If omitted, *fv* shall be present.
pv	number	The present value.
fv	number	The future value, or a cash balance to be attained after the last payment is made. If omitted, it is assumed to be 0 (i.e., the future value of a loan, for example, is 0).
type	number	The timing of the payment, truncated to integer, as follows:

Name	Type	Description
		<table><tr><th>Value</th><th>Timing</th></tr><tr><td>0</td><td>Payment at the end of the period</td></tr><tr><td>1</td><td>Payment at the beginning of the period</td></tr></table>
guess	number	A guess for what the rate will be. If omitted, it is assumed to be 10 percent.

Return Type and Value: number – The interest rate per period of an annuity.

However, if

- *type* is any number other than 0 or 1, `#NUM!` is returned.
- The result has not converged after an implementation-defined number of iterations, `#NUM!` is returned.

[*Example*:

`RATE(4*12,-200,8000)` results in `0.7701%`
`RATE(4*12,-200,8000)*12` results in `9.2418%`

end example]

3.17.7.268 RECEIVED

Syntax:

`RECEIVED ( `*settlement*` , `*maturity*` , `*investment*` , `*discount*` [ , [ `*basis*` ] ] )`

Description: Computes the amount received at maturity for a fully invested security.

Mathematical Formula:

$$RECEIVED = \frac{investment}{1 - (discount \times \frac{DIM}{B})}$$

where:

B = number of days in a year, depending on the year basis.
DIM = number of days from issue to maturity.

Arguments:

Name	Type	Description

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
investment	number	The amount invested in the security.
discount	number	The security's discount rate.
basis	number	The truncated integer type of day count basis to use, as follows: **Value** / **Day Count Basis** 0 or omitted / US (NASD) 30/360 1 / Actual/actual 2 / Actual/360 3 / Actual/365 4 / European 30/360

Time information in the date arguments is ignored.

Return Type and Value: number – The amount received at maturity for a fully invested security.

However, if

- *settlement* or *maturity* is out of range for the current date base value, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *investment* or *discount* ≤ 0, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`RECEIVED(DATE(2008,2,15),DATE(2008,5,15),1000000,0.0575,2)` results in `1014584.65`

end example]

3.17.7.269 REPLACE

Syntax:

REPLACE (*string-1* , *start-pos* , *number-chars* , *string-2*)

Description: Produces a new string that is *string-1* with *number-chars* characters starting at position *start-pos*, replaced by *string-2*. (REPLACE is intended for use with languages that use the single-byte character set (SBCS), whereas REPLACEB (§3.17.7.270) is intended for use with languages that use the double-byte character set (DBCS).)

Arguments:

Name	Type	Description
string-1	text	Designates a string.
start-pos	number	The number of the start position within *string-1* from which characters in *string-1* are to be replaced. The start position of the first character is 1. *start-pos* shall be at least 0. If *start-pos* is beyond the end of *string-1*, the result is a new string that is *string-2* appended to *string-1*. If *start-pos* is within the bounds of *string-1*, but *number-chars* goes beyond the end of *string-1*, the characters starting at position *start-pos* through to the end of *string-1* shall be replaced by *string-2*.
number-chars	number	The number of characters within *string-1* that are to be replaced by the string designated by *string-2*.
string-2	text	Designates a string.

Return Type and Value: text – A copy of *string-1* with replacement characters from *string-2*.

However, if

- *start-pos* < 0, #VALUE! is returned.
- *number-chars* < 0, #VALUE! is returned.

[*Example*:

`REPLACE("abcdefghijk",3,4,"XY")` results in `abXYghijk`
`REPLACE("abcdefghijk",3,1,"12345")` results in `ab12345defghijk`
`REPLACE("abcdefghijk",15,4,"XY")` results in `abcdefghijkXY`

end example]

3.17.7.270 REPLACEB

Syntax:

`REPLACEB ( string-1 , start-pos , number-bytes , string-2 )`

Description: Produces a new string that is *string-1* with *number-bytes* bytes starting at position *start-pos*, replaced by *string-2*. (REPLACEB is intended for use with languages that use the double-byte character set (DBCS), whereas REPLACE (§3.17.7.269) is intended for use with languages that use the single-byte character set (SBCS).)

Arguments:

Name	Type	Description
string-1	text	Designates a string.
start-pos	number	The number of the start position within *string-1* from which characters in *string-1* are to be replaced. The start position of the first character is 1. *start-pos* shall be at least 0. If *start-pos* is beyond the end of *string-1*, the result is a new string that is *string-2* appended to *string-1*. If *start-pos* is within the bounds of *string-1*, but *number- bytes* goes beyond the end of *string-1*, the characters starting at position *start-pos* through to the end of *string-1* shall be replaced by *string-2*.
number-bytes	number	The number of characters within *string-1* that are to be replaced by the string designated by *string-2*.
string-2	text	Designates a string.

Return Type and Value: text – A copy of *string-1* with replacement characters from *string-2*.

However, if

- *start-pos* < 0, #VALUE! is returned.
- *number-bytes* < 0, #VALUE! is returned.

[*Example*: Assuming 1-byte characters:

`REPLACEB("abcdefghijk",3,4,"XY")` results in `abXYghijk`
`REPLACEB("abcdefghijk",3,1,"12345")` results in `ab12345defghijk`
`REPLACEB("abcdefghijk",15,4,"XY")` results in `abcdefghijkXY`

end example]

3.17.7.271 REPT

Syntax:

`REPT` (*string* , *replication-count*)

Description: Creates a string that is *replication-count* number of occurrences of *string* concatenated together.

Arguments:

Name	Type	Description
string	text	Designate the string to be replicated.
replication-count	number	The number of times *string* is to be replicated, truncated to integer. If *replication-count* is 0, the resulting string is empty.

Return Type and Value: text – The final replicated string.

However, if *replication-count* < 0, #VALUE! is returned.

[*Example*:

REPT("ABC",3) results in ABCABCABC
LEN(REPT("ABC",0)) results in 0

end example]

3.17.7.272 RIGHT

Syntax:

RIGHT (*string* [, *number-chars*])

Description: Extracts the right-most *number-chars* characters from *string*. (RIGHT is intended for use with languages that use the single-byte character set (SBCS), whereas RIGHTB (§3.17.7.273) is intended for use with languages that use the double-byte character set (DBCS).)

Arguments:

Name	Type	Description
string	text	Designate the string from which a substring is to be extracted.
number-chars	number	The number of characters to be extracted. If omitted, a count of 1 shall be assumed. *number-chars* shall be at least 0. If *number-chars* exceeds the length of *string*, the whole of *string* shall be extracted.

Return Type and Value: text – A string containing the right-most *number-chars* characters from *string*.

However, if *number-chars* < 0, #VALUE! is returned.

[*Example*:

RIGHT("abcdef",2) results in ef
RIGHT(A10,4) results in z123, when A10 contains xyz123

end example]

3.17.7.273 RIGHTB

Syntax:

RIGHTB (*string* , [*number-bytes*])

Description: Extracts the right-most *number-bytes*-worth of characters from *string*. (RIGHTB is intended for use with languages that use the double-byte character set (DBCS), whereas RIGHT (§3.17.7.272) is intended for use with languages that use the single-byte character set (SBCS).)

Arguments:

Name	Type	Description
string	text	Designate the string from which a substring is to be extracted.
number-bytes	number	The number of bytes to be extracted. If omitted, a count of 1 shall be assumed. *number-bytes* shall be at least 0. If *number-bytes* exceeds the length of *string*, the whole of *string* shall be extracted.

Return Type and Value: text – A string containing the right-most *number-bytes*-worth of characters from *string*.

However, if *number-bytes* < 0, #VALUE! is returned.

[*Example*: Assuming 1-byte characters:

RIGHTB("abcdef",2) results in ef
RIGHTB(A10,4) results in z123, when A10 contains xyz123

end example]

3.17.7.274 ROMAN

Syntax:

ROMAN (*number* , *form*)

Description: Converts the Arabic number, *number*, to a Roman number according to *form*.

Arguments:

Name	Type	Description
number	number	The Arabic number to be converted.
form	number	Specifies the type of Roman numeral to be produced. The Roman numeral style ranges from Classic to Simplified, becoming more concise as the value of *form* increases, as follows: **Value** | **Type**

Name	Type	Description
		0, omitted, or TRUE: Classic. Only subtract powers of ten (but not L or V). Do not subtract a number from one that is more than 10 times greater. If another letter follows the larger one, it must be smaller than the number preceding the larger one. 1: Concise. Allow subtractraction of L and V as well as powers of ten. Do not subtract a number from one that is more than 10 times greater. If another letter follows the larger one, it must be smaller than the number preceding the larger one. 2: More concise. Allow subtractraction of L (but not V) as well as powers of ten. Allow subtraction of a number from one that is more than 10 times greater. If another letter follows the larger one, it must be smaller than the number preceding the larger one. 3: Most concise. Allow subtractraction of L and V as well as powers of ten. Allow subtraction of a number from one that is more than 10 times greater. If another letter follows the larger one, it must be smaller than the number preceding the larger one. 4 or FALSE: Simplified. Produce the fewest Roman digits.

Return Type and Value: text – The corresponding Roman number.

However, if

- *number* < 0 or > 3999, #VALUE! is returned.
- *form* is not one of the values listed above, #VALUE! is returned.

[*Example*:

ROMAN(499,0) results in CDXCIX, which is 100 less than 500, plus 10 less than 100, plus one less than 10.
ROMAN(499,1) results in LDVLIV, which is 50 less than 500, plus 5 less than 50, plus one less than 5.
ROMAN(499,2) results in XDIX, which is 10 less than 500, plus one less than 10.
ROMAN(499,3) results in VDIV, which is 5 less than 500, plus one less than 5.

ROMAN(499,4) results in ID, which is 1 less than 500.
ROMAN(2013,0) results in MMXIII, which is 2,000, plus 10, plus 3.

end example]

3.17.7.275 ROUND

Syntax:

ROUND (*x* , *number-digits*)

Description: Rounds *x* to the number of digits specified by *number-digits*.

Arguments:

Name	Type	Description
x	number	The value to be rounded.
number-digits	number	The number of digits to which *x* is to be rounded. If *number-digits* is greater than 0, *x* is rounded to the specified number of decimal places. If *number-digits* is 0, *x* is rounded to the nearest integer. If *number-digits* is less than 0, *x* is rounded to the left of the decimal point.

Return Type and Value: number – The rounded-down value of *x*.

[*Example*:

ROUND(2.15,1) results in 2.2
ROUND(2.149,1) results in 2.1
ROUND(-1.475,2) results in -1.48
ROUND(21.5,-1) results in 20

end example]

3.17.7.276 ROUNDDOWN

Syntax:

ROUNDDOWN (*x* , *number-digits*)

Description: Computes *x* rounded down, toward zero, to the number of digits specified by *number-digits*. [*Note*: ROUNDDOWN behaves like ROUND (§3.17.7.275), except that ROUNDDOWN always rounds a number down. *end note*]

Arguments:

Name	Type	Description

Name	Type	Description
x	number	The value to be rounded down.
number-digits	number	The number of digits to which *x* is to be rounded. If *number-digits* is greater than 0, *x* is rounded to the specified number of decimal places. If *number-digits* is 0, *x* is rounded to the nearest integer. If *number-digits* is less than 0, *x* is rounded to the left of the decimal point.

Return Type and Value: number – The rounded-down value of *x*.

[*Example*:

`ROUNDDOWN(3.2,0)` rounds 3.2 down to zero decimal places; that is, to 3
`ROUNDDOWN(76.9,0)` rounds 76.9 down to zero decimal places; that is, to 76
`ROUNDDOWN(3.14159,3)` rounds 3.14159 down to three decimal places; that is, to `3.141`
`ROUNDDOWN(-3.14159,1)` rounds -3.14159 down to one decimal place; that is, to `-3.1`
`ROUNDDOWN(31415.92654,-2)` rounds 31415.92654 down to two decimal places to the left of the decimal; that is, to `31400`

end example]

3.17.7.277 ROUNDUP

Syntax:

ROUNDUP (*x* , *number-digits*)

Description: Computes *x* rounded up, away from zero, to the number of digits specified by *number-digits*. [*Note*: ROUNDUP behaves like ROUND (§3.17.7.275), except that ROUNDUP always rounds a number up. *end note*]

Arguments:

Name	Type	Description
x	number	The value to be rounded up.
number-digits	number	The number of digits to which *x* is to be rounded. If *number-digits* is greater than 0, *x* is rounded up to the specified number of decimal places. If *number-digits* is 0, *x* is rounded up to the nearest integer. If *number-digits* is less than 0, *x* is rounded up to the left of the decimal point.

Return Type and Value: number – The rounded-up value of *x*.

[*Example*:

`ROUNDDOWN(3.2,0)` rounds 3.2 down to zero decimal places; that is, to `4`
`ROUNDDOWN(76.9,0)` rounds 76.9 down to zero decimal places; that is, to `77`
`ROUNDDOWN(3.14159,3)` rounds 3.14159 down to three decimal places; that is, to `3.142`
`ROUNDDOWN(-3.14159,1)` rounds -3.14159 down to one decimal place; that is, to `-3.2`
`ROUNDDOWN(31415.92654,-2)` rounds 31415.92654 down to two decimal places to the left of the decimal; that is, to `31500`

end example]

3.17.7.278 ROW

Syntax:

`ROW ( [ reference ] )`

Description: Finds the number of the row(s) corresponding to *reference*.

Arguments:

Name	Type	Description
reference	reference to a single cell or to a range of contiguous cells	If omitted, the behavior is as if *reference* referred to the cell containing the formula.

Return Type and Value: number – If *reference* refers to a single cell or to a single row of cells, the corresponding row is returned. If *reference* refers to a range of cells involving multiple rows, a vertical array of the corresponding rows as numbers is returned.

However, if the range of cells referred to by *reference* is not contiguous, `#REF!` is returned.

[*Example*:

`ROW()` results in 16, when the cell containing the formula is in row 16
`ROW(E17:G17)` results in 17
`ROW(E16:G17)` results in a vertical array containing 16 and 17, respectively

end example]

3.17.7.279 ROWS

Syntax:

`ROWS ( array )`

Description: Finds the number of rows corresponding to *array*.

Arguments:

Name	Type	Description
array	array, reference to a single cell, or a reference to a range of contiguous cells	A set of rows.

Return Type and Value: number – The number of rows corresponding to *array*.

However, if the range of cells referred to by *array* is not contiguous, `#NULL!` is returned.

[*Example*:

`ROWS(E16:H16)` results in 1
`ROWS(E16:G18)` results in 3
`ROWS({1,2;3,4})` results in 2

end example]

3.17.7.280 RSQ

Syntax:

`RSQ` (*known-ys* , *known-xs*)

Description: Computes the square of the Pearson product moment correlation coefficient through data points in known ys and known xs.

Mathematical Formula:

The equation for the Pearson product moment correlation coefficient, r, is:

$$r = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sqrt{\sum (x - \bar{x})^2 \sum (y - \bar{y})^2}}$$

where x and y are the sample means `AVERAGE(`*known-xs*`)` and `AVERAGE(`*known-ys*`)`.

Arguments:

Name	Type	Description
known-xs	number, name,	Designate a set of numeric data points. Logical values and

Name	Type	Description
	array, or reference to number, text, logical	text representations of numbers entered directly into the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.
known-ys	number, name, array, or reference to number, text, logical	Designate a set of numeric data points. Logical values and text representations of numbers entered directly into the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The square of the Pearson product moment correlation coefficient.

However, if

- *known-ys* and *known-xs* are empty or have a different number of data points, the return value is unspecified.
- *known-ys* and *known-xs* contain only one data point, the return value is unspecified.

[*Example*:

`RSQ({2,3,9,1,8,7,5},{6,5,11,7,5,4,4})` results in `0.057950192`

end example]

3.17.7.281 RTD

Syntax:

RTD (*progID* , [*rtd-server*] , *argument-list*)

Description: Retrieves data from a program in real-time. Periodically, this function returns new values and causes recalculation of the expression containing the call to it.

Arguments:

Name	Type	Description
progID	text	The name of the program from which the data is to be retrieved.
rtd-server	text	An optional string that is specific to the program with which RTD is communicating.
argument list	any	The presence and meaning of each *argument* in *argument-list* is specific to the program with which RTD is

Name	Type	Description
		communicating.

Return Type and Value: array – The set of values returned by the program with which RTD is communicating.

[*Example*: Consider a stockprice program that is called as follows:

```
RTD("stockprice.rtd","NASD","MSFT")
```

The result it returns—the price of the stock MSFT according to NASD—changes over time, often every few seconds.

The *rtd-server* program could also be written to accept multiple arguments, allowing calls like the following::

```
RTD("stockprice.rtd","NASD","MSFT","GOOG","AMZN")
```

where three stock values are requested.

end example]

3.17.7.282 SEARCH

Syntax:

SEARCH (*string-1* , *string-2* [, *start-pos*])

Description: Performs a case-insensitive search for the first occurrence of *string-1* in *string-2*, starting at character position *start-pos* within *string-2*. (SEARCH is intended for use with languages that use the single-byte character set (SBCS), whereas SEARCHB (§3.17.7.283) is intended for use with languages that use the double-byte character set (DBCS).)

Arguments:

Name	Type	Description
string-1	text	Designate the string to be searched for within the string designated by *string-2*. *string-1* can contain the following wildcard characters: question mark (?) and asterisk (*). A question mark matches any single character; an asterisk matches any sequence of characters. To search for an actual question mark or asterisk, that character shall be preceded by a tilde (~).
string-2	text	
start-pos	number	The number of the start position within *string-2* for which *string-1* is to be searched. The start position of the first character is 1. If omitted, a position of 1 shall be assumed. *start-pos* shall be at least 0.

Return Type and Value: number – The start position of the first occurrence of *string-1* in *string-2*, starting at character position *start-pos* within *string-2*. If *string-1* is an empty string, it shall always be found in any *string-2* at position *start-pos*, or at position 1 if *start-pos* is omitted.

However, if

- *string-1* is not found within *string-2*, #VALUE! is returned.
- *start-pos* designates a position outside *string-2*, #VALUE! is returned.

[*Example*:

SEARCH("de","abcdEF") results in 4
SEARCH("?c*e","abcdEF") results in 2

end example]

3.17.7.283 SEARCHB

Syntax:

SEARCHB (*string-1* , *string-2* [, *start-pos*])

Description: Performs a case-insensitive search for the first occurrence of *string-1* in *string-2*, starting at byte position *start-pos* within *string-2*. (SEARCHB is intended for use with languages that use the double-byte character set (DBCS), whereas SEARCH (§3.17.7.282) is intended for use with languages that use the single-byte character set (SBCS).)

Arguments:

Name	Type	Description
string-1	text	Designate the string to be searched for within the string designated by *string-2*. *string-1* can contain the following wildcard characters: question mark (?) and asterisk (*). A question mark matches any single character; an asterisk matches any sequence of characters. To search for an actual question mark or asterisk, that character shall be preceded by a tilde (~).
string-2	text	
start-pos	number	The number of the start position within *string-2* for which *string-1* is to be searched. The start position of the first byte is 1. If omitted, a position of 1 shall be assumed. *start-pos* shall be at least 0.

Return Type and Value: number – The start position of the first occurrence of *string-1* in *string-2*, starting at character position *start-pos* within *string-2*. If *string-1* is an empty string, it shall always be found in any *string-2* at position *start-pos*, or at position 1 if *start-pos* is omitted.

However, if

- *string-1* is not found within *string-2*, #VALUE! is returned.
- *start-pos* designates a position outside *string-2*, #VALUE! is returned.

[*Example*: Assuming 1-byte characters

SEARCHB("de","abcdEF") results in 4
SEARCHB("?c*e","abcdEF") results in 2

end example]

3.17.7.284 SECOND

Syntax:

SECOND (*time-value*)

Description: Computes the second for the date and/or time having the given *time-value*.

Arguments:

Name	Type	Description
time-value	number	The date and/or time whose second is to be computed. That date and/or time shall be expressed either as a serial value, in which case, its integer part is ignored, or as a *string-constant* having any valid date and/or time format, in which case, any date information shall be ignored.

Return Type and Value: number – The second for the date and/or time having the given *time-value*.

However, if *time-value* is out of range for the current date base value, #NUM! is returned.

[*Example*:

SECOND(DATE(2006,2,26)+TIME(2,10,20)) results in 20
SECOND(TIME(22,56,34)) results in 34
SECOND(0) results in 0, since serial value 0 represents 00:00:00
SECOND(10.5) results in 0, since serial value .5 represents 12:00:00
SECOND("22-Oct-2001 10:53:12") results in 12
SECOND("10:53:12 pm") results in 12
SECOND("22:53:12") results in 12

end example]

3.17.7.285 SERIESSUM

Syntax:

`SERIESSUM ( input-value , initial-power , step , coefficients )`

Description: Computes the sum of a power series.

Mathematical Formula:

The sum of a power series is based on the formula:

$$\mathrm{SERIES}(x,n,m,a) = a_1 x^{n} + a_2 x^{(n+m)} + a_3 x^{(n+2m)} + \ldots + a_i x^{(n+(i-1)m)}$$

where *input-value* = *x*, *initial-power* = *n*, *step* = *m*, and *coefficients* is the set of *a* values.

Arguments:

Name	Type	Description
input-value	number	The input value to the power series;
initial-power	number	The initial power to which *input-value* is to be raised.
step	number	The step by which to increase *initial-power* for each term in the series;
coefficients	reference	A set of coefficients by which each successive power of *input-value* is multiplied. The number of values in *coefficients* determines the number of terms in the power series.

Return Type and Value: number – The sum of a power series.

[*Example*: Given the following data:

	A
1	1
2	=-1/FACT(2)
3	=1/FACT(4)
4	=-1/FACT(6)

`SERIESSUM(PI()/4,0,2,A1:A4)` results in `0.707103`, an approximation to the cosine of π/4 radians

end example]

3.17.7.286 SIGN

Syntax:

```
SIGN ( x )
```

Description: Determines the sign of *x*.

Arguments:

Name	Type	Description
x	number	The number whose sign is to be determined.

Return Type and Value: number – 1 if *x* is positive, 0 if *x* is 0, and -1 if *x* is negative.

[*Example*:

`SIGN(10.5)` results in `1`
`SIGN(0)` results in `0`
`SIGN(-5.4)` results in `-1`

end example]

3.17.7.287 SIN

Syntax:

```
SIN ( x )
```

Description: Computes the sine of *x*.

Arguments:

Name	Type	Description
x	number	The number whose sine is to be computed.

Return Type and Value: number – The sine of *x*.

[*Example*:

`SIN(-1)` results in `-0.841470985`
`SIN(0)` results in `0`
`SIN(1)` results in `0.841470985`

end example]

3.17.7.288 SINH

Syntax:

`SINH ( x )`

Description: Computes the hyperbolic sine of *x*.

Arguments:

Name	Type	Description
x	number	The number whose hyperbolic sine is to be computed.

Return Type and Value: number – The hyperbolic sine of *x*.

However, if the magnitude of *x* is too large for the result to be represented, `#NUM!` is returned.

[*Example*:|

`SINH(1)` results in `1.175201194`
`SINH(10)` results in `11013.23287`
`SINH(100)` results in `1.34406E+43`

end example]

3.17.7.289 SKEW

Syntax:

`SKEW ( argument-list )`

Description: Computes the skewness of a distribution. [*Note*: Skewness characterizes the degree of asymmetry of a distribution around its mean. Positive skewness indicates a distribution with an asymmetric tail extending toward more positive values. Negative skewness indicates a distribution with an asymmetric tail extending toward more negative values. *end note*]

Mathematical Formula:

$$\frac{n}{(n-1)(n-2)}\sum\left(\frac{x_j-\bar{x}}{s}\right)^3$$

Arguments:

Name	Type	Description
argument-list	logical, number, name, text, array, reference	The *argument*s in *argument-list* designate the numbers for which the skewness is to be computed. Logical values and text representations of numbers that are entered

Name	Type	Description
		directly into the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The skewness of a distribution.

However, if

- There are fewer than three data points, the return value is unspecified.
- The sample standard deviation is zero, the return value is unspecified.

[*Example*:

`SKEW(3,4,5,2,3,4,5,6,4,7)` results in `0.359543071`

end example]

3.17.7.290 SLN

`SLN ( cost , salvage , life )`

Description: Computes the straight-line depreciation of an asset for one period.

Arguments:

Name	Type	Description
cost	number	The number *cost* is the initial cost of the asset.
salvage	number	The value at the end of the depreciation. (This is sometimes called the salvage value of the asset.)
life	number	The number of periods over which the asset is being depreciated. (This is sometimes called the useful life of the asset.)

Return Type and Value: number – The straight-line depreciation of an asset for one period.

[*Example*:

`SLN(30000,7500,10)` results in `2,250.00`

end example]

3.17.7.291 SLOPE

Syntax:

SLOPE (*known-ys* , *known-xs*)

Description: Computes the slope of the linear regression line through data points in known ys and known xs. The slope is the vertical distance divided by the horizontal distance between any two points on the line, which is the rate of change along the regression line.

Mathematical Formula:

$$b = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sum (x - \bar{x})^2}$$

where x and y are the sample means AVERAGE(*known-xs*) and AVERAGE(*known-ys*).

Arguments:

Name	Type	Description
known-xs	number, name, array, or reference to number, text, logical	Designate a set of numeric dependent data points. Logical values and text representations of numbers entered directly into the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.
known-ys	number, name, array, or reference to number, text, logical	Designate a set of numeric independent data points. Logical values and text representations of numbers entered directly into the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The slope of the linear regression line through data points in known ys and known xs.

However, if

- *known-ys* and *known-xs* are empty, the return value is unspecified.
- *known-ys* and *known-xs* have a different number of data points, the return value is unspecified.

[*Example*:

SLOPE({2,3,9,1,8,7,5},{6,5,11,7,5,4,4}) results in 0.305555556

end example]

3.17.7.292 SMALL

Syntax:

SMALL (*array* , *k*)

Description: Computes the k^{th} smallest value in a data set.

Arguments:

Name	Type	Description
array	array, reference	The set of numbers from which the k^{th}-smallest value is to be determined.
k	number	The position (from the smallest) in the array or cell range of data to return.

Return Type and Value: number – The k^{th} smallest value in a data set.

However, if

- *array* is empty, the return value is unspecified.
- $k \leq 0$ or k is greater than the number of data points, #NUM! is returned.

[*Example*:

SMALL({3,5,3,5,4;4,2,4,6,7},3) results in 3
SMALL({3,5,3,5,4;4,2,4,6,7},7) results in 5

end example]

3.17.7.293 SQRT

Syntax:

SQRT (*x*)

Description: Computes the positive square root of x.

Arguments:

Name	Type	Description
x	number	The number whose positive root is to be found.

Return Type and Value: number – The positive square root of x.

However, if x is negative, #NUM! is returned.

[*Example*:

SQRT(2) results in 1.414213562
SQRT(5) results in 2.236067977

end example]

3.17.7.294 SQRTPI

Syntax:

SQRTPI (*x*)

Description: Computes the positive square root of $x \times \pi$.

Arguments:

Name	Type	Description
x	number	The number, which when multiplied by π, whose positive root is to be found.

Return Type and Value: number – The positive square root of $x \times \pi$.

However, if *x* is negative, #NUM! is returned.

[*Example*:

SQRTPI(1) results in 1.772453851
SQRTPI(2) results in 2.506628275

end example]

3.17.7.295 STANDARDIZE

Syntax:

STANDARDIZE (*x* , *mean* , *standard-dev*)

Description: Computes a normalized value from a distribution characterized by *mean* and *standard-dev*.

Mathematical Formula:

$$Z = \frac{X - \mu}{\sigma}$$

Arguments:

Name	Type	Description

Name	Type	Description
x	number	The number whose value is to be normalized.
mean	number	The the arithmetic mean of the distribution.
standard-dev	number	The standard deviation of the distribution.

Return Type and Value: number – A normalized value from a distribution.

However, if *standard-dev* ≤ 0, `#NUM!` is returned.

[*Example*:

`STANDARDIZE(42,40,1.5)` results in `1.333333333`

end example]

3.17.7.296 STDEV

Syntax:

`STDEV` (*argument-list*)

Description: Makes an estimate of the standard deviation based on a sample, using the "unbiased" or "n-1" method. [*Note*: STDEV assumes that its arguments are a sample of the population. If the data represents the entire population, STDEVP should be used instead. If logical values and text representations of numbers in a reference are to be included as part of the calculation, use STDEVA instead. *end note*]

Mathematical Formula:

$$\sqrt{\frac{\sum(x-\bar{x})^2}{(n-1)}}$$

where x is the sample mean `AVERAGE(`*argument-1*`,` *argument-2*`,…,` *argument-n*`)` and n is the sample size.

Arguments:

Name	Type	Description
argument-list	logical, number, name, text, array, reference	The *argument*s in *argument-list* designate the numbers that are samples of the population. *argument-list* can also be an array of numbers. Logical values and text representations of numbers that are entered directly into the list of arguments are included. If an argument is an array or reference, only numbers in that array or reference are included. Empty cells, logical values, text, or error values in the array or reference are ignored.

Return Type and Value: number – An estimate of the standard deviation based on a sample.

[*Example*:

STDEV(123,134,143,173,112,109) results in 23.72902583

end example]

3.17.7.297 STDEVA

Syntax:

STDEVA (*argument-list*)

Description: Makes an estimate of the standard deviation based on a sample, using the "unbiased" or "n-1" method. [*Note*: STDEVA assumes that its arguments are a sample of the population. If the data represents the entire population, STDEVPA should be used instead. If logical values and text representations of numbers in a reference are to be excluded as part of the calculation, use STDEV instead. *end note*]

Mathematical Formula:

$$\sqrt{\frac{\sum(x-\bar{x})^2}{(n-1)}}$$

where x is the sample mean AVERAGE(*argument-1*, *argument-2*,…, *argument-n*) and n is the sample size.

Arguments:

Name	Type	Description
argument-list	logical, number, name, text, array, reference. The argument list can also be an array of numbers.	The *argument*s in *argument-list* designate the numbers that are samples of the population. Arguments that contain TRUE evaluate as 1; arguments that contain text or FALSE evaluate as zero. If an argument is an array or reference, only values in that array or reference are used. Empty cells and text values in the array or reference are ignored.

Return Type and Value: number – An estimate of the standard deviation based on a sample.

[*Example*:

STDEVA(123,134,143,173,112,109) results in 23.72902583

end example]

3.17.7.298 STDEVP

Syntax:

STDEVP (*argument-list*)

Description: Computes the standard deviation of an entire population, using the "biased" or "n" method. [*Note*: STDEVP assumes that its arguments are the total population. If the data represents a population sample only, STDEVA should be used instead. If logical values and text representations of numbers in a reference are to be included as part of the calculation, use STDEVPA instead. *end note*]

Mathematical Formula:

$$\sqrt{\frac{\sum(x-\bar{x})^2}{n}}$$

where x is the sample mean AVERAGE(*argument-1*, *argument-2*,…, *argument-n*) and n is the sample size.

Arguments:

Name	Type	Description
argument-list	logical, number, name, text, array, reference. The argument list can also be an array of numbers.	The *argument*s in *argument-list* designate the numbers that are the members of the population. Logical values, and text representations of numbers that are entered directly into the list of arguments are included. If an argument is an array or reference, only numbers in that array or reference are included. Empty cells, logical values, text, or error values in the array or reference are ignored.

Return Type and Value: number – The standard deviation of an entire population.

[*Example*:

STDEVP(123,134,143,173,112,109) results in 21.66153785

end example]

3.17.7.299 STDEVPA

Syntax:

STDEVPA (*argument-list*)

Description: Computes the standard deviation of an entire population, using the "biased" or "n" method. [*Note*: STDEVPA assumes that its arguments are the total population. If the data represents a population sample only,

STDEVA should be used instead. If logical values and text representations of numbers in a reference are to be excluded as part of the calculation, use STDEVP instead. *end note*]

Mathematical Formula:

$$\sqrt{\frac{\sum(x-\bar{x})^2}{n}}$$

where x is the sample mean AVERAGE(*argument-1*, *argument-2*,…, *argument-n*) and n is the sample size.

Arguments:

Name	Type	Description
argument-list	logical, number, name, text, array, reference. The argument list can also be an array of numbers.	The *argument*s in *argument-list* designate the numbers that are the members of the population.

Return Type and Value: number – The standard deviation of an entire population.

Arguments can be numbers; names, arrays, or references that contain numbers; text representations of numbers; or logical values, in a reference. Text representations of numbers that are entered directly into the list of arguments are included. Arguments that contain TRUE evaluate as 1; arguments that contain text or FALSE evaluate as zero. If an argument is an array or reference, only values in that array or reference are used. Empty cells and text values in the array or reference are ignored.

[*Example*:

STDEVPA(123,134,143,173,112,109) results in 21.66153785

end example]

3.17.7.300 STEYX

Syntax:

STEYX (*known-ys* , *known-xs*)

Description: Computes the standard error of the predicted *y*-value for each *x* in the regression. The standard error is a measure of the amount of error in the prediction of *y* for an individual *x*.

Mathematical Formula:

$$\sqrt{\frac{1}{(n-2)}\left[\sum(y-\bar{y})^2-\frac{\left[\sum(x-\bar{x})(y-\bar{y})\right]^2}{\sum(x-\bar{x})^2}\right]}$$

where x and y are the sample means AVERAGE(*known-xs*) and AVERAGE(*known-ys*), and n is the sample size.

Arguments:

Name	Type	Description
known-xs	number, name, array, or reference to number, text, logical	Designate a set of numeric dependent data points. Logical values and text representations of numbers entered directly into the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.
known-ys	number, name, array, or reference to number, text, logical	Designate a set of numeric independent data points. Logical values and text representations of numbers entered directly into the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The standard error of the predicted y-value for each *x* in the regression.

However, if

- *known-ys* and *known-xs* have a different number of data points, the return value is unspecified.
- *known-ys* and *known-xs* are empty or have less than three data points, the return value is unspecified.

[*Example*:

STEYX({2,3,9,1,8,7,5},{6,5,11,7,5,4,4}) results in 3.30571895

end example]

3.17.7.301 SUBSTITUTE

Syntax:

SUBSTITUTE (*string* , *old-string* , *new-string* [, *occurence*])

Description: Produces a new string that is *string* with one or all occurrences of *old-string* replaced by *new-string*.

Arguments:

Name	Type	Description

Name	Type	Description
string	text	Designates a string.
old-string	text	Designates a string.
new string	text	Designates a string.
occurence	number	The occurrence number of the *old-string* characters within *string-1* that is to be replaced by the string designated by *new-string*. If omitted, all occurrences of *old-string* characters shall be replaced.

Return Type and Value: text – A string that is *string* with one or all occurrences of *old-string* replaced by *new-string*.

However, if *occurance* < 0, #VALUE! is returned.

[*Example*:

`SUBSTITUTE("abcaaabca","a","xx")` results in `xxbcxxxxxxbcxx`
`SUBSTITUTE("abcaaabca","a","",10)` results in `bcbc`
`SUBSTITUTE("abcaaabca","a","xx",3)` results in `abcaxxabca`

end example]

3.17.7.302 SUBTOTAL

Syntax:

SUBTOTAL (*function-number* , *argument-list*)

Description: Computes a value using the function designated by *function-number*, using the *argument*s in *argument-list*.

Arguments:

Name	Type	Description
function-number	number	Indicates the function to be called, as shown in the table below.
argument-list	number	Each *argument* in *argument-list* is passed to the called function, in the order specified. That shall be no more than 254 *argument*s.

function-number (includes hidden values)	*function-number* (excludes hidden values)	Function
1	101	AVERAGE

function-number (includes hidden values)	*function-number* (excludes hidden values)	Function
2	102	COUNT
3	103	COUNTA
4	104	MAX
5	105	MIN
6	106	PRODUCT
7	107	STDEV
8	108	STDEVP
9	109	SUM
10	110	VAR
11	111	VARP

If any argument contains a SUBTOTAL function call, that call shall be ignored to avoid double counting.

For the *function-number* values 1–11, the values of hidden rows are included. For the *function-number* values 101–111, the values of hidden rows are excluded.

The SUBTOTAL function shall ignore any rows that are not included in the result of a filter, regardless of which *function-number* value is used.

The SUBTOTAL function is designed for columns of data, or vertical ranges. It is not designed for rows of data, or horizontal ranges. [*Example*: When a horizontal range is subtotaled using a *function-number* of 101 or greater, hiding a column does not affect the subtotal. However, hiding a row in a subtotal of a vertical range does affect the subtotal. *end example*]

Return Type and Value: number – The result from calling the function designated by *function-number*, using the *argument*s in *argument-list*.

However, if *function-number* does not have one of the values specified above, `#NUM!` is returned.

[*Example*:

`SUBTOTAL(2,E5:E15)` counts the number of values in the cell range E5:E15, including hidden values
`SUBTOTAL(4,E5:E15)` finds the maximum value of the values in the cell range E5: E15, including hidden values
`SUBTOTAL(106,E5:E15)` finds the product of the values in the cell range E5: E15, excluding hidden values

end example]

3.17.7.303 SUM

Syntax:

SUM (*argument-list*)

Description: Adds the numeric values of *argument*s in *argument-list*.

Arguments:

Name	Type	Description
argument-list	logical, number, name, text, array, reference.	The *argument*s in *argument-list* designate the numeric values to be added. Arguments that are numbers, logical values, or text representations of numbers shall be counted. If an argument is an array or reference, only numbers in that array or reference shall be counted. Empty cells, logical values, and text in the array or reference shall be ignored.

Return Type and Value: number – The sum of the values of its arguments.

[*Example*:

SUM(1,2,3,4,5) results in 15
SUM({1,2;3,4}) results in 10
SUM({1,2,3,4,5},6,"7") results in 28
SUM({1,"2",TRUE,4}) results in 5, as the logical value and numeric text are ignored

end example]

3.17.7.304 SUMIF

Syntax:

SUMIF (*cell-range* , *selection-criteria* [, *sum-range*])

Description: Applies selection criteria on the values in one range of cells and sums the values of the cells in a corresponding range.

Arguments:

Name	Type	Description
cell-range	reference	Designates the range of cells to be inspected.
selection-criteria	number, expression, reference, text	Defines which cells will be counted. In the case of text, *selection-criteria* can consist of any comparison operator followed by the operand against which each cell's value is to be compared. *selection-criteria* can include one or more wildcard characters, question mark (?) and asterisk (*). A question mark matches any single character; an asterisk matches any sequence of

Name	Type	Description
		characters. To search for a question mark, asterisk, or tilde character, prefix that character with a tilde (~).
sum-range	reference	If present, *sum-range* designates the cells whose values are summed. In this case, *sum-range* does not have to have the same size and shape as *cell-range*. The actual cells that are added are determined by using the top, left cell in *sum-range* as the beginning cell, and then including cells that correspond in size and shape to *cell-range*. If omitted, *cell-range* also designates the cells whose values are summed.

Return Type and Value: number – The sum of the cells corresponding to those selected.

[*Example*: Given that A1, B1, C1, and D1, respectively, contain the values 3, 10, 7, and 10

`SUMIF(A1:D1,"=10")` results in 20
`SUMIF(A1:D1,">5")` results in 27
`SUMIF(A1:D1,"<>10")` results in 10

Given that A2, B2, C2, and D2, respectively, contain the values `apples`, `melons`, 10, and 15
`SUMIF(A2:B2,"*es",C2:D2)` results in 10

end example]

3.17.7.305 SUMIFS

Syntax:

```
SUMIFS ( sum-range , cell-range-1 , selection-criteria-1
   [ , cell-range-2 , selection-criteria-2 [ , … ] ] )
```

Description: Adds the cells in a range that meet multiple criteria.

Arguments:

Name	Type	Description
sum-range	reference	Designates the cells whose values are summed. In this case, *sum-range* does not have to have the same size and shape as *cell-range-1* through *cell-range-n*. The actual cells that are added are determined by using the top, left cell in *sum-range* as the beginning cell, and then including cells that correspond in size and shape to *cell-range-1* through *cell-range-n*. Each cell in *sum-range* is summed only if all of the corresponding criteria specified are true for that cell. Cells in *sum-range* that contain TRUE

Name	Type	Description
		evaluate to 1; cells in sum-range that contain FALSE evaluate to 0.
cell-range-1	reference	Designates the first range of cells to be inspected.
selection-criteria-1	number, expression, reference, text	Specifies the criteria for the first range of cells that will be counted. In the case of text, *selection-criteria-1* can consist of any comparison operator followed by the operand against which each cell's value is to be compared. *selection-criteria-1* can include one or more wildcard characters, question mark (?) and asterisk (*). A question mark matches any single character; an asterisk matches any sequence of characters. To search for a question mark, asterisk, or tilde character, prefix that character with a tilde (~).
cell-range-n	reference	The optional arguments *selection-criteria-2* through *selection-criteria-n* have corresponding arguments *cell-range-2* through *cell-range-n*, and have the same semantics as *selection-criteria-1* and *cell-range-1*, respectively.
selection-criteria-n	number, expression, reference, text	

Return Type and Value: number – The sum of the cells corresponding to those selected.

[*Example*: Given the following data:

	A	B	C	D
1	Sales Person	Tables	Chairs	Desks
2	Emilio	34	85	97
3	Julie	353	23	18
4	Hans	13	67	14
5	Frederique	0	98	0

`SUMIFS(B2:C5,A2:A5,"=Julie")` results in `353` (the sum of the number of tables and chairs sold by Julie)

`SUMIFS(B2:B5,A2:A5,"=Julie",A2:A5,"=Hans")` results in `0` (the sum of the number of tables sold by Julie and Hans)

`SUMIFS(B2:B5,A3,"=Julie",A4,"=Hans")` results in `34` (the sum of the the number of tables sold by Julie and Hans)

`SUMIFS(B2:D5,A2:A5,"<>Emilio")` results in `768` (the sum of the number of tables, chairs, and desks sold by all sales persons except Emilio)

end example]

3.17.7.306 SUMPRODUCT

Syntax:

SUMPRODUCT (*argument-list*)

Description: Multiplies the corresponding elements in the array *argument*s in *argument-list*, and returns the sum of those products. An array element that is not numeric is treated as if it contained 0.

Arguments:

Name	Type	Description
argument-list	array of numbers	The *argument*s in *argument-list* designate the numeric values to be multiplied.

Return Type and Value: number – The sum of the products of the corresponding elements in the *argument*s in *argument-list*.

However, if the array arguments do not have the same dimensions, #NUM! is returned.

[*Example*:

SUMPRODUCT({2,3}) results in 5
SUMPRODUCT({2,3},{4,5}) results in 23
SUMPRODUCT({2,3},{4,5},{2,2}) results in 46
SUMPRODUCT({2,3;4,5},{2,2;3,4}) results in 42

end example]

3.17.7.307 SUMSQ

Syntax:

SUMSQ (*argument-list*)

Description: Adds the squares of *argument*s in *argument-list*.

Arguments:

Name	Type	Description
argument-list	logical, number, name, text, array, reference	The *argument*s in *argument-list* designate the values whose squares are to be summed. Arguments that are numbers, logical values, or text representations of numbers shall be counted. If an argument is an array or

Name	Type	Description
		reference, only numbers in that array or reference shall be counted. Empty cells, logical values, and text in the array or reference shall be ignored.

Return Type and Value: number – The sum of the squares of its arguments.

[*Example*:

SUMSQ(2) results in 4
SUMSQ(2.5,-3.6)) results in 19.21
SUMSQ({2.5,-3.6)},2.4) results in 24.97

end example]

3.17.7.308 SUMX2MY2

Syntax:

SUMX2MY2 (*array-1* , *array-2*)

Description: Computes the sum of the difference of squares of the corresponding numerical elements in two arrays designated by *array-1* and *array-2*.

Mathematical Formula:

$$\mathrm{SUMX2MY2} = \sum (x^2 - y^2)$$

where *array-1* contains the *x* values, and *array-2* contains the *y* values.

Arguments:

Name	Type	Description
array-1	array, reference	Designated the arrays to be operated on. If an argument contains text, logical values, or empty cells, those elements shall be ignored; however, cells with the value 0 shall be included.
array-2		

Return Type and Value: number – The sum of the difference of squares of the corresponding elements in two arrays designated by *array-1* and *array-2*.

However, if *array-1* and *array-2* have a different number of values, the return value is unspecified.

[*Example*:

`SUMX2MY2({2,3,9,1,8,7,5},{6,5,11,7,5,4,4})` results in 55
`SUMX2MY2({2,3,9;1,8,7},{6,5,11;7,5,4})` results in -64
end example]

3.17.7.309 SUMX2PY2

Syntax:

`SUMX2PY2 ( array-1 , array-2 )`

Description: Computes the sum of the sum of the squares of the corresponding numerical elements in two arrays designated by *array-1* and *array-2*.

Mathematical Formula:

$$SUMX2PY2 = \sum (x^2 + y^2)$$

where *array-1* contains the *x* values, and *array-2* contains the *y* values.

Arguments:

Name	Type	Description
array-1	array, reference	Designated the arrays to be operated on. If an argument contains text, logical values, or empty cells, those elements shall be ignored; however, cells with the value 0 shall be included.
array-2		

Return Type and Value: number – The sum of the sum of the squares of the corresponding elements in two arrays designated by *array-1* and *array-2*.

However, if *array-1* and *array-2* have a different number of values, the return value is unspecified.

[*Example*:

`SUMX2PY2({2,3,9,1,8,7,5},{6,5,11,7,5,4,4})` results in 521
`SUMX2PY2({2,3,9;1,8,7},{6,5,11;7,5,4})` results in 480

end example]

3.17.7.310 SUMXMY2

Syntax:

`SUMXMY2 ( array-1 , array-2 )`

Description: Computes the sum of the squares of the difference between corresponding numerical elements in two arrays designated by *array-1* and *array-2*.

Mathematical Formula:

$$SUMXMY2 = \sum (x - y)^2$$

where *array-1* contains the *x* values, and *array-2* contains the *y* values.

Arguments:

Name	Type	Description
array-1	array, reference	Designated the arrays to be operated on. If an argument contains text, logical values, or empty cells, those elements shall be ignored; however, cells with the value 0 shall be included.
array-2		

Return Type and Value: number – The sum of the squares of the difference between the corresponding elements in two arrays designated by *array-1* and *array-2*.

However, if *array-1* and *array-2* have a different number of values, the return value is unspecified.

[*Example*:

`SUMXMY2({2,3,9,1,8,7,5},{6,5,11,7,5,4,4})` results in 79
`SUMXMY2({2,3,9;1,8,7},{6,5,11;7,5,4})` results in 78

end example]

3.17.7.311 SYD

SYD (*cost* , *salvage* , *life* , *per*)

Description: Computes the sum-of-years' digits depreciation of an asset for a specified period.

Mathematical Formula:

$$SYD = \frac{(cost - salvage) * (life - per + 1) * 2}{(life)(life + 1)}$$

Arguments:

Name	Type	Description
cost	number	The initial cost of the asset.
salvage	number	The value at the end of the depreciation. (This is sometimes called the salvage value of the asset.)

Name	Type	Description
life	number	The number of periods over which the asset is being depreciated. (This is sometimes called the useful life of the asset.)
per	number	The period and shall have the same units as *life*.

Return Type and Value: number – The sum-of-years' digits depreciation of an asset for a specified period.

[*Example*:

`SYD(30000,7500,10,1)` results in `4,090.91`
`SYD(30000,7500,10,10)` results in `409.09`

end example]

3.17.7.312 T

Syntax:

`T ( value )`

Description: Retrieves the text represented by *value*.

Arguments:

Name	Type	Description
value	any	The value to be encoded in text. No conversion shall take place on an argument passed to this function.

Return Type and Value: text – *value* if *value* designates text; otherwise, `""`. [*Note*: T cannot differentiate between text that is an empty string, and any *value* of non-text type. *end note*]

[*Example*:

`T("Hello")` results in `Hello`
`LEN(T(123))` results in `0`
`LEN(T(TRUE))` results in `0`

end example]

3.17.7.313 TAN

Syntax:

`TAN ( x )`

Description: Computes the tangent of *x*.

Arguments:

Name	Type	Description
x	number	The number whose tangent is to be computed.

Return Type and Value: number – The tangent of *x*.

[*Example*:

`TAN(-1)` results in `-1.557407725`
`TAN(0)` results in `0`
`TAN(1)` results in `1.557407725`

end example]

3.17.7.314 TANH

Syntax:

`TANH ( x )`

Description: Computes the hyperbolic tangent of *x*.

Arguments:

Name	Type	Description
x	number	The number whose hyperbolic tangent is to be computed.

Return Type and Value: number – The hyperbolic tangent of *x*

[*Example*:

`TANH(-1)` results in `-0.761594156`
`TANH(0)` results in `0`
`TANH(1)` results in `0.761594156`

end example]

3.17.7.315 TBILLEQ

`TBILLEQ ( settlement , maturity , discount )`

Description: Computes the bond-equivalent yield for a U.S. Treasury bill.

Arguments:

Name	Type	Description
settlement	number	The Treasury bill's settlement date. Any time information in the date is ignored.
maturity	number	The Treasury bill's maturity date. Any time information in the date is ignored.
discount	number	The Treasury bill's discount rate.

Return Type and Value: number – The bond-equivalent yield for a U.S. Treasury bill.

However, if

- *settlement* or *maturity* is out of range for the current date base value, `#NUM!` is returned.
- *settlement* > *maturity*, `#NUM!` is returned.
- *maturity* is more than one year after *settlement*, `#NUM!` is returned.
- *discount* ≤ 0, `#NUM!` is returned.

[*Example*:

`TBILLEQ(DATE(2008,3,31),DATE(2008,6,1),0.0914)` results in `9.4151%`

end example]

3.17.7.316 TBILLPRICE

`TBILLPRICE ( settlement , maturity , discount )`

Description: Computes the price per $100 face value for a U.S. Treasury bill.

Mathematical Formula:

$$TBILLPRICE = 100 \times (1 - \frac{discount \times DSM}{360})$$

where:

DSM = number of days from settlement to maturity, excluding any maturity date that is more than one calendar year after the settlement date.

Arguments:

Name	Type	Description
settlement	number	The Treasury bill's settlement date. Any time information in the date is ignored.

Name	Type	Description
maturity	number	The Treasury bill's maturity date. Any time information in the date is ignored.
discount	number	The Treasury bill's discount rate.

Return Type and Value: number – The price per $100 face value for a U.S. Treasury bill.

However, if

- *settlement* or *maturity* is out of range for the current date base value, `#NUM!` is returned.
- *settlement* > *maturity*, `#NUM!` is returned.
- *maturity* is more than one year after *settlement*, `#NUM!` is returned.
- *discount* ≤ 0, `#NUM!` is returned.

[*Example*:

`TBILLPRICE(DATE(2008,3,31),DATE(2008,6,1),0.09)` results in `98.4500`

end example]

3.17.7.317 TBILLYIELD

`TBILLYIELD ( `*settlement*` , `*maturity*` , `*pr*` )`

Description: Computes the yield for a U.S. Treasury bill.

Mathematical Formula:

$$TBILLYIELD = \frac{100 - pr.}{pr.} \times \frac{360}{DSM}$$

where:

DSM = number of days from settlement to maturity, excluding any maturity date that is more than one calendar year after the settlement date.

Arguments:

Name	Type	Description
settlement	number	The Treasury bill's settlement date. Any time information in the date is ignored.
maturity	number	The Treasury bill's maturity date. Any time information in the date is ignored.
pr	number	The Treasury bill's price per $100 face value.

Return Type and Value: number – The yield for a U.S. Treasury bill.

However, if

- *settlement* or *maturity* is out of range for the current date base value, #NUM! is returned.
- *settlement* > *maturity*, #NUM! is returned.
- *maturity* is more than one year after *settlement*, #NUM! is returned.
- *pr* ≤ 0, #NUM! is returned.

[*Example*:

TBILLYIELD(DATE(2008,3,31),DATE(2008,6,1),98.45) results in 9.1417%

end example]

3.17.7.318 TDIST

Syntax:

TDIST (*x* , *degrees-freedom* , *distribution-tails*)

Description: Computes the Percentage Points (probability) for the Student t-distribution where a numeric value, *x*, is a calculated value of *t* for which the Percentage Points are to be computed.

Mathematical Formula:

If *distribution-tails* = 1, TDIST = P(X>x), where X is a random variable that follows the t-distribution.

If *distribution-tails* = 2, TDIST = P(|X| > x) = P(X > x or X < -x)

Arguments:

Name	Type	Description
x	number	The value at which to evaluate the distribution.
degrees-freedom	number	The number of degrees of freedom, truncated to an integer.
distribution-tails	number	The number of distribution tails to return, truncated to an integer. If 1, TDIST returns the one-tailed distribution. If 2, TDIST returns the two-tailed distribution.

Return Type and Value: number – The Percentage Points (probability) for the Student t-distribution.

However, if

- *degrees-freedom* < 1, #NUM! is returned.
- *tails* has any value other than 1 or 2, #NUM! is returned.

- $x < 0$, `#NUM!` is returned.

[*Example*:

`TDIST(1.959999998,60,1)` results in `0.027322464`
`TDIST(1.959999998,60,2)` results in `0.054644927`

end example]

3.17.7.319 TEXT

Syntax:

`TEXT ( value , format )`

Description: Produces a string containing *value* formatted according to *format*.

Arguments:

Name	Type	Description
value	number	The number that is to be formatted.
format	text	Designates the number, currency, date, or time format to be used. (See §3.8.31 for the set of formats.)

Return Type and Value: text – The string containing *number* formatted according to *format*.

[*Example*:

`TEXT(1234.567,"$0.00")` results in `$1234.57`
`TEXT(.125,"$0.0%")` results in `12.5%`
`TEXT(1234.567,"YYYY-MM-DD HH:MM:SS")` results in `1903-05-18 13:36:29` in the 1900 date-base system.

end example]

3.17.7.320 TIME

Syntax:

`TIME ( hour , minute , second )`

Description: Computes the serial value for the given time.

Arguments:

Name	Type	Description

Name	Type	Description
hour	number	A number in the range 0–32767, inclusive, truncated to integer, that represents the hour. Any value greater than 23 shall be divided by 24 and the remainder shall be treated as the hour value.
minute	number	A number in the range 0–32767, inclusive, truncated to integer, that represents the minute. Any value greater than 59 shall be converted to the corresponding number of hours and minutes.
second	number	A number in the range 0–32767, inclusive, truncated to integer, that represents the second. Any value greater than 59 shall be converted to the corresponding number of hours, minutes, and seconds.

Return Type and Value: number – The serial value for the given time, as a value greater than or equal to 0 and less than or equal to 1.

However, if *hour*, *minute*, or *second* are out of range, `#NUM!` is returned.

[*Example*: The following serial values are displayed with 16 decimal places.

`TIME(0,0,0)` results in a serial value of `0.0000000000000000`
`TIME(0,0,1)` results in a serial value of `0.0000115740740741`
`TIME(0,0,2)` results in a serial value of `0.0000231481481481`
`TIME(0,0,20)` results in a serial value of `0.0002314814814815`
`TIME(2,3,20)` results in a serial value of `0.0856481481481481`
`TIME(12,0,0)` results in a serial value of `0.5000000000000000`
`TIME(23,59,59)` results in a serial value of `0.9999884259259260`
`TIME(26,120,240)` results in a serial value of `0.1694444444444450`

end example]

3.17.7.321 TIMEVALUE

Syntax:

`TIMEVALUE` (*date-time-string*)

Description: Computes the serial value of the date and/or time represented by the string *date-time-string*.

Arguments:

Name	Type	Description
date-time-string	text	The date and/or time whose serial value is to be computed. *date-time-string* can have any valid date

Name	Type	Description
		and/or time format. Any date information in *date-time-string* shall be ignored.

Return Type and Value: number – The serial value of the date and/or time represented by the string *date-time-string*.

However, if *date-time-string* is ill-formed, `#VALUE!` is returned.

[*Example*: The following serial values are displayed with 16 decimal places.

`TIMEVALUE("10:02:34 ")` results in `0.4184490740740740`
`TIMEVALUE("01-Feb-2006 10:15:29 AM")` results in `0.4274189814823330`
`TIMEVALUE("22:02")` results in `0.9180555555555560`

end example]

3.17.7.322 TINV

Syntax:

`TINV ( probability , degrees-freedom )`

Description: Computes the t-value of the Student's t-distribution as a function of the probability and the degrees of freedom.

Arguments:

Name	Type	Description
probability	number	A probability associated with the two-tailed Student's t-distribution.
degrees-freedom	number	The number of degrees of freedom with which to characterize the distribution, truncated to an integer.

Return Type and Value: number – The t-value of the Student's t-distribution.

However, if

- *probability* < 0 or *probability* > 1, `#NUM!` is returned.
- *degrees-freedom* < 1, `#NUM!` is returned.

[*Example*:

`TINV(0.054644927,60)` results in `1.95999999`

end example]

3.17.7.323 TODAY

Syntax:

```
TODAY ( )
```

Description: Computes the serial value of the current date, taking into account the current date base value.

Arguments: None.

Return Type and Value: number – The serial value of the current date.

[*Example*:

On February 25, 2006, `TODAY()` results in `38773` for the 1900 date base system, or `37311` for the 1904 date base system

end example]

3.17.7.324 TRANSPOSE

Syntax:

```
TRANSPOSE ( array )
```

Description: Creates a new array that is the transpose of an existing array, by copying the first row of the existing array to the first column of the new array, the second row of the existing array as the second column of the new array, and so on. The formula containing the call to TRANSPOSE shall be an array formula in a range that has the same number of rows and columns, respectively, as *array* has columns and rows.

Arguments:

Name	Type	Description
array	array, reference	The set of values to be transposed.

Return Type and Value: array – The new array.

[*Example*:

`TRANSPOSE({10,20,30})` results in the array `{10;20;30}`

end example]

3.17.7.325 TREND

Syntax:

TREND (*known-ys* [, [*known-xs*] [, [*new-xs*] [, *const-flag*]])

Description: Computes values along a linear trend. Fits a straight line (using the method of least squares) to the arrays *known-ys* and *known-xs*. The y-values along that line for the array of *new-xs* specified.

Arguments:

Name	Type	Description
known-ys	array	The set of y-values already known in the relationship *y=mx+b*. If that array is in a single column, each column of *known-xs* is interpreted as a separate variable. If that array is in a single row, each row of *known-xs* is interpreted as a separate variable.
known-xs	array	An optional set of x-values that might already be known in the relationship *y=mx+b*. The array *known-xs* can include one or more sets of variables. If only one variable is used, *known-ys* and *known-xs* can be ranges of any shape, as long as they have equal dimensions. If more than one variable is used, *known-ys* shall be a vector. If *known-xs* is omitted, it is assumed to be the array {1,2,3,...} that is the same size as *known-ys*.
new-xs	array	New x-values for which TREND is to return corresponding y-values. *new-xs* shall include a column (or row) for each independent variable, just as *known-xs* does. So, if *known-ys* is in a single column, *known-xs* and *new-xs* shall have the same number of columns. If *known-ys* is in a single row, *known-xs* and *new-xs* shall have the same number of rows. If *new-xs* is omitted, it is assumed to be the same as *known-xs*. If both *known-xs* and *new-xs* are omitted, they are assumed to be the array {1,2,3,...} that is the same size as *known-ys*.
const-flag	logical	Specifies whether to force the constant b to equal 0. If TRUE or omitted, *b* is calculated normally. If FALSE, *b* is set equal to 0 and the m-values are adjusted so that *y=mx*.

Return Type and Value: array – The values along a linear trend, as an array of numbers.

3.17.7.326 TRIM

Syntax:

TRIM (*string*)

Description: Makes a string that is a copy of *string* with the leading and trailing space characters removed, and each sequence of embedded spaces reduced to a single space. The space character referred to here is character U+0020.

Arguments:

Name	Type	Description
string	text	Designates the string to be trimmed.

Return Type and Value: text – The trimmed copy of *string*.

[*Example*:

`TRIM("  abc  def  ")` results in `abc def`

end example]

3.17.7.327 TRIMMEAN

Syntax:

`TRIMMEAN ( array , percent )`

Description: Computes the mean of the interior of a data set by excluding a percentage of data points from the top and bottom tails of a data set. `TRIMMEAN` rounds the number of excluded data points down to the nearest multiple of 2. For symmetry, `TRIMMEAN` excludes a single value from the top and bottom of the data set.

Arguments:

Name	Type	Description
array	array, reference	The numeric values to trim and average.
percent	number	The fractional number of data points to exclude from the calculation. [*Example*: If *percent* = 0.2, 4 points are trimmed from a data set of 20 points (20x0.2): 2 from the top and 2 rom the bottom of the set. *end example*]

Return Type and Value: number – The mean of the interior of a data set.

However, if *percen* < 0 or *percen* > 1, `#NUM!` is returned.

[*Example*:

`TRIMMEAN({4,6,2,5,7,8,9},0.2)` results in `5.857142857`

end example]

3.17.7.328 TRUE

Syntax:

```
TRUE ( )
```

Description: Computes the value TRUE. (A call to function TRUE is equivalent to using the *logical-constant* TRUE.)

Arguments: None.

Return Type and Value: logical – The value TRUE.

[*Example*:

TRUE() results in TRUE

end example]

3.17.7.329 TRUNC

Syntax:

```
TRUNC ( x [ , number-digits ] )
```

Description: Truncates *x* to the number of fractional digits by *number-digits*.

Arguments:

Name	Type	Description
x	array, reference	The value to be rounded down.
number-digits	number	The number of fractional digits to which *x* is to be truncated. The default value for *number-digits* is 0.

Return Type and Value: number – The truncated value of *x*.

[*Example*:

TRUNC(PI()) results in 3
TRUNC(PI(),1) results in 3.1
TRUNC(PI(),3) results in 3.141
TRUNC(PI(),5) results in 3.14159

end example]

3.17.7.330 TTEST

Syntax:

TTEST (*array-1* , *array-2* , *distribution-tails* , *test-type*)

Description: Computes the probability associated with a Student's t-Test.

Arguments:

Name	Type	Description
array-1	array, reference	The first numerical data set.
array-1	array, reference	The first numerical data set.
distribution-tails	number	Specifies the number of distribution tails, truncated to an integer. If 1, TTEST uses the one-tailed distribution. If 2, TTEST uses the two-tailed distribution.
test-type	number	The truncated-to-integer kind of t-Test to perform, as follows:

Value	Test Performed
1	Paired
2	Two-sample equal variance (homoscedastic)
3	Two-sample unequal variance (heteroscedastic)

Return Type and Value: number – The probability associated with a Student's t-Test.

However, if

- *array-1* and *array-2* have a different number of data points, and *test-type* is 1, the return value is unspecified.
- *distribution-tails* is any value other than 1 or 2, #NUM! is returned.

[*Example*: Given the following data:

	A	B
1	Data 1	Data 2
2	3	6
3	4	19
4	5	3
5	8	2
6	9	14

	A	B
7	1	4
8	2	5
9	4	17
10	5	1

`TTEST(A2:A10,B2:B10,2,1)` results in `0.196016`

end example]

3.17.7.331 TYPE

Syntax:

TYPE (*value*)

Description: Computes the type of *value* or, if *value* is a reference to a single cell, the type of the value in that cell.

Arguments:

Name	Type	Description
value	any	The value whose type is to be determined. No conversion shall take place on an argument passed to this function.

Return Type and Value: number – An integer that indicates the type of *value* or, if *value* is a reference to a single cell, the type of the value in that cell, as follows:

Type of *value*	Value Returned
number	1
text	2
logical	4
error value	16
array of any kind	64

[*Example*:

`TYPE(10.5)` results in `1`
`TYPE(A10)` results in `1`, when A10 contains a number
`TYPE("ABC")` results in 2
`TYPE(A10)` results in 2, when A10 contains a string

TYPE(TRUE) results in 4
TYPE(A10) results in 4, when A10 contains a logical value

TYPE(5/0) results in 16
TYPE(A10) results in 16, when A10 contains any error value
TYPE({1,2,3}) results in 64
TYPE({TRUE,2.5,#N/A}) results in 64
TYPE(IF(10>5,"Yes",20)) results in 2
TYPE(IF(10<5,"Yes",20)) results in 1

end example]

3.17.7.332 UPPER

Syntax:

UPPER (*string*)

Description: Makes an uppercase version of *string*.

Arguments:

Name	Type	Description
string	text	Designates the string to be converted.

Return Type and Value: text – The uppercase version of *string*.

[*Example*:

UPPER("AbCd123#$%^") results in ABCD123#$%^
UPPER(A10) results in 234FRTQWC$#%, when A10 contains 234FRTqwc$#%

end example]

3.17.7.333 USDOLLAR

Syntax:

USDOLLAR (*number* [, *num-decimal*])

Description: Produces a string containing *number* rounded to *num-decimal* decimal places. The thousands separator is the comma, the radix point is the period, and the currency symbol is "$". The format used is $#,##0.00;($#,##0.00).

Arguments:

Name	Type	Description

Name	Type	Description
number	number	The number that is to be formatted.
num-decimal	number	Designate the number of decimal places to be used in the resulting string; it is truncated to an integer. If *num-decimal* is negative, *number* is rounded to the left of the decimal point. If omitted, a value of 2 shall be assumed.

Return Type and Value: text – The string containing *number* rounded to *num-decimal* decimal places, and have a currency symbol and thousands separators.

[*Example*:

`USDOLLAR(1234.567)` results in `$1,234.57`
`USDOLLAR(1234.567,-2)` results in `$1,200`
`USDOLLAR(-1234.567,4)` results in `($1,234.5670)`

end example]

3.17.7.334 VALUE

Syntax:

```
VALUE ( string )
```

Description: Converts *string* to a number.

Arguments:

Name	Type	Description
string	text	Designates a string that contains a number formatted using any number, currency, date, or time format. (See §3.8.31 for the set of formats.) Date and time strings are converted to their equivalent serial value.

Return Type and Value: number – The number represented by *string*.

[*Example*:

`VALUE("123.456")` results in `123.456`
`VALUE("$1,000")` results in `1000`
`VALUE("23-Mar-2002")` results in the corresponding serial value
`VALUE("16:48:00")-VALUE("12:17:12")` results in `0.188056`

end example]

3.17.7.335 VAR

Syntax:

VAR (*argument-list*)

Description: Makes an estimate of the variance based on a sample. [*Note*: VAR assumes that its arguments are a sample of the population. If the data represents the entire population, VARP should be used instead. If logical values and text representations of numbers in a reference are to be included as part of the calculation, use VARA instead. *end note*]

Mathematical Formula:

$$\frac{\sum(x-\bar{x})^2}{(n-1)}$$

where x is the sample mean AVERAGE(*argument-1*, *argument-1*,…, *argument-n*) and n is the sample size.

Arguments:

Name	Type	Description
argument-list	logical, number, name, text, array, reference	The *argument*s in *argument-list* designate the numbers that are samples of the population. Logical values, and text representations of numbers that are entered directly into the list of arguments are included. If an argument is an array or reference, only numbers in that array or reference are included. Empty cells, logical values, text, or error values in the array or reference are ignored.

Return Type and Value: number – An estimate of the variance based on a sample.

[*Example*:

VAR(1202,1220,1323,1254,1302) results in 2683.2

end example]

3.17.7.336 VARA

Syntax:

VARA (*argument-list*)

Description: Makes an estimate of the variance based on a sample. [*Note*: VARA assumes that its arguments are a sample of the population. If the data represents the entire population, VARPA should be used instead. If logical values and text representations of numbers in a reference are to be excluded as part of the calculation, use VAR instead. *end note*]

Mathematical Formula:

$$\frac{\sum(x-\bar{x})^2}{(n-1)}$$

where x is the sample mean AVERAGE(*argument-1*, *argument-1*,…, *argument-n*) and n is the sample size.

Arguments:

Name	Type	Description
argument-list	logical, number, name, text, array, reference	The *argument*s in *argument-list* designate the numbers that are samples of the population. Logical values and text representations of numbers that are entered directly into the list of arguments are included. Arguments that contain TRUE evaluate as 1; arguments that contain text or FALSE evaluate as zero. If an argument is an array or reference, only values in that array or reference are used. Empty cells and text values in the array or reference are ignored.

Return Type and Value: number – An estimate of the variance based on a sample.

[*Example*:

VARA(1202,1220,1323,1254,1302) results in 2683.2

end example]

3.17.7.337 VARP

Syntax:

VARP (*argument-list*)

Description: Computes the variance of an entire population. [*Note*: VARP assumes that its arguments are the total population. If the data represents a population sample only, VAR should be used instead. If logical values and text representations of numbers in a reference are to be included as part of the calculation, use VARPA instead. *end note*]

Mathematical Formula:

$$\frac{\sum(x-\bar{x})^2}{n}$$

where x is the sample mean AVERAGE(*argument-1*, *argument-1*,…, *argument-n*) and n is the sample size.

Arguments:

Name	Type	Description
argument-list	logical, number, name, text, array, reference	The *argument*s in *argument-list* designate the numbers that are the members of the population. Logical values, and text representations of numbers that are entered directly into the list of arguments are included. If an argument is an array or reference, only numbers in that array or reference are included. Empty cells, logical values, text, or error values in the array or reference are ignored.

Return Type and Value: number – The variance of an entire population.

[*Example*:

VARP(1202,1220,1323,1254,1302) results in 2146.56

end example]

3.17.7.338 VARPA

Syntax:

VARPA (*argument-list*)

Description: Makes the variance of an entire population. [*Note*: VARPA assumes that its arguments are the total population. If the data represents a population sample only, VARA should be used instead. If logical values and text representations of numbers in a reference are to be excluded as part of the calculation, use VARP instead. *end note*]

Mathematical Formula:

$$\frac{\sum (x-\bar{x})^2}{n}$$

where x is the sample mean AVERAGE(*argument-1*, *argument-1*,…, *argument-n*) and n is the sample size.

Arguments:

Name	Type	Description
argument-list	logical, number, name, text, array, reference	The *argument*s in *argument-list* designate the numbers that are the members of the population.

Return Type and Value: number – The variance of an entire population.

Arguments can be numbers; names, arrays, or references that contain numbers; text representations of numbers; or logical values, in a reference. Text representations of numbers that are entered directly into the list of arguments are included. Arguments that contain TRUE evaluate as 1; arguments that contain text or FALSE evaluate as zero. If an argument is an array or reference, only values in that array or reference are used. Empty cells and text values in the array or reference are ignored.

[*Example*:

`VARPA(1202,1220,1323,1254,1302)` results in `2146.56`

end example]

3.17.7.339 VDB

VDB (*cost* , *salvage* , *life* , *start-period* , *end-period* [, [[*factor*] [, [*no-switch-flag*]]]])

Description: Computes the depreciation of an asset for the period specified, including partial periods, using the double-declining balance or some other specified method.

Arguments:

Name	Type	Description
cost	number	The number *cost* is the initial cost of the asset.
salvage	number	The value at the end of the depreciation. (This is sometimes called the salvage value of the asset.) This value can be 0.
life	number	The number of periods over which the asset is being depreciated. (This is sometimes called the useful life of the asset.)
start-period	number	The starting period for which the depreciation is to be calculated. (*start-period* shall use the same units as *life*.)
end-period	number	The ending period for which the depreciation is to be calculated. (*end-period* shall use the same units as *life*.)
factor	number	The rate at which the balance declines. If omitted, it is assumed to be 2 (the double-declining balance method).
no-switch-flag	logical	Specifies whether to switch to straight-line depreciation when depreciation is greater than the declining balance calculation. If TRUE, straight-line depreciation is not used even when the depreciation is greater than the declining balance calculation. If FALSE or omitted, the straight-line depreciation is used when depreciation is greater than the declining balance calculation.

Return Type and Value: number – The depreciation of an asset for the period specified.

However, if any numerical argument value is non-positive, `#NUM!` is returned.

[*Example*:

`VDB(2400,300,10*365,0,1)` results in `1.32`
`VDB(2400,300,10*12,0,1)` results in `40.00`
`VDB(2400,300,10*12,6,18)` results in `396.31`

end example]

3.17.7.340 VLOOKUP

Syntax:

VLOOKUP (*lookup-value* , *table-array* , *col-index-num* [, [*range-lookup-flag*]])

Description: Performs a vertical search for a value in the left-most column of a table or an array, noting the row in which the matching value is found. From that row, the value from a given column is returned.

Arguments:

Name	Type	Description
lookup-value	value of any type or a reference to a value of any type.	The value to be located in the left-most column of the table. If *range-lookup* is FALSE and *lookup-value* is a string, the wildcard characters, question mark (?) and asterisk (*), can be included in *lookup-value*. A question mark matches any single character; an asterisk matches any sequence of characters. To find a question mark or asterisk, type a tilde (~) before the character.
table-array	array, reference, name	Designates the table of information to be searched. The values in the left-most column of *table-array* can be text, numbers, or logical values. The values in the left-most column of *table-array* shall be placed in "ascending order", as follows: ..., -2, -1, 0, 1, 2, ..., A–Z, FALSE, TRUE. Uppercase and lowercase text is treated as equivalent.
col-index-num	number	The column number in *table-array* from which the matching value is to be returned. (A *col-index-num* of 1 returns the left-most column value in *table-array*, a *col-index-num* of 2 returns the next column in *table-array*, and so on.)
range-lookup-flag	logical	Specifies whether HLOOKUP is to find an exact or approximate match. If TRUE or omitted, an approximate match is returned. That is, if an exact match is not found,

Name	Type	Description
		the next largest value that is less than *lookup-value* is returned. If FALSE, an exact match is performed, in which case, the values in the left-most column of *table-array* need not be sorted. If there are two or more values in the left-most column of *table-array* that match *lookup-value*, the top-most value found is used.

Return Type and Value: any – The value from a given row number, where the column is determined by a search of the top row looking for a match with a given value.

However, if

- An exact match is performed, but no match is found, #N/A is returned.
- *col-index-num* is less than 1, #VALUE! is returned.
- *col-index-num* is greater than the number of columns in *table-array*, #REF! is returned.
- *lookup-value* is smaller than the smallest value in the left-most column of *table-array*, #N/A is returned.

[*Example*: Given the following data:

	A	B	C
1	Density	Bearings	Bolts
2	0.457	3.55	500
3	0.525	3.25	400
4	0.616	2.93	300
5	0.675	2.75	250
6	0.746	2.57	200
7	0.835	2.38	150
8	0.946	2.17	100
9	1.09	1.95	50
10	1.29	1.71	0

VLOOKUP(1,A2:C10,2) results in 2.17
VLOOKUP(1,A2:C10,3,TRUE) results in 100.00
VLOOKUP(2,A2:C10,2,TRUE) results in 1.71

end example]

3.17.7.341 WEEKDAY

Syntax:

WEEKDAY (*serial-value* [, *weekday-start-flag*])

Description: Computes the weekday number for the date having the given *serial-value*, taking into account the current date base value and *weekday-start-flag*, if present. See §3.17.4.1 for special handling of certain days in 1900.

Arguments:

Name	Type	Description
serial-value	number	The date whose weekday number is to be computed. The value of *serial-value* is truncated to an integer.
weekday-start-flag	number	When truncated to integer, indicates the weekday numbering convention to be used, as follows: **Value** / **Meaning** 1 or omitted — 1 (Sunday) through 7 (Saturday) 2 — 1 (Monday) through 7 (Sunday) 3 — 0 (Monday) through 6 (Sunday)

Return Type and Value: number – The weekday number for the date having the given serial value.

However, if

- *serial-value* is out of range for the current date base value, #NUM! is returned.
- *weekday-start-flag* is out of the range specified in the table above, #NUM! is returned.

[*Example*:

WEEKDAY(DATE(2006,2,1)) results in 4 (Wednesday)
WEEKDAY(DATE(2006,2,1),1) results in 4 (Wednesday)
WEEKDAY(DATE(2006,2,1),2) results in 3 (Wednesday)
WEEKDAY(DATE(2006,2,1),3) results in 2 (Wednesday)

end example]

3.17.7.342 WEEKNUM

Syntax:

WEEKNUM (*serial-value* [, *weekday-start-flag*])

Description: Computes the week number of the date corresponding to *serial-value*. The week containing January 1 is the first week of the year, and is numbered week 1.

Arguments:

Name	Type	Description
serial-value	number	The date whose week number is to be computed. The value of *serial-value* is truncated to an integer.
weekday-start-flag	number	When truncated to integer, indicates the weekday on which the week begins, as follows: *weekday-start-flag* / Meaning 1 or omitted / Week begins on Sunday. 2 / Week begins on Monday.

Return Type and Value: number – The week number of the date corresponding to *serial-value*.

However, if

- *serial-value* is out of range for the current date base value, `#NUM!` is returned.
- *weekday-start-flag* is out of the range specified in the table above, `#NUM!` is returned.

[*Example*:

`WEEKNUM(DATE(2006,1,1)` results in `1`
`WEEKNUM(DATE(2006,1,1),1)` results in `1`
`WEEKNUM(DATE(2006,2,1),1)` results in `5`
`WEEKNUM(DATE(2006,2,1),2)` results in `6`

end example]

3.17.7.343 WEIBULL

Syntax:

`WEIBULL ( x , alpha , beta , cumulative-flag )`

Description: Computes the Weibull distribution.

Mathematical Formula:

The equation for the Weibull cumulative distribution function is:

$$F(x;\alpha,\beta) = 1 - e^{-(x/\beta)^{\alpha}}$$

The equation for the Weibull probability density function is:

$$f(x;\alpha,\beta) = \frac{\alpha}{\beta^{\alpha}} x^{\alpha-1} e^{-(x/\beta)^{\alpha}}$$

When *alpha* = 1, `WEIBULL` returns the exponential distribution with:

$$\lambda = \frac{1}{\beta}$$

Arguments:

Name	Type	Description
x	number	The value at which the distribution is to be evaluated.
alpha	number	A parameter of the distribution.
beta	number	A parameter of the distribution.
cumulative-flag	logical	Determines the form of the function. If `TRUE`, `GAMMADIST` returns the cumulative distribution function; if `FALSE`, it returns the probability density function.

Return Type and Value: number – The Weibull distribution.

However, if

- *x* < 0, `#NUM!` is returned.
- *alpha* ≤ 0, `#NUM!` is returned.
- *beta* ≤ 0, `#NUM!` is returned.

[*Example*:

`WEIBULL(105,20,100,TRUE)` results in `0.92958139`
`WEIBULL(105,20,100,FALSE)` results in `0.035588864`

end example]

3.17.7.344 WORKDAY

Syntax:

`WORKDAY ( start-date , day-offset [ , holidays ] )`

Description: Computes the serial value of the date that is *day-offset* working days offset from *start-date*. Weekend days and any holidays specified by *holidays* are not considered as working days.

Arguments:

Name	Type	Description
start-date	number	The start date, truncated to integer.

Name	Type	Description
day-offset	number	The number of working days before or after *start-date*. A positive value yields a future date; a negative value yields a past date; a zero value yields the date *start-date*. *day-offset* is truncated to an integer.
holidays	reference, array	An optional set of one or more dates that are to be excluded from the working day calendar. *holidays* shall be a range of cells that contain the dates, or an array constant of the serial values that represent those dates. The ordering of dates or serial values in *holidays* can be arbitrary.

Return Type and Value: number – The serial value of the date that is *day-offset* working days offset from *start-date*, excluding the specified holidays.

However, if

- *start-date* is out of range for the current date base value, `#NUM!` is returned.
- Any date in *holidays* is out of range for the current date base value, `#NUM!` is returned.
- *start-date* plus *day-offset* yields an invalid date, `#NUM!` is returned.

[*Example*:

`WORKDAY(DATE(2006,1,1),0)` results in a serial value corresponding to 1-Jan-2006
`WORKDAY(DATE(2006,1,1),10)` results in a serial value corresponding to 13-Jan-2006
`WORKDAY(DATE(2006,1,1),-10)` results in a serial value corresponding to 19-Dec-2005
`WORKDAY(DATE(2006,1,1),20,{"2006/1/2","2006/1/16"})` results in a serial value corresponding to 31-Jan-2006

end example]

3.17.7.345 XIRR

`XIRR ( values , dates [ , [ guess ] ] )`

Description: Computes the internal rate of return for a schedule of cash flows that is not necessarily periodic. `XIRR` uses an iterative calculation technique that cycles through the calculation until the result is accurate within 0.000001 percent.

Mathematical Formula:

Using a changing rate (starting with *guess*), `XIRR` cycles through the calculation until the result is accurate within 0.000001 percent. The rate is changed until:

$$0 = \sum_{i=1}^{M} \frac{P_i}{(1+rate)^{\frac{(d_i - d_1)}{365}}}$$

where:

di = the i^{th}, or last, payment date.
d1 = the 0^{th} payment date.
Pi = the i^{th}, or last, payment.

Arguments:

Name	Type	Description
values	array, reference	A series of cash flows that corresponds to a schedule of payment dates specified in *dates*. The first payment is optional and corresponds to a cost or payment that occurs at the beginning of the investment. If the first value is a cost or payment, it shall have a negative value. All succeeding payments are discounted based on a 365-day year. The series of values shall contain at least one positive and one negative value.
dates	reference	A schedule of payment dates that corresponds to the cash flow payments in *values*. The first payment date indicates the beginning of the schedule of payments. All other dates shall be later than this date, but they can occur in any order. Time information in the date arguments is ignored.
guess	number	An estimate of the result of XIRR. If omitted, it is assumed to be 0.1 (i.e., 10 percent).

Return Type and Value: number – The internal rate of return for a schedule of cash flows that is not necessarily periodic.

However, if

- Any date in *dates* is out of range for the current date base value, #NUM! is returned.
- Any date in *dates* precedes the starting date, #NUM! is returned.
- *values* and *dates* contain different numbers of values, #NUM! is returned.
- The calculation has not converged after an implementation-defined number of tries, #NUM! is returned.

[*Example*: When the cells F2397:J2397contain the dates January 1, 2008; March 1,2008; October 30, 2008; February 15, 2009, and April 1, 2009:

XIRR({-10000,2750,4250,3250,2750},F2397:J2397,0.1) results in 37.34%

end example]

3.17.7.346 XNPV

XNPV (*rate* , *values* , *dates*)

Description: Computes the net present value for a schedule of cash flows that is not necessarily periodic.

Mathematical Formula:

$$XNPV = \sum_{i=1}^{N} \frac{P_i}{(1+rate)^{\frac{(d_i - d_1)}{365}}}$$

where:

di = the i^{th}, or last, payment date.
d1 = the 0^{th} payment date.
Pi = the i^{th}, or last, payment.

Arguments:

Name	Type	Description
rate	number	The discount rate to apply to the cash flows.
values	array, reference	A series of cash flows that corresponds to a schedule of payment dates specified in *dates*. The first payment is optional and corresponds to a cost or payment that occurs at the beginning of the investment. If the first value is a cost or payment, it shall have a negative value. All succeeding payments are discounted based on a 365-day year. The series of values shall contain at least one positive and one negative value.
dates	reference	A schedule of payment dates that corresponds to the cash flow payments in *values*. The first payment date indicates the beginning of the schedule of payments. All other dates shall be later than this date, but they can occur in any order. Time information in the date arguments is ignored.

Return Type and Value: number – The net present value for a schedule of cash flows that is not necessarily periodic.

However, if

- Any date in *dates* is out of range for the current date base value, `#NUM!` is returned.
- Any date in *dates* precedes the starting date, `#NUM!` is returned.

- *values* and *dates* contain different numbers of values, #NUM! is returned.

[*Example*: When the cells F2397:J2397 contain the dates January 1, 2008; March 1,2008; October 30, 2008; February 15, 2009, and April 1, 2009:

XNPV(0.09,{-10000,2750,4250,3250,2750},F2397:J2397) results in 2086.65

end example]

3.17.7.347 YEAR

Syntax:

YEAR (*date-value*)

Description: Computes the numeric Gregorian year for the date and/or time having the given *date-value*, taking into account the current date base value. That date and/or time shall be expressed either as a serial value, in which case, its fractional part is ignored, or as a *string-constant* having any valid date and/or time format, in which case, any time information shall be ignored.

Arguments:

Name	Type	Description
date-value	number, text	The date and/or time whose year is to be computed. That date and/or time shall be expressed either as a serial value, in which case, its fractional part is ignored, or as a *string-constant* having any valid date and/or time format, in which case, any time information shall be ignored.

Return Type and Value: number – The Gregorian year for the date and/or time having the given *date-value*. For the 1900 date base system, the returned value shall be in the range 1900–9999. For the 1904 date base system, the returned value shall be in the range 1904–9999.

However, if *date-value* is out of range for the current date base value, #NUM! is returned.

[*Example*:

YEAR(DATE(2006,1,2)) results in 2006
YEAR(DATE(2006,0,2)) results in 2005
YEAR("2006/1/2 10:45 AM") results in 2006
YEAR(30000) results in 1982 for the 1900 date base system, or 1986 for the 1904 date base system

end example]

3.17.7.348 YEARFRAC

Syntax:

YEARFRAC (*start-date* , *end-date* [, *basis*])

Description: Computes the fractional number of years represented by the number of whole days between two dates, *start-date* and *end-date*., according to *basis*.

Arguments:

Name	Type	Description
start-date	number	The period's starting date. *start-date* can be earlier than, the same as, or later than *end-date*.
end-date	number	The period's ending date.
day-count-basis	number	The security's issue date.
basis	number	The truncated integer type of day count basis to use, as follows: **Value** / **Day Count Basis** 0 or omitted / US (NASD) 30/360 1 / Actual/actual 2 / Actual/360 3 / Actual/365 4 / European 30/360

All arguments are truncated to integers.

Return Type and Value: number – The fractional number of years represented by the number of whole days between two dates, *start-date* and *end-date*., according to *basis*.

However, if the value of *basis* is out of range, #NUM! is returned.

[*Example*:

YEARFRAC(DATE(2006,1,1),DATE(2006,3,26)) results in 0.236111111
YEARFRAC(DATE(2006,3,26),DATE(2006,1,1)) results in 0.236111111
YEARFRAC(DATE(2006,1,1),DATE(2006,7,1)) results in 0.5
YEARFRAC(DATE(2006,1,1),DATE(2007,9,1)) results in 1.666666667
YEARFRAC(DATE(2006,1,1),DATE(2006,7,1),0) results in 0.5
YEARFRAC(DATE(2006,1,1),DATE(2006,7,1),1) results in 0.495890411
YEARFRAC(DATE(2006,1,1),DATE(2006,7,1),2) results in 0.502777778

YEARFRAC(DATE(2006,1,1),DATE(2006,7,1),3) results in 0.495890411
YEARFRAC(DATE(2006,1,1),DATE(2006,7,1),4) results in 0.5

end example]

3.17.7.349 YIELD

Syntax:

YIELD (*settlement* , *maturity* , *rate* , *pr* , *redemption* , *frequency* [, [*basis*]])

Description: Computes the yield on a security that pays periodic interest.

Mathematical Formula:

If there is one coupon period or less until redemption, YIELD is calculated as follows:

$$YIELD = \frac{(\frac{redemption}{100} + \frac{rate}{frequency}) - (\frac{par}{100} + (\frac{A}{E} \times \frac{rate}{frequency}))}{\frac{par}{100} + (\frac{A}{E} \times \frac{rate}{frequency})} \times \frac{frequency \times E}{DSR}$$

where:

A = number of days from the beginning of the coupon period to the settlement date (accrued days).
DSR = number of days from the settlement date to the redemption date.
E = number of days in the coupon period.

If there is more than one coupon period until redemption, YIELD is calculated through some number of iterations. The resolution uses the Newton method, based on the formula used for the function PRICE. The yield is changed until the estimated price given the yield is close to price.

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
rate	number	The security's interest rate.
pr	number	The security's price.
redemption	number	The security's redemption value per $100 face value.
frequency	number	the number of coupon payments per year. (For annual payments, *frequency* is 1; for semiannual payments, *frequency* is 2; for quarterly payments, *frequency* is 4.) *frequency* is truncated to an integer.
basis	number	The truncated integer type of day count basis to use, as follows:

Name	Type	Description
		<table><tr><th>Value</th><th>Day Count Basis</th></tr><tr><td>0 or omitted</td><td>US (NASD) 30/360</td></tr><tr><td>1</td><td>Actual/actual</td></tr><tr><td>2</td><td>Actual/360</td></tr><tr><td></td><td>Actual/365</td></tr><tr><td>4</td><td>European 30/360</td></tr></table>

Time information in the date arguments is ignored.

Return Type and Value: number – The yield on a security that pays periodic interest.

However, if

- *settlement* or *maturity* is out of range for the current date base value, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *rate* < 0, `#NUM!` is returned.
- *pr* or *redemption* ≤ 0, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`YIELD(DATE(2008,2,15),DATE(2016,11,15),0.0575,95.04287,100,2,0)` results in `6.5000%`

end example]

3.17.7.350 YIELDDISC

Syntax:

`YIELDDISC ( `*settlement*` , `*maturity*` , `*pr*` , `*redemption*` [ , [ `*basis*` ] ] )`

Description: Computes the annual yield for a discounted security.

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
pr	number	The security's price.

Name	Type	Description
redemption	number	The security's redemption value per $100 face value.
basis	number	The truncated integer type of day count basis to use, as follows: 0 or omitted: US (NASD) 30/360 1: Actual/actual 2: Actual/360 3: Actual/365 4: European 30/360

Time information in the date arguments is ignored.

Return Type and Value: number – The annual yield for a discounted security.

However, if

- *settlement* or *maturity* is out of range for the current date base value, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *pr* or *redemption* ≤ 0, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`YIELDDISC(DATE(2008,2,16),DATE(2008,3,1),99.795,100,2)` results in `5.2823%`

end example]

3.17.7.351 YIELDMAT

Syntax:

`YIELDMAT ( settlement , maturity , issue , rate , pr [ , [ basis ] ] )`

Description: Computes the annual yield of a security that pays interest at maturity.

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
issue	number	The security's issue date.

Name	Type	Description
rate	number	The security's interest rate.
pr	number	The security's price.
basis	number	The truncated integer type of day count basis to use, as follows: 0 or omitted: US (NASD) 30/360 1: Actual/actual 2: Actual/360 3: Actual/365 4: European 30/360

Time information in the date arguments is ignored.

Return Type and Value: number – The annual yield of a security that pays interest at maturity.

However, if

- *settlement*, *maturity*, or *issue* is out of range for the current date base value, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *rate* or *pr* ≤ 0, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

```
YIELDMAT(DATE(2008,3,15),DATE(2008,11,3),DATE(2007,11,8),0.0625,
  100.0123,0) results in 6.0954%
```

end example]

3.17.7.352 ZTEST

Syntax:

```
ZTEST ( array , test-value [ , sigma ] )
```

Description: Computes the one-tailed probability-value of a z-test. For a given hypothesized population mean, *test-value*, ZTEST returns the probability that the sample mean would be greater than the average of observations in the data set *array*; that is, the observed sample mean.

Mathematical Formula:

When *sigma* is present:

$$ZTEST(array, \mu_0) = 1 - NORMSDIST((\bar{x} - \mu_0)/(sigma / \sqrt{n}))$$

When *sigma* is omitted:

$$ZTEST(array, \mu_0) = 1 - NORMSDIST((\bar{x} - \mu_0)/(s / \sqrt{n}))$$

where x is the sample mean AVERAGE(*array*); s is the sample standard deviation STDEV(*array*); and n is the number of observations in the sample COUNT(*array*).

Arguments:

Name	Type	Description
array	array	The set of numerical data against which to test *test-value*.
test-value	number	The number to test.
sigma	number	The number is the population (known) standard deviation. If omitted, the sample standard deviation is used.

Return Type and Value: number – The one-tailed probability-value of a z-test.

However, if *array* is empty, the return value is unspecified.

[*Example*:

`ZTEST({3,6,7,8,6,5,4,2,1,9},4)` results in `0.090574197`
`ZTEST({3,6,7,8,6,5,4,2,1,9},6)` results in `0.863043389`

end example]

3.18 Simple Types

This is the complete list of simple types in the http://schemas.openxmlformats.org/spreadsheetml/2006/main namespace.

3.18.1 ST_Axis (PivotTable Axis)

This simple type defines the axes for a PivotTable selection.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description

Enumeration Value	Description
axisCol (Column Axis)	Column axis
axisPage (Include Count Filter)	Page axis
axisRow (Row Axis)	Row axis
axisValues (Values Axis)	Values axis

Referenced By
pivotArea@axis (§3.3.1.66); pivotField@axis (§3.10.1.69); pivotSelection@axis (§3.3.1.67)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Axis">
   <restriction base="xsd:string">
      <enumeration value="axisRow"/>
      <enumeration value="axisCol"/>
      <enumeration value="axisPage"/>
      <enumeration value="axisValues"/>
   </restriction>
</simpleType>
```

3.18.2 ST_BorderId (Border Id)

Zero-based index of the border record used by this cell format.

This simple type's contents are a restriction of the XML Schema unsignedInt datatype.

Referenced By
xf@borderId (§3.8.45)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_BorderId">
   <restriction base="xsd:unsignedInt"/>
</simpleType>
```

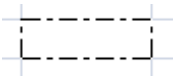
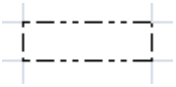
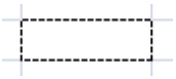
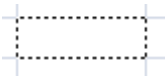
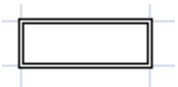
3.18.3 ST_BorderStyle (Border Line Styles)

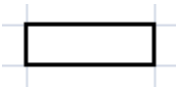
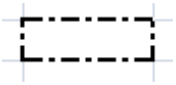
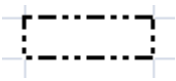
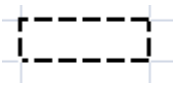
The line style of a border in a cell.

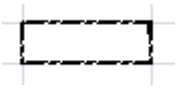
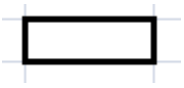
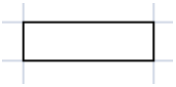
This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
dashDot (Dash Dot)	The line style of a border is dash-dot. [*Example*:

Enumeration Value	Description
	 *end example*]
dashDotDot (Dash Dot Dot)	The line style of a border is dash-dot-dot. [*Example*:  *end example*]
dashed (Dashed)	The line style of a border is dashed. [*Example*:  *end example*]
dotted (Dotted)	The line style of a border is dotted. [*Example*:  *end example*]
double (Double Line)	The line style of a border is double line. [*Example*:  *end example*]
hair (Hairline Border)	The line style of a border is hairline. [*Example*: 

Enumeration Value	Description
	end example]
medium (Medium Border)	The line style of a border is medium. [*Example*:  *end example*]
mediumDashDot (Medium Dash Dot)	The line style of a border is medium dash-dot. [*Example*:  *end example*]
mediumDashDotDot (Medium Dash Dot Dot)	The line style of a border is medium dash-dot-dot. [*Example*:  *end example*]
mediumDashed (Medium Dashed)	The line style of a border is medium dashed. [*Example*:  *end example*]
none (None)	The line style of a border is none (no border visible). [*Example*: *end example*]
slantDashDot (Slant Dash Dot)	The line style of a border is slant-dash-dot.

Enumeration Value	Description
	[*Example*:  *end example*]
thick (Thick Line Border)	The line style of a border is 'thick'. [*Example*:  *end example*]
thin (Thin Border)	The line style of a border is thin. [*Example*:  *end example*]

Referenced By
bottom@style (§3.8.6); diagonal@style (§3.8.13); horizontal@style (§3.8.24); left@style (§3.8.27); right@style (§3.8.35); top@style (§3.8.43); vertical@style (§3.8.44)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_BorderStyle">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="thin"/>
      <enumeration value="medium"/>
      <enumeration value="dashed"/>
      <enumeration value="dotted"/>
      <enumeration value="thick"/>
      <enumeration value="double"/>
      <enumeration value="hair"/>
      <enumeration value="mediumDashed"/>
      <enumeration value="dashDot"/>
      <enumeration value="mediumDashDot"/>
      <enumeration value="dashDotDot"/>
      <enumeration value="mediumDashDotDot"/>
      <enumeration value="slantDashDot"/>
   </restriction>
</simpleType>
```

3.18.4 ST_CalcMode (Calculation Mode)

This simple type defines the supported modes for performing calculations on workbook data.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
auto (Automatic)	Indicates that calculations in the workbook will be performed automatically when cell values change. The application recalculates those cells that are dependent on other cells that contain changed values. This type of calculation helps to avoid unnecessary calculations.
autoNoTable (Automatic Calculation (No Tables))	Indicates tables be excluded during automatic calculation.
manual (Manual Calculation Mode)	Indicates that calculations in the workbook be triggered manually by the user. For example, the application might expose a command in the user interface.

Referenced By
calcPr@calcMode (§3.2.2)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_CalcMode">
   <restriction base="xsd:string">
      <enumeration value="manual"/>
      <enumeration value="auto"/>
      <enumeration value="autoNoTable"/>
   </restriction>
</simpleType>
```

3.18.5 ST_CalendarType (Calendar Type)

Calendar type enumeration for date grouped items of a filter.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
gregorian (Gregorian)	Gregorian (localized) calendar
gregorianArabic (Gregorian Arabic Calendar)	Gregorian Arabic calendar
gregorianMeFrench (Gregorian Middle East French Calendar)	Gregorian Middle East French calendar
gregorianUs (Gregorian (U.S.) Calendar)	Gregorian (U.S.) calendar
gregorianXlitEnglish (Gregorian Transliterated English Calendar)	Gregorian Transliterated English calendar
gregorianXlitFrench (Gregorian Transliterated French Calendar)	Gregorian Transliterated French calendar
hebrew (Hebrew (Lunar) Calendar)	Hebrew (Lunar) calendar
hijri (Hijri (Arabic Lunar) Calendar)	Hijri (Arabic Lunar) calendar
japan (Japanese Emperor Era Calendar)	Japanese Emperor Era calendar
korea (Korean Tangun Era Calendar)	Korean Tangun Era calendar
none (No Calendar Type)	No calendar type specified
taiwan (Taiwan Era Calendar)	Taiwan Era calendar
thai (Thai Calendar)	Thai calendar

Referenced By
filters@calendarType (§3.3.2.8)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_CalendarType">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="gregorian"/>
      <enumeration value="gregorianUs"/>
      <enumeration value="japan"/>
      <enumeration value="taiwan"/>
      <enumeration value="korea"/>
      <enumeration value="hijri"/>
      <enumeration value="thai"/>
      <enumeration value="hebrew"/>
      <enumeration value="gregorianMeFrench"/>
      <enumeration value="gregorianArabic"/>
      <enumeration value="gregorianXlitEnglish"/>
      <enumeration value="gregorianXlitFrench"/>
   </restriction>
</simpleType>
```

3.18.6 ST_CellComments (Cell Comments)

These enumerations specify how cell comments shall be displayed for paper printing purposes.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
asDisplayed (Print Comments As Displayed)	Print cell comments as displayed.
atEnd (Print At End)	Print cell comments at end of document.
none (None)	Do not print cell comments.

Referenced By
pageSetup@cellComments (§3.3.1.61)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_CellComments">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="asDisplayed"/>
      <enumeration value="atEnd"/>
   </restriction>
</simpleType>
```

3.18.7 ST_CellFormulaType (Formula Type)

Indicates the type of formula in the cell.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
array (Array Entered)	Formula is an array entered formula.
dataTable (Table Formula)	Formula is a data table formula.
normal (Normal)	Formula is a regular cell formula.
shared (Shared Formula)	Formula is part of a shared formula.

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_CellFormulaType">
    <restriction base="xsd:string">
        <enumeration value="normal"/>
        <enumeration value="array"/>
        <enumeration value="dataTable"/>
        <enumeration value="shared"/>
    </restriction>
</simpleType>
```

3.18.8 ST_CellRef (Cell Reference)

Represents a single cell reference in a SpreadsheetML document.

This simple type's contents are a restriction of the XML Schema string datatype.

Referenced By
c@r (§3.6.1); c@r (§3.3.1.3); cell@r (§3.14.1); cellSmartTags@r (§3.3.1.6); cellWatch@r (§3.3.1.7); customSheetView@topLeftCell (§3.3.1.23); inputCells@r (§3.3.1.49); nc@r (§3.11.1.3); oc@r (§3.11.1.5); pane@topLeftCell (§3.3.1.64); rcmt@cell (§3.11.1.11); selection@activeCell (§3.3.1.75); sheetView@topLeftCell (§3.3.1.83); singleXmlCell@r (§3.5.2.1); tr@r (§3.15.4); undo@r (§3.11.1.25)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_CellRef">
    <restriction base="xsd:string"/>
</simpleType>
```

3.18.9 ST_CellSpan (Cell Span Type)

A single cell span item.

This simple type's contents are a restriction of the XML Schema string datatype.

Referenced By
ST_CellSpan (§3.18.9)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_CellSpan">
   <restriction base="xsd:string"/>
</simpleType>
```

3.18.10 ST_CellSpans (Cell Spans)

List of the cell spans of the item.

This simple type allows a list of items of the ST_CellSpan simple type (§3.18.9).

Referenced By
row@spans (§3.3.1.71)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_CellSpans">
   <list itemType="ST_CellSpan"/>
</simpleType>
```

3.18.11 ST_CellStyleXfId (Cell Style Format Id)

Used by xf records and cellStyle records to reference xf records defined in the cellStyleXfs collection.

This simple type's contents are a restriction of the XML Schema unsignedInt datatype.

Referenced By
cellStyle@xfId (§3.8.7); xf@xfId (§3.8.45)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_CellStyleXfId">
   <restriction base="xsd:unsignedInt"/>
</simpleType>
```

3.18.12 ST_CellType (Cell Type)

Indicates the cell's data type.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
b (Boolean)	Cell containing a boolean.
e (Error)	Cell containing an error.
inlineStr (Inline String)	Cell containing an (inline) rich string, i.e., one not in the shared string table. If this cell type is used, then the cell value is in the is element rather than the v

Enumeration Value	Description
	element in the cell (c element).
n (Number)	Cell containing a number.
s (Shared String)	Cell containing a shared string.
str (String)	Cell containing a formula string.

Referenced By
c@t (§3.3.1.3); cell@t (§3.14.1); nc@t (§3.11.1.3); oc@t (§3.11.1.5)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_CellType">
   <restriction base="xsd:string">
      <enumeration value="b"/>
      <enumeration value="n"/>
      <enumeration value="e"/>
      <enumeration value="s"/>
      <enumeration value="str"/>
      <enumeration value="inlineStr"/>
   </restriction>
</simpleType>
```

3.18.13 ST_CfType (Conditional Format Type)

Conditional format rule type.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
aboveAverage (Above or Below Average)	This conditional formatting rule highlights cells that are above or below the average for all values in the range.
beginsWith (Begins With)	This conditional formatting rule highlights cells in the range that begin with the given text. Equivalent to using the LEFT() sheet function and comparing values.
cellIs (Cell Is)	This conditional formatting rule compares a cell value to a formula calculated result, using an operator.
colorScale (Color Scale)	This conditional formatting rule creates a gradated color scale on the cells.
containsBlanks (Contains Blanks)	This conditional formatting rule highlights cells that are completely blank. Equivalent of using LEN(TRIM()). This means that if the cell contains only characters that TRIM() would remove, then it is considered blank.

Enumeration Value	Description
	An empty cell is also considered blank.
containsErrors (Contains Errors)	This conditional formatting rule highlights cells with formula errors. Equivalent to using ISERROR() sheet function to determine if there is a formula error.
containsText (Contains Text)	This conditional formatting rule highlights cells containing given text. Equivalent to using the SEARCH() sheet function to determine whether the cell contains the text.
dataBar (Data Bar)	This conditional formatting rule displays a gradated data bar in the range of cells.
duplicateValues (Duplicate Values)	This conditional formatting rule highlights duplicated values.
endsWith (Ends With)	This conditional formatting rule highlights cells ending with given text. Equivalent to using the RIGHT() sheet function and comparing values.
expression (Expression)	This conditional formatting rule contains a formula to evaluate. When the formula result is true, the cell is highlighted.
iconSet (Icon Set)	This conditional formatting rule applies icons to cells according to their values.
notContainsBlanks (Contains No Blanks)	This conditional formatting rule highlights cells that are not blank. Equivalent of using LEN(TRIM()). This means that if the cell contains only characters that TRIM() would remove, then it is considered blank. An empty cell is also considered blank.
notContainsErrors (Contains No Errors)	This conditional formatting rule highlights cells without formula errors. Equivalent to using ISERROR() sheet function to determine if there is a formula error.
notContainsText (Does Not Contain Text)	This conditional formatting rule highlights cells that do not contain given text. Equivalent to using the SEARCH() sheet function.
timePeriod (Time Period)	This conditional formatting rule highlights cells containing dates in the specified time period. The underlying value of the cell is evaluated, therefore the cell does not need to be formatted as a date to be evaluated. For example, with a cell containing the value 38913 the conditional format shall be applied if the rule requires a value of 7/14/2006.
top10 (Top 10)	This conditional formatting rule highlights cells whose values fall in the top N or bottom N bracket, as specified.

Enumeration Value	Description
uniqueValues (Unique Values)	This conditional formatting rule highlights unique values in the range.

Referenced By
cfRule@type (§3.3.1.9)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_CfType">
   <restriction base="xsd:string">
      <enumeration value="expression"/>
      <enumeration value="cellIs"/>
      <enumeration value="colorScale"/>
      <enumeration value="dataBar"/>
      <enumeration value="iconSet"/>
      <enumeration value="top10"/>
      <enumeration value="uniqueValues"/>
      <enumeration value="duplicateValues"/>
      <enumeration value="containsText"/>
      <enumeration value="notContainsText"/>
      <enumeration value="beginsWith"/>
      <enumeration value="endsWith"/>
      <enumeration value="containsBlanks"/>
      <enumeration value="notContainsBlanks"/>
      <enumeration value="containsErrors"/>
      <enumeration value="notContainsErrors"/>
      <enumeration value="timePeriod"/>
      <enumeration value="aboveAverage"/>
   </restriction>
</simpleType>
```

3.18.14 ST_CfvoType (Conditional Format Value Object Type)

This simple type expresses the type of the conditional formatting value object (cfvo). In general the cfvo specifies one value used in the gradated scale (max, min, midpoint, etc).

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
formula (Formula)	The minimum/ midpoint / maximum value for the gradient is determined by a formula.
max (Maximum)	Indicates that the maximum value in the range shall be used as the maximum value for the gradient.
min (Minimum)	Indicates that the minimum value in the range shall be used as the minimum value for the gradient.

Enumeration Value	Description
num (Number)	Indicates that the minimum / midpoint / maximum value for the gradient is specified by a constant numeric value.
percent (Percent)	Value indicates a percentage between the minimum and maximum values in the range shall be used as the minimum / midpoint / maximum value for the gradient.
percentile (Percentile)	Value indicates a percentile ranking in the range shall be used as the minimum / midpoint / maximum value for the gradient.

Referenced By
cfvo@type (§3.3.1.10)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_CfvoType">
   <restriction base="xsd:string">
      <enumeration value="num"/>
      <enumeration value="percent"/>
      <enumeration value="max"/>
      <enumeration value="min"/>
      <enumeration value="formula"/>
      <enumeration value="percentile"/>
   </restriction>
</simpleType>
```

3.18.15 ST_Comments (Comment Display Types)

This simple type defines options for displaying comments in the user interface.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
commIndAndComment (Show Comment & Indicator)	Indicates that both the comment indicator and comment text be show in the user interface.
commIndicator (Show Comment Indicator)	Indicates that only the comment indicator be shown in the user interface.
commNone (No Comments)	Indicates that comments not be shown in the user interface.

Referenced By

Referenced By
customWorkbookView@showComments (§3.2.3)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Comments">
   <restriction base="xsd:string">
      <enumeration value="commNone"/>
      <enumeration value="commIndicator"/>
      <enumeration value="commIndAndComment"/>
   </restriction>
</simpleType>
```

3.18.16 ST_ConditionalFormattingOperator (Conditional Format Operators)

These conditional format operators are used for "Highlight Cells That Contain…" rules. For example, "highlight cells that begin with "M2" and contain "Mountain Gear"".

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
beginsWith (Begins With)	'Begins with' operator
between (Between)	'Between' operator
containsText (Contains)	'Contains' operator
endsWith (Ends With)	'Ends with' operator
equal (Equal)	'Equal to' operator
greaterThan (Greater Than)	'Greater than' operator
greaterThanOrEqual (Greater Than Or Equal)	'Greater than or equal to' operator
lessThan (Less Than)	'Less than' operator
lessThanOrEqual (Less Than Or Equal)	'Less than or equal to' operator
notBetween (Not Between)	'Not between' operator
notContains (Does Not Contain)	'Does not contain' operator
notEqual (Not Equal)	'Not equal to' operator

Referenced By
cfRule@operator (§3.3.1.9)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ConditionalFormattingOperator">
   <restriction base="xsd:string">
      <enumeration value="lessThan"/>
      <enumeration value="lessThanOrEqual"/>
      <enumeration value="equal"/>
      <enumeration value="notEqual"/>
      <enumeration value="greaterThanOrEqual"/>
      <enumeration value="greaterThan"/>
      <enumeration value="between"/>
      <enumeration value="notBetween"/>
      <enumeration value="containsText"/>
      <enumeration value="notContains"/>
      <enumeration value="beginsWith"/>
      <enumeration value="endsWith"/>
   </restriction>
</simpleType>
```

3.18.17 ST_CredMethod (Credentials Method)

Credentials method used for server access.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
integrated (Integrated Authentication)	Integrated authentication.
none (No Credentials)	Use no credentials at all.
prompt (Prompt Credentials)	Prompt for credentials.
stored (Stored Credentials)	Use stored credentials.

Referenced By
connection@credentials (§3.13.1)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_CredMethod">
   <restriction base="xsd:string">
      <enumeration value="integrated"/>
      <enumeration value="none"/>
      <enumeration value="stored"/>
      <enumeration value="prompt"/>
   </restriction>
</simpleType>
```

3.18.18 ST_DataConsolidateFunction (Data Consolidation Functions)

Data consolidation functions specified by the user and used to consolidate ranges of data.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
average (Average)	The average of the values.
count (Count)	The number of data values. The Count consolidation function works the same as the COUNTA worksheet function.
countNums (CountNums)	The number of data values that are numbers. The Count Nums consolidation function works the same as the COUNT worksheet function.
max (Maximum)	The largest value.
min (Minimum)	The smallest value.
product (Product)	The product of the values.
stdDev (StdDev)	An estimate of the standard deviation of a population, where the sample is a subset of the entire population.
stdDevp (StdDevP)	The standard deviation of a population, where the population is all of the data to be summarized.
sum (Sum)	The sum of the values.
var (Variance)	An estimate of the variance of a population, where the sample is a subset of the entire population.
varp (VarP)	The variance of a population, where the population is all of the data to be summarized.

Referenced By
dataConsolidate@function (§3.3.1.27); dataField@subtotal (§3.10.1.22)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DataConsolidateFunction">
   <restriction base="xsd:string">
      <enumeration value="average"/>
      <enumeration value="count"/>
      <enumeration value="countNums"/>
      <enumeration value="max"/>
      <enumeration value="min"/>
      <enumeration value="product"/>
      <enumeration value="stdDev"/>
      <enumeration value="stdDevp"/>
      <enumeration value="sum"/>
      <enumeration value="var"/>
      <enumeration value="varp"/>
   </restriction>
</simpleType>
```

3.18.19 ST_DataValidationErrorStyle (Data Validation Error Styles)

The style of data validation error alert.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
information (Information Icon)	This data validation error style uses an information icon in the error alert. 
stop (Stop Icon)	This data validation error style uses a stop icon in the error alert. 
warning (Warning Icon)	This data validation error style uses a warning icon in the error alert. 

Referenced By
dataValidation@errorStyle (§3.3.1.30)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DataValidationErrorStyle">
   <restriction base="xsd:string">
      <enumeration value="stop"/>
      <enumeration value="warning"/>
      <enumeration value="information"/>
   </restriction>
</simpleType>
```

3.18.20 ST_DataValidationImeMode (Data Validation IME Mode)

These values specify that the IME (input method editor) mode is controlled by data validation.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
disabled (Disabled IME Mode)	IME mode is disabled. Forces the IME control to be disabled when this cell is selected.
fullAlpha (Full-Width Alpha-Numeric IME Mode)	Forces the IME control to be on and in full-width alph-numeric input mode when the cell is first selected.
fullHangul (Full Width Hangul)	Forces the IME control to be on and in full-width Hangul input mode when first selecting the cell. Applies when the application's language is Korean and a Korean IME control is selected.
fullKatakana (Full Katakana IME Mode)	Forces the IME control to be on and in full-width Katakana input mode when first selecting the cell. Applies when the application's language is Japanese and a Japanese IME control is selected.
halfAlpha (Half Alpha IME)	Forces the IME control to be on and in half-width alph-numeric input mode when the cell is first selected.
halfHangul (Half-Width Hangul IME Mode)	Forces the IME control to be on and in half-width Hangul input mode when first selecting the cell. Applies when the application's language is Korean and a Korean IME control is selected.
halfKatakana (Half-Width Katakana)	Forces the IME control to be on and in half-width Katakana input mode when first selecting the cell. Applies when the application's language is Japanese and a Japanese IME control is selected.
hiragana (Hiragana IME Mode)	Forces the IME control to be on and in Hiragana input mode when first selecting the cell. Applies when the application's language is Japanese and a Japanese IME control is selected
noControl (IME Mode Not Controlled)	Data validation does not control the IME control's

Enumeration Value	Description
	mode.
off (IME Off)	Forces the IME control to be off when first selecting the cell (goes to direct cell input mode).
on (IME On)	Forces the IME control to be on when first selecting the cell.

Referenced By
dataValidation@imeMode (§3.3.1.30)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DataValidationImeMode">
   <restriction base="xsd:string">
      <enumeration value="noControl"/>
      <enumeration value="off"/>
      <enumeration value="on"/>
      <enumeration value="disabled"/>
      <enumeration value="hiragana"/>
      <enumeration value="fullKatakana"/>
      <enumeration value="halfKatakana"/>
      <enumeration value="fullAlpha"/>
      <enumeration value="halfAlpha"/>
      <enumeration value="fullHangul"/>
      <enumeration value="halfHangul"/>
   </restriction>
</simpleType>
```

3.18.21 ST_DataValidationOperator (Data Validation Operator)

The relational operator used in data validation.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
between (Between)	Data validation which checks if a value is between two other values.
equal (Equal)	Data validation which checks if a value is equal to a specified value.
greaterThan (Greater Than)	Data validation which checks if a value is greater than a specified value.
greaterThanOrEqual (Greater Than Or Equal)	Data validation which checks if a value is greater than or equal to a specified value.
lessThan (Less Than)	Data validation which checks if a value is less than a

Enumeration Value	Description
	specified value.
lessThanOrEqual (Less Than Or Equal)	Data validation which checks if a value is less than or equal to a specified value.
notBetween (Not Between)	Data validation which checks if a value is not between two other values.
notEqual (Not Equal)	Data validation which checks if a value is not equal to a specified value.

Referenced By
dataValidation@operator (§3.3.1.30)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DataValidationOperator">
   <restriction base="xsd:string">
      <enumeration value="between"/>
      <enumeration value="notBetween"/>
      <enumeration value="equal"/>
      <enumeration value="notEqual"/>
      <enumeration value="lessThan"/>
      <enumeration value="lessThanOrEqual"/>
      <enumeration value="greaterThan"/>
      <enumeration value="greaterThanOrEqual"/>
   </restriction>
</simpleType>
```

3.18.22 ST_DataValidationType (Data Validation Type)

Specifies the type of data validation used to validate user input.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
custom (Custom)	Data validation which uses a custom formula to check the cell value.
date (Date)	Data validation which checks for date values satisfying the given condition.
decimal (Decimal)	Data validation which checks for decimal values satisfying the given condition.
list (List)	Data validation which checks for a value matching one of list of values.
none (None)	No data validation.

Enumeration Value	Description
textLength (Text Length)	Data validation which checks for text values, whose length satisfies the given condition.
time (Time)	Data validation which checks for time values satisfying the given condition.
whole (Whole Number)	Data validation which checks for whole number values satisfying the given condition.

Referenced By
dataValidation@type (§3.3.1.30)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DataValidationType">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="whole"/>
      <enumeration value="decimal"/>
      <enumeration value="list"/>
      <enumeration value="date"/>
      <enumeration value="time"/>
      <enumeration value="textLength"/>
      <enumeration value="custom"/>
   </restriction>
</simpleType>
```

3.18.23 ST_DateTimeGrouping (Date Time Grouping)

Specifies how to group dateTime values.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
day (Day)	Group by day
hour (Group by Hour)	Group by hour
minute (Group by Minute)	Group by minute
month (Month)	Group by month
second (Second)	Group by second
year (Group by Year)	Group by year

Referenced By

Referenced By
dateGroupItem@dateTimeGrouping (§3.3.2.4)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DateTimeGrouping">
   <restriction base="xsd:string">
      <enumeration value="year"/>
      <enumeration value="month"/>
      <enumeration value="day"/>
      <enumeration value="hour"/>
      <enumeration value="minute"/>
      <enumeration value="second"/>
   </restriction>
</simpleType>
```

3.18.24 ST_DdeValueType (DDE Value Types)

This simple type indicates the type of the DDE value.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
b (Boolean)	Indicates that the value is a boolean.
e (Error)	Indicates that the value is an error.
n (Real Number)	Indicates that the value is a real number.
nil (Nil)	Indicates that the value is nil.
str (String)	Indicates that the value is a string.

Referenced By
value@t (§3.14.18)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DdeValueType">
   <restriction base="xsd:string">
      <enumeration value="nil"/>
      <enumeration value="b"/>
      <enumeration value="n"/>
      <enumeration value="e"/>
      <enumeration value="str"/>
   </restriction>
</simpleType>
```

3.18.25 ST_DvAspect (Data View Aspect Type)

Specifies the desired data or view aspect of the object when drawing or getting data.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
DVASPECT_CONTENT (Object Display Content)	Provides a representation of an object so it can be displayed as an embedded object inside of a container.
DVASPECT_ICON (Object Display Icon)	Provides an iconic representation of an object.

Referenced By
oleObject@dvAspect (§3.3.1.57)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DvAspect">
   <restriction base="xsd:string">
      <enumeration value="DVASPECT_CONTENT"/>
      <enumeration value="DVASPECT_ICON"/>
   </restriction>
</simpleType>
```

3.18.26 ST_DxfId (Format Id)

This simple type defines the identifier to CT_Dxfs in the styles part. This a zero-based index. See §3.8.30 in Style for more information on formats.

This simple type's contents are a restriction of the XML Schema unsignedInt datatype.

Referenced By
cfRule@dxfId (§3.3.1.9); colorFilter@dxfId (§3.3.2.1); format@dxfId (§3.10.1.35); sortCondition@dxfId (§3.3.1.88); table@dataDxfId (§3.5.1.2); table@headerRowBorderDxfId (§3.5.1.2); table@headerRowDxfId (§3.5.1.2); table@tableBorderDxfId (§3.5.1.2); table@totalsRowBorderDxfId (§3.5.1.2); table@totalsRowDxfId (§3.5.1.2); tableColumn@dataDxfId (§3.5.1.3); tableColumn@headerRowDxfId (§3.5.1.3); tableColumn@totalsRowDxfId (§3.5.1.3); tableStyleElement@dxfId (§3.8.41)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DxfId">
   <restriction base="xsd:unsignedInt"/>
</simpleType>
```

3.18.27 ST_DynamicFilterType (Dynamic Filter)

These are the dynamic filter types. A dynamic filter returns a result set which may vary due to a change in the data itself or a change in the date on which the filter is being applied. For example, for a set of data {1,1,2,3}, the

aboveAverage filter would return or highlight the last two values in the set. If the data is refreshed or changed to {1,1,1,2}, then only the last value would be highlighted. Similarly, the meaning of "lastQuarter" shall be the same for the dates in January, February, and March, but shall change meaning once the date advances from March to April.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
aboveAverage (Above Average)	Shows values that are above average.
belowAverage (Below Average)	Shows values that are below average.
lastMonth (Last Month)	Shows last month's dates.
lastQuarter (Last Quarter)	Shows last quarter's dates.
lastWeek (Last Week)	Shows last week's dates.
lastYear (Last Year)	Shows last year's dates.
M1 (1st Month)	Shows the dates that are in January, regardless of year.
M10 (10th Month)	Shows the dates that are in October, regardless of year.
M11 (11th Month)	Shows the dates that are in November, regardless of year.
M12 (12th Month)	Shows the dates that are in December, regardless of year.
M2 (2nd Month)	Shows the dates that are in Februray, regardless of year.
M3 (3rd Month)	Shows the dates that are in March, regardless of year.
M4 (4th Month)	Shows the dates that are in April, regardless of year.
M5 (5th Month)	Shows the dates that are in May, regardless of year.
M6 (6th Month)	Shows the dates that are in June, regardless of year.
M7 (7th Month)	Shows the dates that are in July, regardless of year.
M8 (8th Month)	Shows the dates that are in August, regardless of year.
M9 (9th Month)	Shows the dates that are in September, regardless of year.
nextMonth (Next Month)	Shows next month's dates.
nextQuarter (Next Quarter)	Shows next quarter's dates.
nextWeek (Next Week)	Shows next week's dates.
nextYear (Next Year)	Shows next year's dates.
null (Null)	Common filter type not available.

Enumeration Value	Description
Q1 (1st Quarter)	Shows the dates that are in the 1st quarter, regardless of year.
Q2 (2nd Quarter)	Shows the dates that are in the 2nd quarter, regardless of year.
Q3 (3rd Quarter)	Shows the dates that are in the 3rd quarter, regardless of year.
Q4 (4th Quarter)	Shows the dates that are in the 4th quarter, regardless of year.
thisMonth (This Month)	Shows this month's dates.
thisQuarter (This Quarter)	Shows this quarter's dates.
thisWeek (This Week)	Shows this week's dates.
thisYear (This Year)	Shows this year's dates.
today (Today)	Shows today's dates.
tomorrow (Tomorrow)	Shows tomorrow's dates.
yearToDate (Year To Date)	Shows the dates between the beginning of the year and today, inclusive.
yesterday (Yesterday)	Shows yesterday's dates.

Referenced By
dynamicFilter@type (§3.3.2.5)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DynamicFilterType">
   <restriction base="xsd:string">
      <enumeration value="null"/>
      <enumeration value="aboveAverage"/>
      <enumeration value="belowAverage"/>
      <enumeration value="tomorrow"/>
      <enumeration value="today"/>
      <enumeration value="yesterday"/>
      <enumeration value="nextWeek"/>
      <enumeration value="thisWeek"/>
      <enumeration value="lastWeek"/>
      <enumeration value="nextMonth"/>
      <enumeration value="thisMonth"/>
      <enumeration value="lastMonth"/>
      <enumeration value="nextQuarter"/>
      <enumeration value="thisQuarter"/>
      <enumeration value="lastQuarter"/>
      <enumeration value="nextYear"/>
      <enumeration value="thisYear"/>
      <enumeration value="lastYear"/>
      <enumeration value="yearToDate"/>
      <enumeration value="Q1"/>
      <enumeration value="Q2"/>
      <enumeration value="Q3"/>
      <enumeration value="Q4"/>
      <enumeration value="M1"/>
      <enumeration value="M2"/>
      <enumeration value="M3"/>
      <enumeration value="M4"/>
      <enumeration value="M5"/>
      <enumeration value="M6"/>
      <enumeration value="M7"/>
      <enumeration value="M8"/>
      <enumeration value="M9"/>
      <enumeration value="M10"/>
      <enumeration value="M11"/>
      <enumeration value="M12"/>
   </restriction>
</simpleType>
```

3.18.28 ST_ExternalConnectionType (Text Field Datatype)

These are the possible data types to use when importing text into the SpreadsheetML document. Strings are converted to these data types in the worksheet.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description

Enumeration Value	Description
DMY (Day Month Year)	Field contains a date in the order: day, month, year.
DYM (Day Year Month)	Field contains a date in the order: day, year, month.
EMD (East Asian Year Month Day)	Field contains an East Asian date in the order: EA era year, month, day.
general (General)	The SpreadsheetML application decides the best fit data type based on the content.
MDY (Month Day Year)	Field contains a date in the order: month, day, year.
MYD (Month Day Year)	Field contains a date in the order: month, year, day.
skip (Skip Field)	Don't import this field at all.
text (Text)	Field contains text.
YDM (Year Day Month)	Field contains a date in the order: year, day, month.
YMD (Year Month Day)	Field contains a date in the order: year, month, day.

Referenced By
textField@type (§3.13.10)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ExternalConnectionType">
   <restriction base="xsd:string">
      <enumeration value="general"/>
      <enumeration value="text"/>
      <enumeration value="MDY"/>
      <enumeration value="DMY"/>
      <enumeration value="YMD"/>
      <enumeration value="MYD"/>
      <enumeration value="DYM"/>
      <enumeration value="YDM"/>
      <enumeration value="skip"/>
      <enumeration value="EMD"/>
   </restriction>
</simpleType>
```

3.18.29 ST_FieldSortType (Field Sort Type)

This simple type defines the sort orders that can be applied to fields in a PivotTable.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
ascending (Ascending)	Indicates the field is sorted in ascending order.

Enumeration Value	Description
descending (Descending)	Indicates the field is sorted in descending order.
manual (Manual Sort)	Indicates the field is sorted manually.

Referenced By
pivotField@sortType (§3.10.1.69)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FieldSortType">
   <restriction base="xsd:string">
      <enumeration value="manual"/>
      <enumeration value="ascending"/>
      <enumeration value="descending"/>
   </restriction>
</simpleType>
```

3.18.30 ST_FileType (File Type)

The file type being used for text import.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
dos (DOS)	DOS (PC-8).
mac (Macintosh)	Macintosh.
win (Windows (ANSI))	Windows (ANSI).

Referenced By
textPr@fileType (§3.13.12)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FileType">
   <restriction base="xsd:string">
      <enumeration value="mac"/>
      <enumeration value="win"/>
      <enumeration value="dos"/>
   </restriction>
</simpleType>
```

3.18.31 ST_FillId (Fill Id)

Zero-based index used to reference a fill record.

This simple type's contents are a restriction of the XML Schema unsignedInt datatype.

Referenced By
xf@fillId (§3.8.45)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FillId">
    <restriction base="xsd:unsignedInt"/>
</simpleType>
```

3.18.32 ST_FilterOperator (Filter Operator)

Operator enumerations for filtering.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
equal (Equal)	Show results which are equal to criteria.
greaterThan (Greater Than)	Show results which are greater than criteria.
greaterThanOrEqual (Greater Than Or Equal)	Show results which are greater than or equal to criteria.
lessThan (Less Than)	Show results which are less than criteria.
lessThanOrEqual (Less Than Or Equal)	Show results which are less than or equal to criteria.
notEqual (Not Equal)	Show results which are not equal to criteria.

Referenced By
customFilter@operator (§3.3.2.2)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FilterOperator">
    <restriction base="xsd:string">
        <enumeration value="equal"/>
        <enumeration value="lessThan"/>
        <enumeration value="lessThanOrEqual"/>
        <enumeration value="notEqual"/>
        <enumeration value="greaterThanOrEqual"/>
        <enumeration value="greaterThan"/>
    </restriction>
</simpleType>
```

3.18.33 ST_FontId (Font Id)

An integer that represents a zero based index into the <fonts> collection in the style sheet.

This simple type's contents are a restriction of the XML Schema unsignedInt datatype.

Referenced By
phoneticPr@fontId (§3.4.3); xf@fontId (§3.8.45)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FontId">
   <restriction base="xsd:unsignedInt"/>
</simpleType>
```

3.18.34 ST_FontScheme (Font scheme Styles)

Defines the font scheme, if any, to which this font belongs.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
major (Major Font)	This font is the major font for this theme.
minor (Minor Font)	This font is the minor font for this theme.
none (None)	This font is not a theme font.

Referenced By
scheme@val (§3.8.36)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FontScheme">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="major"/>
      <enumeration value="minor"/>
   </restriction>
</simpleType>
```

3.18.35 ST_FormatAction (PivotTable Format Types)

This simple type defines the type of formats that can be applied to PivotTables.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
blank (Blank)	Indicates no format is applied to the PivotTable. This value is used when formatting is cleared from already

Enumeration Value	Description
	formatted cells in the PivotTable.
drill (Drill Type)	Indicates the PivotTable has drill-through format.
formatting (Formatting)	Indicates the PivotTable has formatting.
formula (Formula Type)	Indicates the PivotTable has formulas.

Referenced By
format@action (§3.10.1.35)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FormatAction">
   <restriction base="xsd:string">
      <enumeration value="blank"/>
      <enumeration value="formatting"/>
      <enumeration value="drill"/>
      <enumeration value="formula"/>
   </restriction>
</simpleType>
```

3.18.36 ST_Formula (Formula)

A formula

This simple type's contents are a restriction of the ST_Xstring simple type (§3.18.96).

Referenced By
formula (§3.3.1.40); formula1 (§3.3.1.41); formula2 (§3.3.1.42); oldFormula (§3.11.1.7)

3.18.37 ST_FormulaExpression (Formula Expression Type)

This simple type specifies an expression type that can comprise a formula.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
area (Area)	Reference to a range of cells.
areaError (Area Error)	Reference to a range of cells that now evaluates to an error.
computedArea (Computed Area)	Computed area reference.
ref (Reference)	Single cell reference.
refError (Reference Is Error)	Single cell reference that now evaluates to an error.

Referenced By
undo@exp (§3.11.1.25)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FormulaExpression">
   <restriction base="xsd:string">
      <enumeration value="ref"/>
      <enumeration value="refError"/>
      <enumeration value="area"/>
      <enumeration value="areaError"/>
      <enumeration value="computedArea"/>
   </restriction>
</simpleType>
```

3.18.38 ST_GradientType (Gradient Type)

Type of gradient fill being used, either linear or path.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
linear (Linear Gradient)	This gradient fill is of linear gradient type. Linear gradient type means that the transition from one color to the next is along a line (e.g., horizontal, vertical, diagonal, etc.).
path (Path)	This gradient fill is of path gradient type. Path gradient type means the that the boundary of transition from one color to the next is a rectangle, defined by top, bottom, left, and right attributes on the gradientFill element.

Referenced By
gradientFill@type (§3.8.23)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_GradientType">
   <restriction base="xsd:string">
      <enumeration value="linear"/>
      <enumeration value="path"/>
   </restriction>
</simpleType>
```

3.18.39 ST_GroupBy (Values Group By)

This simple type defines types of data grouping that can be performed on a PivotTable.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
days (Days)	Indicates a grouping on "days" for date values.
hours (Hours)	Indicates a grouping on "hours" for date values.
minutes (Minutes)	Indicates a grouping on "minutes" for date values.
months (Months)	Indicates a grouping on "months" for date values.
quarters (Quarters)	Indicates a grouping on "quarters" for date values.
range (Group By Numeric Ranges)	Indicates a grouping by numeric ranges for numeric values.
seconds (Seconds)	Indicates a grouping on "seconds" for date values.
years (Years)	Indicates a grouping on "years" for date values.

Referenced By
rangePr@groupBy (§3.10.1.78)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_GroupBy">
   <restriction base="xsd:string">
      <enumeration value="range"/>
      <enumeration value="seconds"/>
      <enumeration value="minutes"/>
      <enumeration value="hours"/>
      <enumeration value="days"/>
      <enumeration value="months"/>
      <enumeration value="quarters"/>
      <enumeration value="years"/>
   </restriction>
</simpleType>
```

3.18.40 ST_GrowShrinkType (Grow Shrink Type)

This type enumerates behavior patterns for refreshing external data in a query table.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
insertClear (Insert & Clear On Refresh)	Insert entire rows for new data, clear unused cells.
insertDelete (Insert & Delete On Refresh)	Insert cells for new data, delete unused cells.

Enumeration Value	Description
overwriteClear (Overwrite & Clear On Refresh)	Overwrite existing cells with new data, clear unused cells.

Referenced By
queryTable@growShrinkType (§3.12.2)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_GrowShrinkType">
   <restriction base="xsd:string">
      <enumeration value="insertDelete"/>
      <enumeration value="insertClear"/>
      <enumeration value="overwriteClear"/>
   </restriction>
</simpleType>
```

3.18.41 ST_Guid (Globally Unique Identifier)

This simple type defines a 128 bit Globally Unique Identifier (GUID).

This simple type's contents are a restriction of the XML Schema token datatype.

This simple type also specifies the following restrictions:

- This simple type's contents must match the following regular expression pattern: \{[0-9A-F]{8}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{12}\}.

Referenced By
comment@guid (§3.7.3); customSheetView@guid (§3.3.1.22); customSheetView@guid (§3.3.1.23); customWorkbookView@guid (§3.2.3); fileVersion@codeName (§3.2.13); header@guid (§3.11.1.1); headers@guid (§3.11.1.2); headers@lastGuid (§3.11.1.2); rcmt@guid (§3.11.1.11); rcv@guid (§3.11.1.12); userInfo@guid (§3.11.2.1)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Guid">
   <restriction base="xsd:token">
      <pattern value="\{[0-9A-F]{8}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{12}\}"/>
   </restriction>
</simpleType>
```

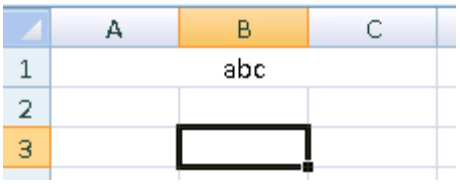
3.18.42 ST_HorizontalAlignment (Horizontal Alignment Type)

The enumeration value indicating the portion of Cell Alignment in a cell format (XF) that is horizontal alignment, i.e., whether it is aligned general, left, right, horizontally centered, filled (replicated), justified, centered across multiple cells, or distributed.

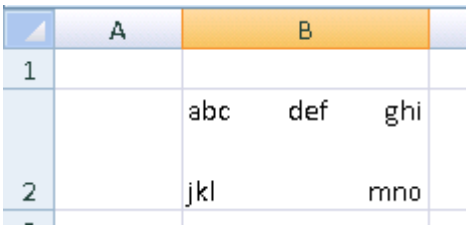
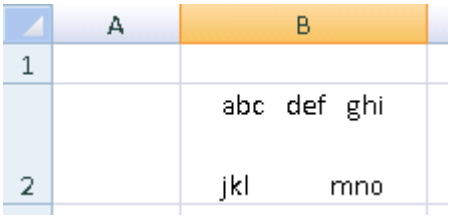
This simple type's contents are a restriction of the XML Schema string datatype.

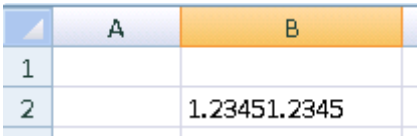
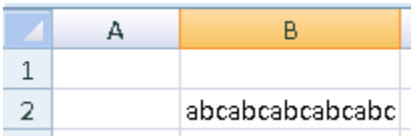
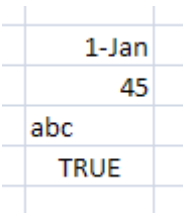
The following are possible enumeration values for this type:

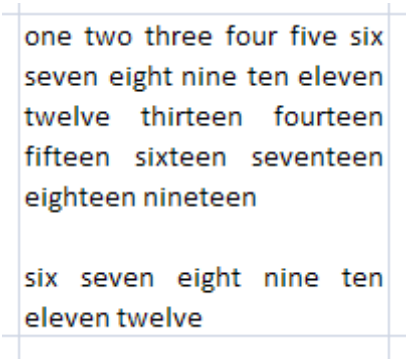
Enumeration Value	Description
center (Centered Horizontal Alignment)	The horizontal alignment is centered, meaning the text is centered across the cell.

centerContinuous (Center Continuous Horizontal Alignment)	The horizontal alignment is centered across multiple cells. The information about how many cells to span is expressed in the Sheet Part, in the row of the cell in question. For each cell that is spanned in the alignment, a cell element needs to be written out, with the same style Id which references the `centerContinuous` alignment.
	
	[*Example*:
	This shows the value of A1 centered across A1:C1:
	
	 
	
	The XML from the Sheet Part:
	
	`<row r="1" spans="1:3">`
	`<c r="A1" s="1" t="s">`
	`<v>0</v>`
	`</c>`
	`<c r="B1" s="1"/>`
	`<c r="C1" s="1"/>`
	`</row>`
	The XML from the Styles Part:
	
	`<cellXfs count="2">`
	`<xf numFmtId="0" fontId="0" fillId="0" borderId="0" xfId="0"/>`
	`<xf numFmtId="0" fontId="0" fillId="0" borderId="0" xfId="0" applyAlignment="1">`
	`<alignment horizontal="centerContinuous"/>`
	`</xf>`
	`</cellXfs>`
	
	end example]

distributed (Distributed Horizontal Alignment)	I/ndicates that each 'word' in each line of text inside

Enumeration Value	Description
	the cell is evenly distributed across the width of the cell, with flush right and left margins. When there is also an indent value to apply, both the left and right side of the cell are padded by the indent value. A 'word' is a set of characters with no space character in them. Two lines inside a cell are separated by a carriage return. [*Example*: This shows three lines of text evenly distributed horizontally across the cell. The first line is "abc def ghi", the second line is blank, and the third line is "jkl mno".  This shows the same example, with an indent value of 2:  Note: there is no vertical component to the alignment being shown here. The row has been manually adjusted to display the text. *end example*]
fill (Fill)	Indicates that the value of the cell should be filled across the entire width of the cell. If blank cells to the right also have the fill alignment, they are also filled with the value, using a convention similar to `centerContinuous`.

Enumeration Value	Description
	Additional rules: • Only whole values can be appended, not partial values. • The column will not be widened to 'best fit' the filled value • If appending an additional occurrence of the value exceeds the boundary of the cell left/right edge, don't append the additional occurrence of the value. • The display value of the cell is filled, not the underlying raw number. [*Example*: This cell is filled with the value 1.2345 and has a width of 15 characters:  This cell is filled with the value abc and has width of 15 characters:  *end example*]
general (General Horizontal Alignment)	The horizontal alignment is general-aligned. Text data is left-aligned. Numbers, dates, and times are right-aligned. Boolean types are centered. Changing the alignment does not change the type of data. [*Example*: These cells are `general` aligned:  *end example*]

Enumeration Value	Description
justify (Justify)	The horizontal alignment is justified (flush left and right). For each line of text, aligns each line of the wrapped text in a cell to the right and left (except the last line). If no single line of text wraps in the cell, then the text is not justified. [*Example*: There are two lines of text in this cell, and the cell's horizontal alignment is `justify`:  *end example*]
left (Left Horizontal Alignment)	The horizontal alignment is left-aligned, even in Right-to-Left mode. Aligns contents at the left edge of the cell. If an indent amount is specified, the contents of the cell is indented from the left by the specified number of character spaces. The character spaces are based on the default font and font size for the workbook.
right (Right Horizontal Alignment)	The horizontal alignment is right-aligned, meaning that cell contents are aligned at the right edge of the cell, even in Right-to-Left mode.

Referenced By
alignment@horizontal (§3.8.1)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_HorizontalAlignment">
   <restriction base="xsd:string">
      <enumeration value="general"/>
      <enumeration value="left"/>
      <enumeration value="center"/>
      <enumeration value="right"/>
      <enumeration value="fill"/>
      <enumeration value="justify"/>
      <enumeration value="centerContinuous"/>
      <enumeration value="distributed"/>
   </restriction>
</simpleType>
```

3.18.43 ST_HtmlFmt (HTML Formatting Handling)

How to handle formatting from the HTML source.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
all (All)	Transfer all HTML formatting into the worksheet along with data.
none (No Formatting)	Bring data in as unformatted text (setting data types still occurs).
rtf (Honor Rich Text)	Translate HTML formatting to rich text formatting on the data brought into the worksheet.

Referenced By
webPr@htmlFormat (§3.13.13)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_HtmlFmt">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="rtf"/>
      <enumeration value="all"/>
   </restriction>
</simpleType>
```

3.18.44 ST_IconSetType (Icon Set Type)

Icon set type for conditional formatting. The threshold values for triggering the different icons within a set are configurable, and the icon order is reversible. See element iconSet for more information.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
3Arrows (3 Arrows)	3 arrows icon set. 
3ArrowsGray (3 Arrows (Gray))	3 gray arrows icon set. 
3Flags (3 Flags)	3 flags icon set. 
3Signs (3 Signs)	3 signs icon set. 
3Symbols (3 Symbols Circled)	3 symbols icon set. 
3Symbols2 (3 Symbols)	3 Symbols icon set. 
3TrafficLights1 (3 Traffic Lights)	3 traffic lights icon set (#1). 
3TrafficLights2 (3 Traffic Lights Black)	3 traffic lights icon set with thick black border. 

Enumeration Value	Description
4Arrows (4 Arrows)	4 arrows icon set. 
4ArrowsGray (4 Arrows (Gray))	4 gray arrows icon set. 
4Rating (4 Ratings)	4 ratings icon set. 
4RedToBlack (4 Red To Black)	4 'red to black' icon set. 
4TrafficLights (4 Traffic Lights)	4 traffic lights icon set. 
5Arrows (5 Arrows)	5 arrows icon set. 
5ArrowsGray (5 Arrows (Gray))	5 gray arrows icon set. 
5Quarters (5 Quarters)	5 quarters icon set. 
5Rating (5 Ratings Icon Set)	5 rating icon set.

Enumeration Value	Description
	

Referenced By
iconFilter@iconSet (§3.3.2.9); iconSet@iconSet (§3.3.1.46); sortCondition@iconSet (§3.3.1.88)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_IconSetType">
   <restriction base="xsd:string">
      <enumeration value="3Arrows"/>
      <enumeration value="3ArrowsGray"/>
      <enumeration value="3Flags"/>
      <enumeration value="3TrafficLights1"/>
      <enumeration value="3TrafficLights2"/>
      <enumeration value="3Signs"/>
      <enumeration value="3Symbols"/>
      <enumeration value="3Symbols2"/>
      <enumeration value="4Arrows"/>
      <enumeration value="4ArrowsGray"/>
      <enumeration value="4RedToBlack"/>
      <enumeration value="4Rating"/>
      <enumeration value="4TrafficLights"/>
      <enumeration value="5Arrows"/>
      <enumeration value="5ArrowsGray"/>
      <enumeration value="5Rating"/>
      <enumeration value="5Quarters"/>
   </restriction>
</simpleType>
```

3.18.45 ST_ItemType (PivotItem Type)

This simple type defines the type for a pivotItem.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
avg (Average)	Indicates the pivot item represents an "average" aggregate function.
blank (Blank Pivot Item)	Indicates the pivot item represents a blank line.
count (Count)	Indicates the pivot item represents custom the "count" aggregate."
countA (CountA)	Indicates the pivot item represents the "count

Enumeration Value	Description
	numbers" aggregate function.
data (Data)	Indicate the pivot item represents data.
default (Default)	Indicates the pivot item represents the default type for this PivotTable. The default pivot item type is the "total" aggregate function.
grand (Grand Total Item)	Indicates the pivot items represents the grand total line.
max (Max)	Indicates the pivot item represents the "maximum" aggregate function.
min (Min)	Indicates the pivot item represents the "minimum" aggregate function.
product (Product)	Indicates the pivot item represents the "product" function.
stdDev (stdDev)	Indicates the pivot item represents the "standard deviation" aggregate function.
stdDevP (StdDevP)	Indicates the pivot item represents the "standard deviation population" aggregate function.
sum (Sum)	Indicates the pivot item represents the "sum" aggregate value.
var (Var)	Indicates the pivot item represents the "variance" aggregate value.
varP (VarP)	Indicates the pivot item represents the "variance population" aggregate value.

Referenced By
i@t (§3.10.1.44); item@t (§3.10.1.45)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ItemType">
   <restriction base="xsd:string">
      <enumeration value="data"/>
      <enumeration value="default"/>
      <enumeration value="sum"/>
      <enumeration value="countA"/>
      <enumeration value="avg"/>
      <enumeration value="max"/>
      <enumeration value="min"/>
      <enumeration value="product"/>
      <enumeration value="count"/>
      <enumeration value="stdDev"/>
      <enumeration value="stdDevP"/>
      <enumeration value="var"/>
      <enumeration value="varP"/>
      <enumeration value="grand"/>
      <enumeration value="blank"/>
   </restriction>
</simpleType>
```

3.18.46 ST_MdxFunctionType (MDX Function Type)

This simple type is an enumeration representing different MDX function types.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
c (Cube Set Count)	CUBESETCOUNT
k (Cube KPI Member)	CUBEKPIMEMBER
m (Cube Member)	CUBEMEMBER
p (Cube Member Property)	CUBEMEMBERPROPERTY
r (Cube Ranked Member)	CUBERANKEDMEMBER
s (Cube Set)	CUBESET
v (Cube Value)	CUBEVALUE

Referenced By
mdx@f (§3.9.6)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_MdxFunctionType">
   <restriction base="xsd:string">
      <enumeration value="m"/>
      <enumeration value="v"/>
      <enumeration value="s"/>
      <enumeration value="c"/>
      <enumeration value="r"/>
      <enumeration value="p"/>
      <enumeration value="k"/>
   </restriction>
</simpleType>
```

3.18.47 ST_MdxKPIProperty (MDX KPI Property)

An enumeration representing the different types of KPI properties.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
g (Goal)	Goal.
m (Current Time Member)	Current time member.
s (Status)	Status.
t (Trend)	Trend.
v (Value)	Value.
w (Weight)	Weight.

Referenced By
k@p (§3.9.5)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_MdxKPIProperty">
   <restriction base="xsd:string">
      <enumeration value="v"/>
      <enumeration value="g"/>
      <enumeration value="s"/>
      <enumeration value="t"/>
      <enumeration value="w"/>
      <enumeration value="m"/>
   </restriction>
</simpleType>
```

3.18.48 ST_MdxSetOrder (MDX Set Order)

This type represents an enumeration specifying an MDX set order.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
a (Ascending)	Sort ascending.
aa (Alpha Ascending Sort Order)	Sorted alphabetically in ascending order by the caption.
ad (Alpha Descending Sort Order)	Sort in descending order alphabetically by the caption.
d (Descending)	Sort descending.
na (Natural Ascending)	Sorted in ascending order by the natural order of the data - usually by the key. For instance if there is a list of accounts in a general ledger, this may be in order of account number.
nd (Natural Descending)	Sorted in descending order by the natural order of the data - usually by the key. For instance if there is a list of accounts in a general ledger, this may be in order of account number.
u (Unsorted)	Unsorted.

Referenced By
ms@o (§3.9.12)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_MdxSetOrder">
   <restriction base="xsd:string">
      <enumeration value="u"/>
      <enumeration value="a"/>
      <enumeration value="d"/>
      <enumeration value="aa"/>
      <enumeration value="ad"/>
      <enumeration value="na"/>
      <enumeration value="nd"/>
   </restriction>
</simpleType>
```

3.18.49 ST_NumFmtId (Number Format Id)

This simple type defines the identifier to a style sheet number format entry in CT_NumFmts. Number formats are written to the styles part. See §1.1.31 in Style for more information on number formats.

This simple type's contents are a restriction of the XML Schema unsignedInt datatype.

Referenced By

Referenced By
cacheField@numFmtId (§3.10.1.3); dataField@numFmtId (§3.10.1.22); inputCells@numFmtId (§3.3.1.49); numFmt@numFmtId (§3.8.30); pivotField@numFmtId (§3.10.1.69); rcc@numFmtId (§3.11.1.9); xf@numFmtId (§3.8.45)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_NumFmtId">
   <restriction base="xsd:unsignedInt"/>
</simpleType>
```

3.18.50 ST_Objects (Object Display Types)

This simple type defines how the application displays objects in this workbook. Objects might include charts, images, and other object data that the application supports.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
all (All)	Indicates that all objects be shown in the workbook.
none (None)	Indicates that all objects be hidden in the workbook.
placeholders (Show Placeholders)	Indicates that the application show placeholders for objects in the workbook.

Referenced By
customWorkbookView@showObjects (§3.2.3); workbookPr@showObjects (§3.2.28)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Objects">
   <restriction base="xsd:string">
      <enumeration value="all"/>
      <enumeration value="placeholders"/>
      <enumeration value="none"/>
   </restriction>
</simpleType>
```

3.18.51 ST_OleUpdate (OLE Update Types)

Indicates whether the linked object updates the cached data for the linked object automatically or only when the container calls IOleObject::Update or IOleLink::Update methods.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
OLEUPDATE_ALWAYS (Always Update OLE)	Update the link object whenever possible, this option corresponds to the 'automatic update' option in the Links dialog box.
OLEUPDATE_ONCALL (Update OLE On Call)	Update the link object only when IOleObject::Update or IOleLink::Update is called, this option coresponds to the Manual update option in the Links dialog box.

Referenced By
oleObject@oleUpdate (§3.3.1.57)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_OleUpdate">
   <restriction base="xsd:string">
      <enumeration value="OLEUPDATE_ALWAYS"/>
      <enumeration value="OLEUPDATE_ONCALL"/>
   </restriction>
</simpleType>
```

3.18.52 ST_Orientation (Orientation)

Print orientation for this sheet.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
default (Default)	Orientation not specified, use the default.
landscape (Landscape)	Landscape orientation.
portrait (Portrait)	Portrait orientation.

Referenced By
pageSetup@orientation (§3.3.1.62); pageSetup@orientation (§3.3.1.61)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Orientation">
   <restriction base="xsd:string">
      <enumeration value="default"/>
      <enumeration value="portrait"/>
      <enumeration value="landscape"/>
   </restriction>
</simpleType>
```

3.18.53 ST_PageOrder (Page Order)

Specifies printed page order.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
downThenOver (Down Then Over)	Order pages vertically first, then move horizontally.
overThenDown (Over Then Down)	Order pages horizontally first, then move vertically

Referenced By
pageSetup@pageOrder (§3.3.1.61)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PageOrder">
   <restriction base="xsd:string">
      <enumeration value="downThenOver"/>
      <enumeration value="overThenDown"/>
   </restriction>
</simpleType>
```

3.18.54 ST_Pane (Pane Types)

Defines the names of the four possible panes into which the view of a workbook in the application can be split.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
bottomLeft (Bottom Left Pane)	Bottom left pane, when both vertical and horizontal splits are applied. This value is also used when only a horizontal split has been applied, dividing the pane into upper and lower regions. In that case, this value specifies the bottom pane.
bottomRight (Bottom Right Pane)	Bottom right pane, when both vertical and horizontal splits are applied.
topLeft (Top Left Pane)	Top left pane, when both vertical and horizontal splits are applied. This value is also used when only a horizontal split has been applied, dividing the pane into upper and lower

Enumeration Value	Description
	regions. In that case, this value specifies the top pane. This value is also used when only a vertical split has been applied, dividing the pane into right and left regions. In that case, this value specifies the left pane
topRight (Top Right Pane)	Top right pane, when both vertical and horizontal splits are applied. This value is also used when only a vertical split has been applied, dividing the pane into right and left regions. In that case, this value specifies the right pane.

Referenced By
pane@activePane (§3.3.1.64); pivotSelection@pane (§3.3.1.67); selection@pane (§3.3.1.75)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Pane">
   <restriction base="xsd:string">
      <enumeration value="bottomRight"/>
      <enumeration value="topRight"/>
      <enumeration value="bottomLeft"/>
      <enumeration value="topLeft"/>
   </restriction>
</simpleType>
```

3.18.55 ST_PaneState (Pane State)

State of the sheet's pane.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
frozen (Frozen)	Panes are frozen, but were not split being frozen. In this state, when the panes are unfrozen again, a single pane results, with no split. In this state, the split bars are not adjustable.
frozenSplit (Frozen Split)	Panes are frozen and were split before being frozen. In this state, when the panes are unfrozen again, the split remains, but is adjustable.
split (Split)	Panes are split, but not frozen. In this state, the split

Enumeration Value	Description
	bars are adjustable by the user.

Referenced By
pane@state (§3.3.1.64)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PaneState">
   <restriction base="xsd:string">
      <enumeration value="split"/>
      <enumeration value="frozen"/>
      <enumeration value="frozenSplit"/>
   </restriction>
</simpleType>
```

3.18.56 ST_ParameterType (Parameter Type)

Parameter Type.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
cell (Parameter From Cell)	Get the parameter value from a cell on each refresh.
prompt (Prompt on Refresh)	Prompt the user on each refresh for a parameter value.
value (Value)	Use a constant value on each refresh for the parameter value.

Referenced By
parameter@parameterType (§3.13.6)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ParameterType">
   <restriction base="xsd:string">
      <enumeration value="prompt"/>
      <enumeration value="value"/>
      <enumeration value="cell"/>
   </restriction>
</simpleType>
```

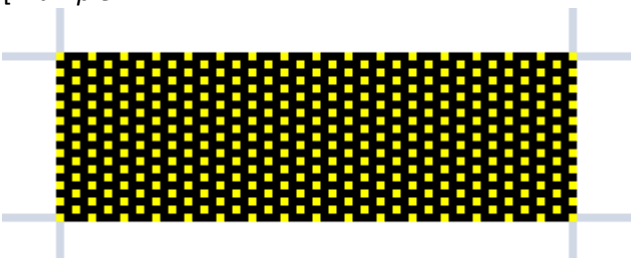
3.18.57 ST_PatternType (Pattern Type)

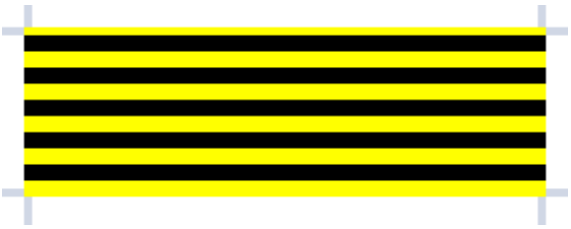
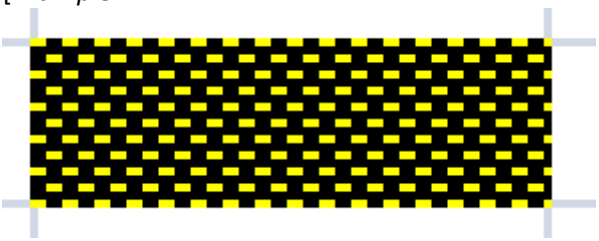
Indicates the style of fill pattern being used for a cell format.

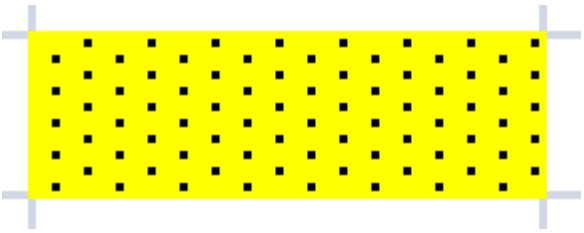
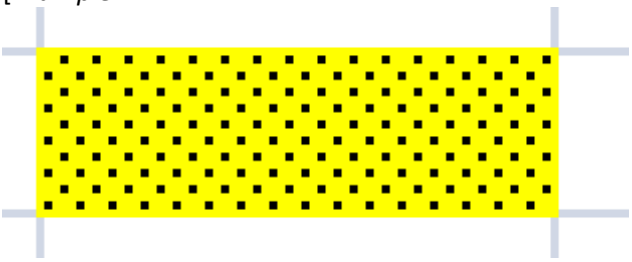
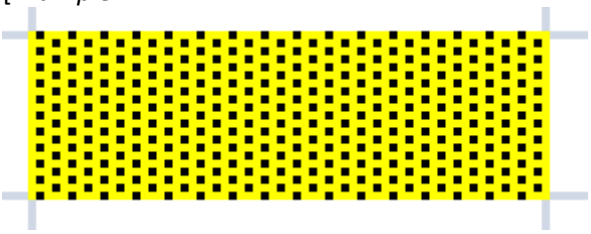
The examples below use yellow background and black foreground colors.

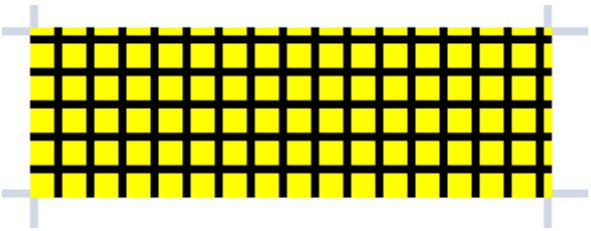
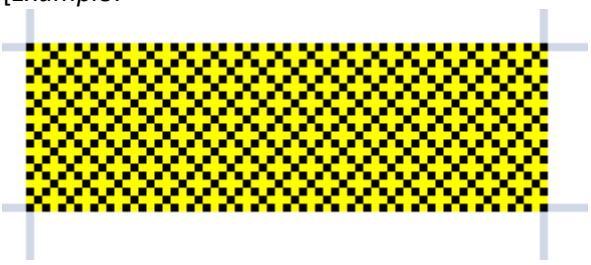
This simple type's contents are a restriction of the XML Schema string datatype.

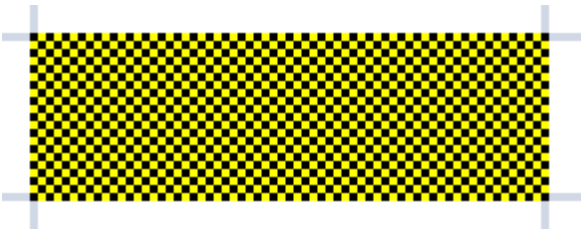
The following are possible enumeration values for this type:

Enumeration Value	Description
darkDown (Dark Down)	The fill style is 'dark down'. [*Example*:  *end example*]
darkGray (Dary Gray)	The fill style is 'dark gray'. [*Example*:  *end example*]
darkGrid (Dark Grid)	The fill style is 'dark grid'. [*Example*:  *end example*]
darkHorizontal (Dark Horizontal)	The fill style is dark horizontal. [*Example*:

Enumeration Value	Description
	 *end example*]
darkTrellis (Dark Trellis)	The fill style is 'dark trellis'. [*Example*:  *end example*]
darkUp (Dark Up)	The fill style is 'dark up'. [*Example*:  *end example*]
darkVertical (Dark Vertical)	The fill style is 'dark vertical'. [*Example*:  *end example*]
gray0625 (Gray 0.0625)	The fill style is grayscale of 0.0625 (1/16) value. [*Example*:

Enumeration Value	Description
	 *end example*]
gray125 (Gray 0.125)	The fill style is grayscale of 0.125 (1/8) value. [*Example*:  *end example*]
lightDown (Light Down)	The fill style is 'light down'. [*Example*:  *end example*]
lightGray (Light Gray)	The fill style is light gray. [*Example*:  *end example*]
lightGrid (Light Grid)	The fill style is 'light grid'. [*Example*:

Enumeration Value	Description
	 *end example*]
lightHorizontal (Light Horizontal)	The fill style is light horizontal. [*Example*:  *end example*]
lightTrellis (Light Trellis)	The fill style is 'light trellis'. [*Example*:  *end example*]
lightUp (Light Up)	The fill style is light up. [*Example*:  *end example*]
lightVertical (Light Vertical)	The fill style is light vertical.
mediumGray (Medium Gray)	The fill style is medium gray. [*Example*:

Enumeration Value	Description
	 *end example*]
none (None)	The fill style is none (no fill). When foreground and/or background colors are specified, a pattern of 'none' overrides and means the cell has no fill. [*Example*: *end example*]
solid (Solid)	The fill style is solid. When solid is specified, the foreground color (fgColor) is the only color rendered, even when a background color (bgColor) is also specified. [*Example*: *end example*]

Referenced By
patternFill@patternType (§3.8.32)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PatternType">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="solid"/>
      <enumeration value="mediumGray"/>
      <enumeration value="darkGray"/>
      <enumeration value="lightGray"/>
      <enumeration value="darkHorizontal"/>
      <enumeration value="darkVertical"/>
      <enumeration value="darkDown"/>
      <enumeration value="darkUp"/>
      <enumeration value="darkGrid"/>
      <enumeration value="darkTrellis"/>
      <enumeration value="lightHorizontal"/>
      <enumeration value="lightVertical"/>
      <enumeration value="lightDown"/>
      <enumeration value="lightUp"/>
      <enumeration value="lightGrid"/>
      <enumeration value="lightTrellis"/>
      <enumeration value="gray125"/>
      <enumeration value="gray0625"/>
   </restriction>
</simpleType>
```

3.18.58 ST_PhoneticAlignment (Phonetic Alignment Types)

Phonetic alignment settings. These specify how to align the phonetic text, which represent the sounds, above the base text or base word.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
center (Center Alignment)	Center the phonetic characters over the base word, per word.
distributed (Distributed)	Each phonetic character is distributed above each base word character, per word.
left (Left Alignment)	Each phonetic character is left justified with respect to the base text., per word.
noControl (No Control)	Each phonetic character is left justified without respect to the base text (so it is not per word).

Referenced By
phoneticPr@alignment (§3.4.3)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PhoneticAlignment">
   <restriction base="xsd:string">
      <enumeration value="noControl"/>
      <enumeration value="left"/>
      <enumeration value="center"/>
      <enumeration value="distributed"/>
   </restriction>
</simpleType>
```

3.18.59 ST_PhoneticType (Phonetic Type)

Represents the different type of East Asian character sets that shall be used for displaying phonetic hints.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
fullwidthKatakana (Full-Width Katakana)	Full-width Katakana is used
halfwidthKatakana (Half-Width Katakana)	Half-width Katakana is used, this is the same Katakana character set, just half as wide so it takes up less space.
Hiragana (Hiragana)	Hiragana is used
noConversion (No Conversion)	Any type of characters are allowed. In this case the spreadsheet application shall leave the text as entered.

Referenced By
phoneticPr@type (§3.4.3)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PhoneticType">
   <restriction base="xsd:string">
      <enumeration value="halfwidthKatakana"/>
      <enumeration value="fullwidthKatakana"/>
      <enumeration value="Hiragana"/>
      <enumeration value="noConversion"/>
   </restriction>
</simpleType>
```

3.18.60 ST_PivotAreaType (Rule Type)

Indicates the type of rule being used to describe an area or aspect of the PivotTable.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
all (All)	Refers to the whole PivotTable.
button (Field Button)	Refers to a field button.
data (Data)	Refers to something in the data area.
none (None)	Refers to no Pivot area.
normal (Normal)	Refers to a header or item.
origin (Origin)	Refers to the blank cells at the top-left of the PivotTable (top-right for RTL sheets).
topRight (Top Right)	Refers to the blank cells at the top-right of the PivotTable (top-left for RTL sheets).

Referenced By
pivotArea@type (§3.3.1.66)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PivotAreaType">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="normal"/>
      <enumeration value="data"/>
      <enumeration value="all"/>
      <enumeration value="origin"/>
      <enumeration value="button"/>
      <enumeration value="topRight"/>
   </restriction>
</simpleType>
```

3.18.61 ST_PivotFilterType (Pivot Filter Types)

This simple type defines filters that can be applied to PivotTables.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
captionBeginsWith (Caption Begins With)	Indicates the "begins with" filter for field captions.
captionBetween (Caption Is Between)	Indicates the "is between" filter for field captions.
captionContains (Caption Contains)	Indicates the "contains" filter for field captions.
captionEndsWith (Caption Ends With)	Indicates the "ends with" filter for field captions.
captionEqual (Caption Equals)	Indicates the "equal" filter for field captions.

Enumeration Value	Description
captionGreaterThan (Caption Is Greater Than)	Indicates the "is greater than" filter for field captions.
captionGreaterThanOrEqual (Caption Is Greater Than Or Equal To)	Indicates the "is greater than or equal to" filter for field captions.
captionLessThan (Caption Is Less Than)	Indicates the "is less than" filter for field captions.
captionLessThanOrEqual (Caption Is Less Than Or Equal To)	Indicates the "is less than or equal to" filter for field captions.
captionNotBeginsWith (Caption Does Not Begin With)	Indicates the "does not begin with" filter for field captions.
captionNotBetween (Caption Is Not Between)	Indicates the "is not between" filter for field captions.
captionNotContains (Caption Does Not Contain)	Indicates the "does not contain" filter for field captions.
captionNotEndsWith (Caption Does Not End With)	Indicates the "does not end with" filter for field captions.
captionNotEqual (Caption Not Equal)	Indicates the "not equal" filter for field captions.
count (Count)	Indicates the "count" filter.
dateBetween (Date Between)	Indicates the "between" filter for date values.
dateEqual (Date Equals)	Indicates the "equals" filter for date values.
dateNewerThan (Date Newer Than)	Indicates the "newer than" filter for date values.
dateNewerThanOrEqual (Date Newer Than or Equal To)	Indicates the "newer than or equal to" filter for date values.
dateNotBetween (Date Not Between)	Indicates the "not between" filter for date values.
dateNotEqual (Date Does Not Equal)	Indicates the "does not equal" filter for date values.
dateOlderThan (Date Older Than)	Indicates the "older than" filter for date values.
dateOlderThanOrEqual (Date Older Than Or Equal)	Indicates the "older than or equal to" filter for date values.
lastMonth (Last Month)	Indicates the "last month" filter for date values.
lastQuarter (Last Quarter)	Indicates the "last quarter" filter for date values.
lastWeek (Last Week)	Indicates the "last week" filter for date values.
lastYear (Last Year)	Indicates the "last year" filter for date values.
M1 (January)	Indicates the "January" filter for date values.
M10 (Dates in October)	Indicates the "October" filter for date values.
M11 (Dates in November)	Indicates the "November" filter for date values.
M12 (Dates in December)	Indicates the "December" filter for date values.
M2 (Dates in February)	Indicates the "February" filter for date values.
M3 (Dates in March)	Indicates the "March" filter for date values.
M4 (Dates in April)	Indicates the "April" filter for date values.

Enumeration Value	Description
M5 (Dates in May)	Indicates the "May" filter for date values.
M6 (Dates in June)	Indicates the "June" filter for date values.
M7 (Dates in July)	Indicates the "July" filter for date values.
M8 (Dates in August)	Indicates the "August" filter for date values.
M9 (Dates in September)	Indicates the "September" filter for date values.
nextMonth (Next Month)	Indicates the "next month" filter for date values.
nextQuarter (Next Quarter)	Indicates the "next quarter" for date values.
nextWeek (Next Week)	Indicates the "next week" for date values.
nextYear (Next Year)	Indicates the "next year" filter for date values.
percent (Percent)	Indicates the "percent" filter for numeric values.
Q1 (First Quarter)	Indicates the "first quarter" filter for date values.
Q2 (Second Quarter)	Indicates the "second quarter" filter for date values.
Q3 (Third Quarter)	Indicates the "third quarter" filter for date values.
Q4 (Fourth Quarter)	Indicates the "fourth quarter" filter for date values.
sum (Sum)	Indicates the "sum" filter for numeric values.
thisMonth (This Month)	Indicates the "this month" filter for date values.
thisQuarter (This Quarter)	Indicates the "this quarter" filter for date values.
thisWeek (This Week)	Indicates the "this week" filter for date values.
thisYear (This Year)	Indicate the "this year" filter for date values.
today (Today)	Indicates the "today" filter for date values.
tomorrow (Tomorrow)	Indicates the "tomorrow" filter for date values.
unknown (Unknown)	Indicates the PivotTable filter is unknown to the application.
valueBetween (Value Between)	Indicates the "Value between" filter for text and numeric values.
valueEqual (Value Equal)	Indicates the "value equal" filter for text and numeric values.
valueGreaterThan (Value Greater Than)	Indicates the "value greater than" filter for text and numeric values.
valueGreaterThanOrEqual (Value Greater Than Or Equal To)	Indicates the "value greater than or equal to" filter for text and numeric values.
valueLessThan (Value Less Than)	Indicates the "value less than" filter for text and numeric values.
valueLessThanOrEqual (Value Less Than Or Equal To)	Indicates the "value less than or equal to" filter for text and numeric values
valueNotBetween (Value Not Between)	Indicates the "value not between" filter for text and

Enumeration Value	Description
	numeric values.
valueNotEqual (Value Not Equal)	Indicates the "value not equal" filter for text and numeric values.
yearToDate (Year-To-Date)	Indicates the "year-to-date" filter for date values.
yesterday (Yesterday)	Indicates the "yesterday" filter for date values.

Referenced By
filter@type (§3.10.1.33)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PivotFilterType">
   <restriction base="xsd:string">
      <enumeration value="unknown"/>
      <enumeration value="count"/>
      <enumeration value="percent"/>
      <enumeration value="sum"/>
      <enumeration value="captionEqual"/>
      <enumeration value="captionNotEqual"/>
      <enumeration value="captionBeginsWith"/>
      <enumeration value="captionNotBeginsWith"/>
      <enumeration value="captionEndsWith"/>
      <enumeration value="captionNotEndsWith"/>
      <enumeration value="captionContains"/>
      <enumeration value="captionNotContains"/>
      <enumeration value="captionGreaterThan"/>
      <enumeration value="captionGreaterThanOrEqual"/>
      <enumeration value="captionLessThan"/>
      <enumeration value="captionLessThanOrEqual"/>
      <enumeration value="captionBetween"/>
      <enumeration value="captionNotBetween"/>
      <enumeration value="valueEqual"/>
      <enumeration value="valueNotEqual"/>
      <enumeration value="valueGreaterThan"/>
      <enumeration value="valueGreaterThanOrEqual"/>
      <enumeration value="valueLessThan"/>
      <enumeration value="valueLessThanOrEqual"/>
      <enumeration value="valueBetween"/>
      <enumeration value="valueNotBetween"/>
      <enumeration value="dateEqual"/>
      <enumeration value="dateNotEqual"/>
      <enumeration value="dateOlderThan"/>
      <enumeration value="dateOlderThanOrEqual"/>
      <enumeration value="dateNewerThan"/>
      <enumeration value="dateNewerThanOrEqual"/>
      <enumeration value="dateBetween"/>
      <enumeration value="dateNotBetween"/>
      <enumeration value="tomorrow"/>
      <enumeration value="today"/>
      <enumeration value="yesterday"/>
      <enumeration value="nextWeek"/>
      <enumeration value="thisWeek"/>
      <enumeration value="lastWeek"/>
      <enumeration value="nextMonth"/>
      <enumeration value="thisMonth"/>
      <enumeration value="lastMonth"/>
      <enumeration value="nextQuarter"/>
      <enumeration value="thisQuarter"/>
      <enumeration value="lastQuarter"/>
      <enumeration value="nextYear"/>
      <enumeration value="thisYear"/>
      <enumeration value="lastYear"/>
      <enumeration value="yearToDate"/>
```

```
        <enumeration value="Q1"/>
        <enumeration value="Q2"/>
        <enumeration value="Q3"/>
        <enumeration value="Q4"/>
        <enumeration value="M1"/>
        <enumeration value="M2"/>
        <enumeration value="M3"/>
        <enumeration value="M4"/>
        <enumeration value="M5"/>
        <enumeration value="M6"/>
        <enumeration value="M7"/>
        <enumeration value="M8"/>
        <enumeration value="M9"/>
        <enumeration value="M10"/>
        <enumeration value="M11"/>
        <enumeration value="M12"/>
    </restriction>
</simpleType>
```

3.18.62 ST_PrintError (Print Errors)

This enumeration specifies how to display cells with errors when printing the worksheet.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
blank (Show Cell Errors As Blank)	Display cell errors as blank.
dash (Dash Cell Errors)	Display cell errors as dashes.
displayed (Display Cell Errors)	Display cell errors as displayed on screen.
NA (NA)	Display cell errors as #N/A.

Referenced By
pageSetup@errors (§3.3.1.61)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PrintError">
    <restriction base="xsd:string">
        <enumeration value="displayed"/>
        <enumeration value="blank"/>
        <enumeration value="dash"/>
        <enumeration value="NA"/>
    </restriction>
</simpleType>
```

3.18.63 ST_Qualifier (Qualifier)

Qualifier to use to denote string data types in when text is imported from an external file.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
doubleQuote (Double Quote)	Quotation mark -- double quote (").
none (No Text Qualifier)	No text string qualifier used.
singleQuote (Single Quote)	Apostrophe mark -- single quote (').

Referenced By
textPr@qualifier (§3.13.12)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Qualifier">
   <restriction base="xsd:string">
      <enumeration value="doubleQuote"/>
      <enumeration value="singleQuote"/>
      <enumeration value="none"/>
   </restriction>
</simpleType>
```

3.18.64 ST_Ref (Cell References)

This simple type defines a reference to a range of cells within a sheet in the workbook. A reference identifies a cell or a range of cells on a worksheet and tells the application where to look for the values or data you want to use in a formula. With references, you can use data contained in different parts of a worksheet in one formula or use the value from one cell in several formulas. You can also refer to cells on other sheets in the same workbook, and to other workbooks. References to cells in other workbooks are called links.

SpreadsheetML defines two reference styles defined in the ST_RefMode simple type.

This simple type's contents are a restriction of the XML Schema string datatype.

Referenced By
autoFilter@ref (§3.3.1.1); comment@ref (§3.7.3); dataRef@ref (§3.3.1.28); dimension@ref (§3.3.1.33); hyperlink@ref (§3.3.1.44); location@ref (§3.10.1.49); mergeCell@ref (§3.3.1.53); oleSize@ref (§3.2.16); pivotArea@offset (§3.3.1.66); raf@ref (§3.11.1.8); rangeSet@ref (§3.10.1.79); rm@destination (§3.11.1.19); rm@source (§3.11.1.19); rqt@ref (§3.11.1.20); rrc@ref (§3.11.1.21); sheetPr@syncRef (§3.3.1.79); sortCondition@ref (§3.3.1.88); sortState@ref (§3.3.1.89); ST_Ref (§3.18.64); table@ref (§3.5.1.2); webPublishItem@sourceRef (§3.3.1.94); worksheetSource@ref (§3.10.1.95)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Ref">
   <restriction base="xsd:string"/>
</simpleType>
```

3.18.65 ST_RefA (Single Cell Reference)

This simple type specifies a single cell reference that may be absolute.

This simple type's contents are a restriction of the XML Schema string datatype.

Referenced By
undo@dr (§3.11.1.25)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_RefA">
   <restriction base="xsd:string"/>
</simpleType>
```

3.18.66 ST_RefMode (Reference Mode)

This simple type defines the supported reference styles or modes for a workbook in SpreadsheetML.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
A1 (A1 Mode)	Indicates that the workbook uses A1 reference style. This is the default for SpreadsheetML. A1 reference style refers to columns with letters and refers to rows with numbers. For example, A1 refers to the cell at the intersection of column A and row 1.
R1C1 (R1C1 Reference Mode)	Indicates that the workbook uses the R1C1 reference style. R1C1 reference style refers to both the rows and the columns on the worksheet with numbers. The location of a cell is indicated with an "R" followed by a row number and a "C" followed by a column number. For example, R1C1 refers to the cell at the intersection of row R1 and column C1.

Referenced By
calcPr@refMode (§3.2.2)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_RefMode">
   <restriction base="xsd:string">
      <enumeration value="A1"/>
      <enumeration value="R1C1"/>
   </restriction>
</simpleType>
```

3.18.67 ST_RevisionAction (Revision Action Types)

Identifies what kind of action the user performed. Applies to Comment and Custom View revision record.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
add (Add)	Add action.
delete (Delete)	Delete action.

Referenced By
rcmt@action (§3.11.1.11); rcv@action (§3.11.1.12)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_RevisionAction">
   <restriction base="xsd:string">
      <enumeration value="add"/>
      <enumeration value="delete"/>
   </restriction>
</simpleType>
```

3.18.68 ST_rwColActionType (Row Column Action Type)

Identifies what kind of an action was applied to a row or column.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
deleteCol (Delete Column)	Column delete revision.
deleteRow (Delete Row)	Row delete revision.
insertCol (Column Insert)	Column insert revision.
insertRow (Insert Row)	Row insert revision.

Referenced By
rrc@action (§3.11.1.21)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_rwColActionType">
   <restriction base="xsd:string">
      <enumeration value="insertRow"/>
      <enumeration value="deleteRow"/>
      <enumeration value="insertCol"/>
      <enumeration value="deleteCol"/>
   </restriction>
</simpleType>
```

3.18.69 ST_Scope (Conditional Formatting Scope)

This simple type defines the scope of conditional formatting applied in the PivotTable.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
data (Data Fields)	Indicates that conditional formatting is applied to the selected data fields.
field (Field Intersections)	Indicates that conditional formatting is applied to the selected PivotTable field intersections.
selection (Selection)	Indicates that conditional formatting is applied to the selected cells.

Referenced By
conditionalFormat@scope (§3.10.1.18)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Scope">
   <restriction base="xsd:string">
      <enumeration value="selection"/>
      <enumeration value="data"/>
      <enumeration value="field"/>
   </restriction>
</simpleType>
```

3.18.70 ST_SheetState (Sheet Visibility Types)

This simple type defines the possible states for sheet visibility.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
hidden (Hidden)	Indicates the book window is hidden, but can be shown by the user via the user interface.
veryHidden (Very Hidden)	Indicates the sheet is hidden and cannot be shown in the user interface (UI). This state is only available programmatically.
visible (Visible)	Indicates the sheet is visible.

Referenced By
customSheetView@state (§3.3.1.22); customSheetView@state (§3.3.1.23); sheet@state (§3.2.19)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SheetState">
   <restriction base="xsd:string">
      <enumeration value="visible"/>
      <enumeration value="hidden"/>
      <enumeration value="veryHidden"/>
   </restriction>
</simpleType>
```

3.18.71 ST_SheetViewType (Sheet View Type)

Defines the view setting of the sheet.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
normal (Normal View)	Normal view
pageBreakPreview (Page Break Preview)	Page break preview
pageLayout (Page Layout View)	Page Layout View

Referenced By
customSheetView@view (§3.3.1.23); sheetView@view (§3.3.1.83)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SheetViewType">
   <restriction base="xsd:string">
      <enumeration value="normal"/>
      <enumeration value="pageBreakPreview"/>
      <enumeration value="pageLayout"/>
   </restriction>
</simpleType>
```

3.18.72 ST_ShowDataAs (Show Data As)

This simple type defines the data formats for a field in the PivotTable.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
difference (Difference)	Indicates the field is shown as the "difference from" a value.
index (Index)	Indicates the field is shown as the "index.
normal (Normal Data Type)	Indicates that the field is shown as its normal data type.
percent (Percentage Of)	Indicates the field is show as the "percentage of
percentDiff (Percentage Difference)	Indicates the field is shown as the "percentage difference from" a value.
percentOfCol (Percent of Column)	Indicates the field is shown as the percentage of column.
percentOfRow (Percentage of Row)	Indicates the field is shown as the percentage of row
percentOfTotal (Percentage of Total)	Indicates the field is shown as percentage of total.
runTotal (Running Total)	Indicates the field is shown as running total in the table.

Referenced By
dataField@showDataAs (§3.10.1.22)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ShowDataAs">
   <restriction base="xsd:string">
      <enumeration value="normal"/>
      <enumeration value="difference"/>
      <enumeration value="percent"/>
      <enumeration value="percentDiff"/>
      <enumeration value="runTotal"/>
      <enumeration value="percentOfRow"/>
      <enumeration value="percentOfCol"/>
      <enumeration value="percentOfTotal"/>
      <enumeration value="index"/>
   </restriction>
</simpleType>
```

3.18.73 ST_SmartTagShow (Smart Tag Display Types)

This simple type defines options for displaying smart tags in the user interface.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
all (All)	Indicates that smart tags are enabled and shown in the user interface.
noIndicator (No Smart Tag Indicator)	Indicates that the smart tags are enabled but the indicator not be shown in the user interface.
none (None)	Indicates that smart tags are disabled and not displayed in the user interface.

Referenced By
smartTagPr@show (§3.2.21)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SmartTagShow">
   <restriction base="xsd:string">
      <enumeration value="all"/>
      <enumeration value="none"/>
      <enumeration value="noIndicator"/>
   </restriction>
</simpleType>
```

3.18.74 ST_SortBy (Sort By)

Specifies what to sort by. In many cases a range of cells are sorted by their values. However, cells can also be sorted by their background color, font color, and type of icon in the cell.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
cellColor (Sort by Cell Color)	Sort by cell color
fontColor (Sort by Font Color)	Sort by font color
icon (Sort by Icon)	Sort by icon
value (Value)	Sort by value

Referenced By
sortCondition@sortBy (§3.3.1.88)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SortBy">
   <restriction base="xsd:string">
      <enumeration value="value"/>
      <enumeration value="cellColor"/>
      <enumeration value="fontColor"/>
      <enumeration value="icon"/>
   </restriction>
</simpleType>
```

3.18.75 ST_SortMethod (Sort Method)

Sort method. Chinese Simplified, Chinese Traditional, and Japanese support alternate sort methods (multiple sort options are available). All other languages support only 1 sort option. In that case, the value `pinYin` is used.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
none (None)	Not specified, use default sort method.
pinYin (PinYin Sort)	Default sort method. This is the only sort option for most languages. For Chinese Simplified, Chinese Traditional, and Japanese, `pinYin` means sort by phonetic value.
stroke (Sort by Stroke)	Sort by stroke count of the characters. Only applies to Chinese Simplified, Chinese Traditional, and Japanese.

Referenced By
sortState@sortMethod (§3.3.1.89)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SortMethod">
   <restriction base="xsd:string">
      <enumeration value="stroke"/>
      <enumeration value="pinYin"/>
      <enumeration value="none"/>
   </restriction>
</simpleType>
```

3.18.76 ST_SortType (Set Sort Order)

This simple type defines the possible sort order for the PivotTable.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
ascending (Ascending)	Indicates that the PivotTable data is sorted in ascending order.
ascendingAlpha (Ascending Alpha)	Indicates that the PivotTable data is sorted in alphabetic order with ascending values.
ascendingNatural (Ascending Natural)	Indicates that the PivotTable data is sorted in natural order with ascending.
descending (Descending)	Indicates that the PivotTable data is sorted in descending.
descendingAlpha (Alphabetic Order Descending)	Indicates that the PivotTable data is sorted in alphabetic order with descending values.
descendingNatural (Natural Order Descending)	Indicates that the PivotTable data is sorted in natural order with descending values.
none (None)	Indicates that the PivotTable data is not sorted.

Referenced By
set@sortType (§3.10.1.88)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SortType">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="ascending"/>
      <enumeration value="descending"/>
      <enumeration value="ascendingAlpha"/>
      <enumeration value="descendingAlpha"/>
      <enumeration value="ascendingNatural"/>
      <enumeration value="descendingNatural"/>
   </restriction>
</simpleType>
```

3.18.77 ST_SourceType (PivotCache Type)

This simple type defines the cache types for PivotTables.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
consolidation (Consolidation Ranges)	Indicates that the cache contains data that consolidates ranges.
external (External)	Indicates that the cache contains data from an external data source.
scenario (Scenario Summary Report)	Indicates that the cache contains a scenario summary report
worksheet (Worksheet)	Indicates that the cache contains worksheet data.

Referenced By
cacheSource@type (§3.10.1.7)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SourceType">
   <restriction base="xsd:string">
      <enumeration value="worksheet"/>
      <enumeration value="external"/>
      <enumeration value="consolidation"/>
      <enumeration value="scenario"/>
   </restriction>
</simpleType>
```

3.18.78 ST_Sqref (Reference Sequence)

A sequence of cell references, space delimited.

This simple type allows a list of items of the ST_Ref simple type (§3.18.64).

Referenced By
conditionalFormatting@sqref (§3.3.1.17); dataValidation@sqref (§3.3.1.30); ignoredError@sqref (§3.3.1.47); protectedRange@sqref (§3.3.1.69); rfmt@sqref (§3.11.1.17); scenarios@sqref (§3.3.1.74); selection@sqref (§3.3.1.75)

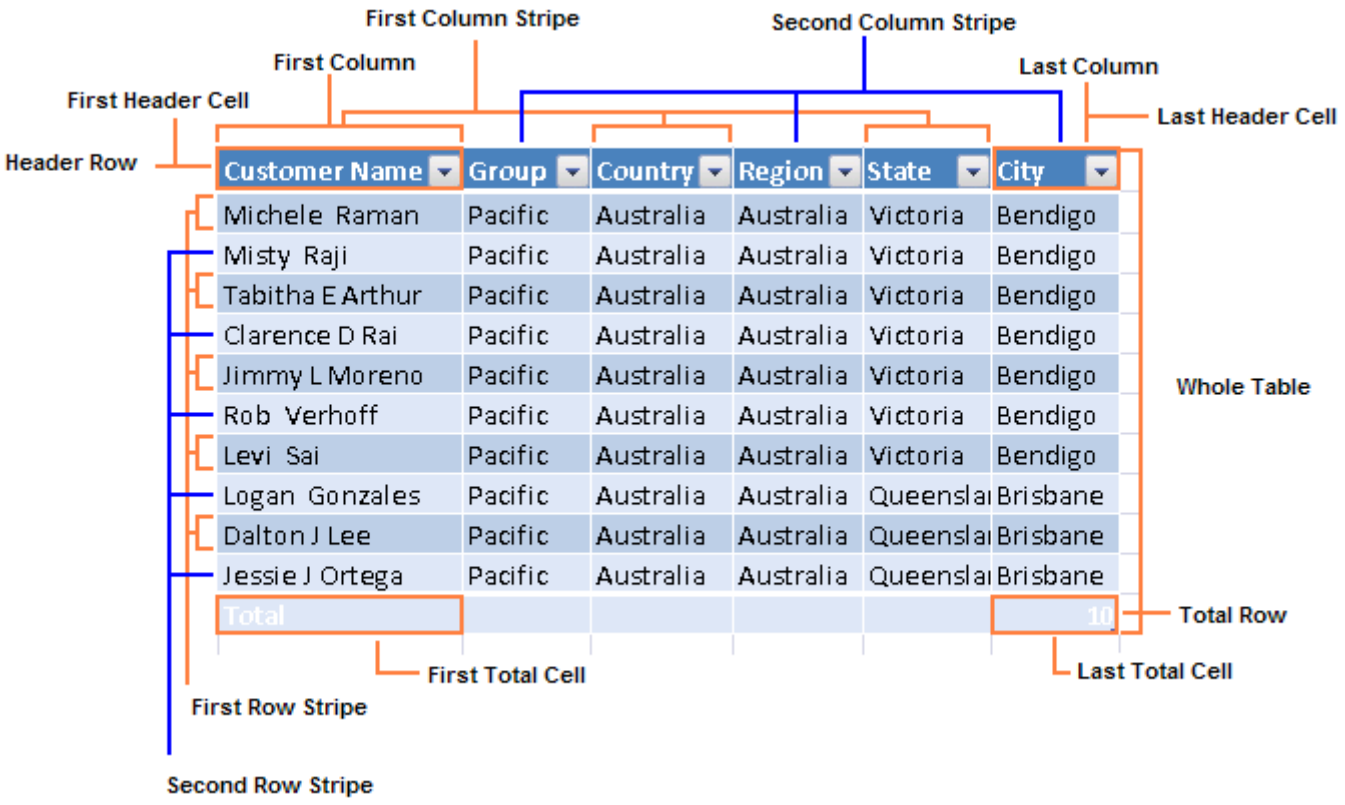
The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Sqref">
   <list itemType="ST_Ref"/>
</simpleType>
```

3.18.79 ST_TableStyleType (Table Style Type)

Enumeration of the different structured regions of a Table or PivotTable which can be formatted. Specifies which region is being formatted by this table style element.

Table Regions



PivotTable Regions

Blank Row

Only applies when "Insert blank row after each item" is ON.

Country	(All)	
Sum of Sales Amount	Column Labels	
	2001	
	3	
	July	July Total
Row Labels	No Discount	
New South Wales	75625.4664	75625.4664
Coffs Harbour	699.0982	699.0982
Bikes	699.0982	699.0982
Road Bikes	699.0982	699.0982
Road-150 Red, 48		
Road-150 Red, 52		
Road-150 Red, 62		
Road-650 Red, 44	699.0982	699.0982
Darlinghurst	3578.27	3578.27
Bikes	3578.27	3578.27
Road Bikes	3578.27	3578.27
Road-150 Red, 48		
Road-150 Red, 56	3578.27	3578.27
Road-150 Red, 62		
Goulburn	18412.1682	18412.1682
Bikes	18412.1682	18412.1682
Mountain Bikes	3399.99	3399.99
Mountain-100 Silver, 44	3399.99	3399.99
Road Bikes	15012.1782	15012.1782

Whole Table

Country	(All)	
Sum of Sales Amount	Column Labels	
	2001	
	3	
	July	July Total
Row Labels	No Discount	
New South Wales	75625.4664	75625.4664
Coffs Harbour	699.0982	699.0982
Bikes	699.0982	699.0982
Road Bikes	699.0982	699.0982
Road-150 Red, 48		
Road-150 Red, 52		
Road-150 Red, 62		
Road-650 Red, 44	699.0982	699.0982
Darlinghurst	3578.27	3578.27
Bikes	3578.27	3578.27
Road Bikes	3578.27	3578.27
Road-150 Red, 48		
Road-150 Red, 56	3578.27	3578.27
Road-150 Red, 62		

Page Field Labels

Country	(All)	
Sum of Sales Amount	Column Labels	
	2001	
	3	
	July	July Total
Row Labels	No Discount	
New South Wales	75625.4664	75625.4664
Coffs Harbour	699.0982	699.0982
Bikes	699.0982	699.0982
Road Bikes	699.0982	699.0982

Page Field Values

Country	(All)	
Sum of Sales Amount	Column Labels	
	2001	
	3	
	July	July Total
Row Labels	No Discount	
New South Wales	75625.4664	75625.4664
Coffs Harbour	699.0982	699.0982
Bikes	699.0982	699.0982
Road Bikes	699.0982	699.0982

First Column Stripe

Country	(All)				
Sum of Sales Amount	Column Labels				
	2001				
	3				
	July	July Total	August	August Total	September
Row Labels	No Discount		No Discount		No Discount
New South Wales	75625.4664	75625.4664	119823.881	119823.881	83698.4064
Coffs Harbour	699.0982	699.0982	7156.54	7156.54	7156.54
Bikes	699.0982	699.0982	7156.54	7156.54	7156.54
Road Bikes	699.0982	699.0982	7156.54	7156.54	7156.54
Road-150 Red, 48					3578.27
Road-150 Red, 52			3578.27	3578.27	3578.27
Road-150 Red, 62			3578.27	3578.27	

Second Column Stripe

Country	(All)					
Sum of Sales Amount	Column Labels					
	2001					
	3					
	July	July Total	August	August Total	September	
Row Labels	No Discount		No Discount		No Discount	
New South Wales	75625.4664	75625.4664	119823.881	119823.881	83698.4064	
Coffs Harbour	699.0982	699.0982	7156.54	7156.54	7156.54	
Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	
Road Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	
Road-150 Red, 48					3578.27	
Road-150 Red, 52			3578.27	3578.27	3578.27	
Road-150 Red, 62			3578.27	3578.27		
Road-650 Red, 44	699.0982	699.0982				
Darlinghurst	3578.27	3578.27	3578.27	3578.27	7156.54	
Bikes	3578.27	3578.27	3578.27	3578.27	7156.54	
Road Bikes	3578.27	3578.27	3578.27	3578.27	7156.54	
Road-150 Red, 48			3578.27	3578.27		
Road-150 Red, 56	3578.27	3578.27			3578.27	
Road-150 Red, 62					3578.27	

First Row Stripe

Country	(All)			
Sum of Sales Amount	Column Labels			
	2001			
	3			
	July	July Total	August	August Total
Row Labels	No Discount		No Discount	
New South Wales	75625.4664	75625.4664	119823.881	119823.881
Coffs Harbour	699.0982	699.0982	7156.54	7156.54
Bikes	699.0982	699.0982	7156.54	7156.54
Road Bikes	699.0982	699.0982	7156.54	7156.54
Road-150 Red, 48				
Road-150 Red, 52			3578.27	3578.27
Road-150 Red, 62			3578.27	3578.27
Road-650 Red, 44	699.0982	699.0982		
Darlinghurst	3578.27	3578.27	3578.27	3578.27
Bikes	3578.27	3578.27	3578.27	3578.27
Road Bikes	3578.27	3578.27	3578.27	3578.27
Road-150 Red, 48			3578.27	3578.27
Road-150 Red, 56	3578.27	3578.27		
Road-150 Red, 62				

Second Row Stripe

Country	(All)			
Sum of Sales Amount	Column Labels			
	2001			
	3			
	July	July Total	August	August Total
Row Labels	No Discount		No Discount	
New South Wales	75625.4664	75625.4664	119823.881	119823.881
Coffs Harbour	699.0982	699.0982	7156.54	7156.54
Bikes	699.0982	699.0982	7156.54	7156.54
Road Bikes	699.0982	699.0982	7156.54	7156.54
Road-150 Red, 48				
Road-150 Red, 52			3578.27	3578.27
Road-150 Red, 62			3578.27	3578.27
Road-650 Red, 44	699.0982	699.0982		
Darlinghurst	3578.27	3578.27	3578.27	3578.27
Bikes	3578.27	3578.27	3578.27	3578.27
Road Bikes	3578.27	3578.27	3578.27	3578.27
Road-150 Red, 48			3578.27	3578.27
Road-150 Red, 56	3578.27	3578.27		
Road-150 Red, 62				

First Column

Country	(All)		
Sum of Sales Amount	Column Labels		
	2001		
	3		
	July	July Total	August
Row Labels	No Discount		No Discount
New South Wales	75625.4664	75625.4664	119823.881
Coffs Harbour	699.0982	699.0982	7156.54
Bikes	699.0982	699.0982	7156.54
Road Bikes	699.0982	699.0982	7156.54
Road-150 Red, 48			
Road-150 Red, 52			3578.27
Road-150 Red, 62			3578.27
Road-650 Red, 44	699.0982	699.0982	
Darlinghurst	3578.27	3578.27	3578.27
Bikes	3578.27	3578.27	3578.27
Road Bikes	3578.27	3578.27	3578.27
Road-150 Red, 48			3578.27
Road-150 Red, 56	3578.27	3578.27	
Road-150 Red, 62			

Header Row

Country	(All)		
Sum of Sales Amount	Column Labels		
	2001		
	3		
	July	July Total	August
Row Labels	No Discount		No Discount
New South Wales	75625.4664	75625.4664	119823.881
Coffs Harbour	699.0982	699.0982	7156.54
Bikes	699.0982	699.0982	7156.54
Road Bikes	699.0982	699.0982	7156.54
Road-150 Red, 48			
Road-150 Red, 52			3578.27
Road-150 Red, 62			3578.27
Road-650 Red, 44	699.0982	699.0982	

First Header Cell

Country	(All)		
Sum of Sales Amount	Column Labels		
	2001		
	3		
	July	July Total	August
Row Labels	No Discount		No Discount
New South Wales	75625.4664	75625.4664	119823.881
Coffs Harbour	699.0982	699.0982	7156.54
Bikes	699.0982	699.0982	7156.54
Road Bikes	699.0982	699.0982	7156.54
Road-150 Red, 48			
Road-150 Red, 52			3578.27
Road-150 Red, 62			3578.27
Road-650 Red, 44	699.0982	699.0982	

Subtotal Column 1

Country	(All)							
Sum of Sales Amount	Column Labels							
	2001							2001 Total
	3						3 Total	
	July	July Total	August	August Total	September	September Total		
Row Labels	No Discount		No Discount		No Discount			
New South Wales	75625.4664	75625.4664	119823.881	119823.881	83698.4064	83698.4064	279147.7538	279147.7538
Coffs Harbour	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Road Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Road-150 Red, 48					3578.27	3578.27	3578.27	3578.27
Road-150 Red, 52			3578.27	3578.27	3578.27	3578.27	7156.54	7156.54
Road-150 Red, 62			3578.27	3578.27			3578.27	3578.27
Road-650 Red, 44	699.0982	699.0982					699.0982	699.0982

Subtotal Column 2

Country	(All)							
Sum of Sales Amount	Column Labels							
	2001							2001 Total
	3						3 Total	
	July	July Total	August	August Total	September	September Total		
Row Labels	No Discount		No Discount		No Discount			
New South Wales	75625.4664	75625.4664	119823.881	119823.881	83698.4064	83698.4064	279147.7538	279147.7538
Coffs Harbour	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Road Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Road-150 Red, 48					3578.27	3578.27	3578.27	3578.27
Road-150 Red, 52			3578.27	3578.27	3578.27	3578.27	7156.54	7156.54
Road-150 Red, 62			3578.27	3578.27			3578.27	3578.27
Road-650 Red, 44	699.0982	699.0982					699.0982	699.0982

Subtotal Column 3

Country	(All)							
Sum of Sales Amount	Column Labels							
	2001							2001 Total
	3						3 Total	
	July	July Total	August	August Total	September	September Total		
Row Labels	No Discount		No Discount		No Discount			
New South Wales	75625.4664	75625.4664	119823.881	119823.881	83698.4064	83698.4064	279147.7538	279147.7538
Coffs Harbour	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Road Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Road-150 Red, 48					3578.27	3578.27	3578.27	3578.27
Road-150 Red, 52			3578.27	3578.27	3578.27	3578.27	7156.54	7156.54
Road-150 Red, 62			3578.27	3578.27			3578.27	3578.27
Road-650 Red, 44	699.0982	699.0982					699.0982	699.0982

Subtotal Row 1

Country	(All)				
Sum of Sales Amount	Column Labels				
	2001				
	3				
	July	July Total	August	August Total	September
Row Labels	No Discount		No Discount		No Discount
Bendigo					
Mountain Bikes					
Mountain-100 Black, 38	3374.99	3374.99			
Mountain-100 Silver, 42	3399.99	3399.99			
Mountain-100 Silver, 48	3399.99	3399.99			
Mountain Bikes Total	10174.97	10174.97			
Road Bikes					
Road-150 Red, 44	3578.27	3578.27			
Road-150 Red, 52	3578.27	3578.27			
Road-150 Red, 62	3578.27	3578.27			3578.27
Road Bikes Total	10734.81	10734.81			3578.27
Bendigo Total	20909.78	20909.78			3578.27
Brisbane					
Road Bikes					
Road-150 Red, 44	3578.27	3578.27	3578.27	3578.27	
Road-150 Red, 62			3578.27	3578.27	
Road Bikes Total	3578.27	3578.27	7156.54	7156.54	
Brisbane Total	3578.27	3578.27	7156.54	7156.54	

Subtotal Row 2

Country	(All)					
Sum of Sales Amount	Column Labels					
	2001					
	3					
	July	July Total	August	August Total	September	
Row Labels	No Discount		No Discount		No Discount	
Bendigo						
Mountain Bikes						
Mountain-100 Black, 38	3374.99	3374.99				
Mountain-100 Silver, 42	3399.99	3399.99				
Mountain-100 Silver, 48	3399.99	3399.99				
Mountain Bikes Total	10174.97	10174.97				
Road Bikes						
Road-150 Red, 44	3578.27	3578.27				
Road-150 Red, 52	3578.27	3578.27				
Road-150 Red, 62	3578.27	3578.27			3578.27	
Road Bikes Total	10734.81	10734.81			3578.27	
Bendigo Total	20909.78	20909.78			3578.27	
Brisbane						
Road Bikes						
Road-150 Red, 44	3578.27	3578.27	3578.27	3578.27		
Road-150 Red, 62			3578.27	3578.27		
Road Bikes Total	3578.27	3578.27	7156.54	7156.54		

Subtotal Row 3

Country	(All)					
Sum of Sales Amount	Column Labels					
	2001					
	3					
	July	July Total	August	August Total	September	
Row Labels	No Discount		No Discount		No Discount	
New South Wales						
Coffs Harbour						
Road Bikes						
Road-150 Red, 48						3578.27
Road-150 Red, 52			3578.27	3578.27		3578.27
Road-150 Red, 62			3578.27	3578.27		
Road-650 Red, 44	699.0982	699.0982				
Road Bikes Total	699.0982	699.0982	7156.54	7156.54		7156.54
Coffs Harbour Total	699.0982	699.0982	7156.54	7156.54		7156.54
Darlinghurst						
Road Bikes						
Road-150 Red, 48			3578.27	3578.27		
Road-150 Red, 56	3578.27	3578.27				3578.27
Road-150 Red, 62						3578.27
Road Bikes Total	3578.27	3578.27	3578.27	3578.27		7156.54
Darlinghurst Total	3578.27	3578.27	3578.27	3578.27		7156.54

Column Subheading 1

Country	(All)							
Sum of Sales Amount	Column Labels							
	2001							2001 Total
	3						3 Total	
	July	July Total	August	August Total	September	September Total		
Row Labels	No Discount		No Discount		No Discount			
New South Wales	75625.4664	75625.4664	119823.881	119823.881	83698.4064	83698.4064	279147.7538	279147.7538
Coffs Harbour	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Road Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782

Column Subheading 2

Country	(All)							
Sum of Sales Amount	Column Labels							
	2001							
	3							3 Total
	July	July Total	August	August Total	September	September Total		
Row Labels	No Discount		No Discount		No Discount			
New South Wales	75625.4664	75625.4664	119823.881	119823.881	83698.4064	83698.4064	279147.7538	
Coffs Harbour	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	
Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	
Road Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	

Column Subheading 3

Country	(All)							
Sum of Sales Amount	Column Labels							
	2001							2001 Total
	3						3 Total	
	July	July Total	August	August Total	September	September Total		
Row Labels	No Discount		No Discount		No Discount			
New South Wales	75625.4664	75625.4664	119823.881	119823.881	83698.4064	83698.4064	279147.7538	279147.7538
Coffs Harbour	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Road Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782

Row Subheading 1

Country	(All)				
Sum of Sales Amount	Column Labels				
	2001				
	3				
	July	July Total	August	August Total	September
Row Labels	No Discount		No Discount		No Discount
New South Wales					
Coffs Harbour					
Road Bikes					
Road-150 Red, 48					3578.27
Road-150 Red, 52			3578.27	3578.27	3578.27
Road-150 Red, 62			3578.27	3578.27	
Road-650 Red, 44	699.0982	699.0982			
Road Bikes Total	699.0982	699.0982	7156.54	7156.54	7156.54
Coffs Harbour Total	699.0982	699.0982	7156.54	7156.54	7156.54

Row Subheading 2

Country	(All)				
Sum of Sales Amount	Column Labels				
	2001				
	3				
	July	July Total	August	August Total	September
Row Labels	No Discount		No Discount		No Discount
New South Wales					
Coffs Harbour					
Road Bikes					
Road-150 Red, 48					3578.27
Road-150 Red, 52			3578.27	3578.27	3578.27
Road-150 Red, 62			3578.27	3578.27	
Road-650 Red, 44	699.0982	699.0982			
Road Bikes Total	699.0982	699.0982	7156.54	7156.54	7156.54
Coffs Harbour Total	699.0982	699.0982	7156.54	7156.54	7156.54
Darlinghurst					
Road Bikes					
Road-150 Red, 48			3578.27	3578.27	
Road-150 Red, 56	3578.27	3578.27			3578.27
Road-150 Red, 62					3578.27
Road Bikes Total	3578.27	3578.27	3578.27	3578.27	7156.54
Darlinghurst Total	3578.27	3578.27	3578.27	3578.27	7156.54
Goulburn					
Mountain Bikes					
Mountain-100 Silver, 44	3399.99	3399.99			

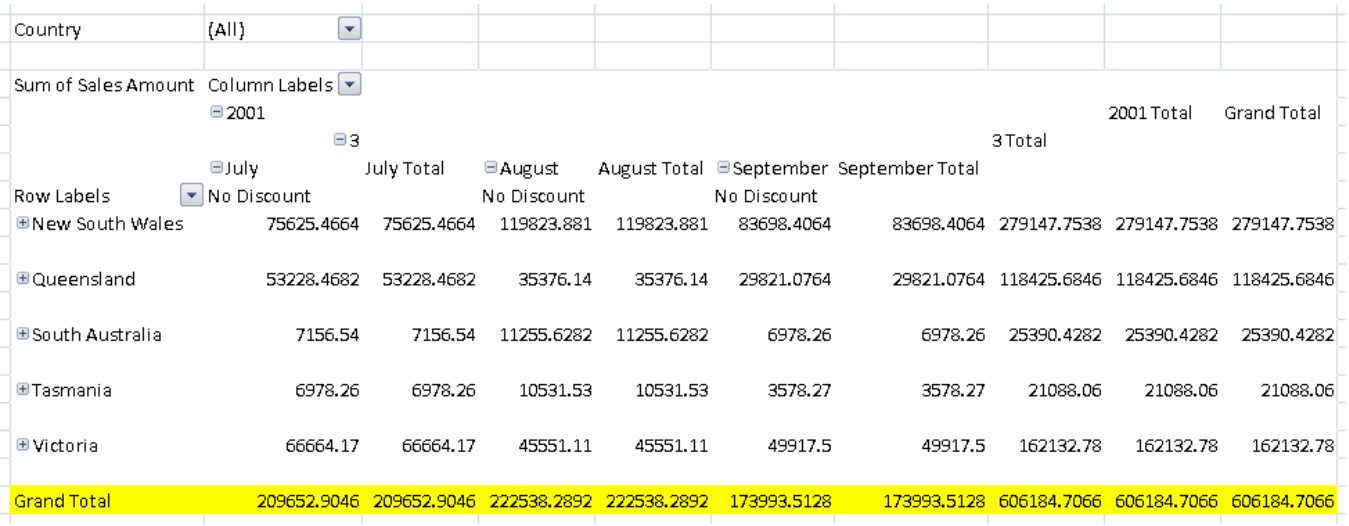
Row Subheading 3

Country	(All)				
Sum of Sales Amount	Column Labels				
	2001				
	3				
	July	July Total	August	August Total	September
Row Labels	No Discount		No Discount		No Discount
New South Wales					
Coffs Harbour					
Road Bikes					
Road-150 Red, 48					3578.27
Road-150 Red, 52			3578.27	3578.27	3578.27
Road-150 Red, 62			3578.27	3578.27	
Road-650 Red, 44	699.0982	699.0982			
Road Bikes Total	699.0982	699.0982	7156.54	7156.54	7156.54
Coffs Harbour Total	699.0982	699.0982	7156.54	7156.54	7156.54
Darlinghurst					
Road Bikes					
Road-150 Red, 48			3578.27	3578.27	
Road-150 Red, 56	3578.27	3578.27			3578.27
Road-150 Red, 62					3578.27
Road Bikes Total	3578.27	3578.27	3578.27	3578.27	7156.54
Darlinghurst Total	3578.27	3578.27	3578.27	3578.27	7156.54
Goulburn					
Mountain Bikes					
Mountain-100 Silver, 44	3399.99	3399.99			

Grand Total Column

Country	(All)								
Sum of Sales Amount	Column Labels								
	2001							2001 Total	Grand Total
	3						3 Total		
	July	July Total	August	August Total	September	September Total			
Row Labels	No Discount		No Discount		No Discount				
New South Wales	75625.4664	75625.4664	119823.881	119823.881	83698.4064	83698.4064	279147.7538	279147.7538	279147.7538
Coffs Harbour	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782	15012.1782
Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782	15012.1782
Road Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782	15012.1782
Road-150 Red, 48					3578.27	3578.27	3578.27	3578.27	3578.27
Road-150 Red, 52			3578.27	3578.27	3578.27	3578.27	7156.54	7156.54	7156.54
Road-150 Red, 62			3578.27	3578.27			3578.27	3578.27	3578.27
Road-650 Red, 44	699.0982	699.0982					699.0982	699.0982	699.0982

Grand Total Row



Country	(All)								
Sum of Sales Amount	Column Labels								
	2001							2001 Total	Grand Total
	3						3 Total		
	July	July Total	August	August Total	September	September Total			
Row Labels	No Discount		No Discount		No Discount				
New South Wales	75625.4664	75625.4664	119823.881	119823.881	83698.4064	83698.4064	279147.7538	279147.7538	279147.7538
Queensland	53228.4682	53228.4682	35376.14	35376.14	29821.0764	29821.0764	118425.6846	118425.6846	118425.6846
South Australia	7156.54	7156.54	11255.6282	11255.6282	6978.26	6978.26	25390.4282	25390.4282	25390.4282
Tasmania	6978.26	6978.26	10531.53	10531.53	3578.27	3578.27	21088.06	21088.06	21088.06
Victoria	66664.17	66664.17	45551.11	45551.11	49917.5	49917.5	162132.78	162132.78	162132.78
Grand Total	209652.9046	209652.9046	222538.2892	222538.2892	173993.5128	173993.5128	606184.7066	606184.7066	606184.7066

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
blankRow (Blank Row Style)	Table style element that applies to PivotTable's blank rows.
firstColumn (First Column Style)	Table style element that applies to table's first column.
firstColumnStripe (First Column Stripe Style)	Table style element that applies to table's first column stripes.
firstColumnSubheading (First Column Subheading Style)	Table style element that applies to PivotTable's first column subheading.
firstHeaderCell (First Header Row Style)	Table style element that applies to table's first header row cell.
firstRowStripe (First Row Stripe Style)	Table style element that applies to table's first row stripes.
firstRowSubheading (First Row Subheading Style)	Table style element that applies to PivotTable's first row subheading.
firstSubtotalColumn (First Subtotal Column Style)	Table style element that applies to PivotTable's first subtotal column.
firstSubtotalRow (First Subtotal Row Style)	Table style element that applies to pivot table's first subtotal row.
firstTotalCell (First Total Row Style)	Table style element that applies to table's first total

Enumeration Value	Description
	row cell.
headerRow (Header Row Style)	Table style element that applies to table's header row.
lastColumn (Last Column Style)	Table style element that applies to table's last column.
lastHeaderCell (Last Header Style)	Table style element that applies to table's last header row cell.
lastTotalCell (Last Total Row Style)	Table style element that applies to table's last total row cell.
pageFieldLabels (Page Field Labels Style)	Table style element that applies to pivot table's page field labels.
pageFieldValues (Page Field Values Style)	Table style element that applies to pivot table's page field values.
secondColumnStripe (Second Column Stipe Style)	Table style element that applies to table's second column stripes.
secondColumnSubheading (Second Column Subheading Style)	Table style element that applies to pivot table's second column subheading.
secondRowStripe (Second Row Stripe Style)	Table style element that applies to table's second row stripes.
secondRowSubheading (Second Row Subheading Style)	Table style element that applies to pivot table's second row subheading.
secondSubtotalColumn (Second Subtotal Column Style)	Table style element that applies to PivotTable's second subtotal column.
secondSubtotalRow (Second Subtotal Row Style)	Table style element that applies to PivotTable's second subtotal row.
thirdColumnSubheading (Third Column Subheading Style)	Table style element that applies to PivotTable's third column subheading.
thirdRowSubheading (Third Row Subheading Style)	Table style element that applies to PivotTable's third row subheading.
thirdSubtotalColumn (Third Subtotal Column Style)	Table style element that applies to pivot table's third subtotal column.
thirdSubtotalRow (Third Subtotal Row Style)	Table style element that applies to PivotTable's third subtotal row.
totalRow (Total Row Style)	Table style element that applies to table's total row.
wholeTable (Whole Table Style)	Table style element that applies to table's entire content.

Referenced By
tableStyleElement@type (§3.8.41)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TableStyleType">
   <restriction base="xsd:string">
      <enumeration value="wholeTable"/>
      <enumeration value="headerRow"/>
      <enumeration value="totalRow"/>
      <enumeration value="firstColumn"/>
      <enumeration value="lastColumn"/>
      <enumeration value="firstRowStripe"/>
      <enumeration value="secondRowStripe"/>
      <enumeration value="firstColumnStripe"/>
      <enumeration value="secondColumnStripe"/>
      <enumeration value="firstHeaderCell"/>
      <enumeration value="lastHeaderCell"/>
      <enumeration value="firstTotalCell"/>
      <enumeration value="lastTotalCell"/>
      <enumeration value="firstSubtotalColumn"/>
      <enumeration value="secondSubtotalColumn"/>
      <enumeration value="thirdSubtotalColumn"/>
      <enumeration value="firstSubtotalRow"/>
      <enumeration value="secondSubtotalRow"/>
      <enumeration value="thirdSubtotalRow"/>
      <enumeration value="blankRow"/>
      <enumeration value="firstColumnSubheading"/>
      <enumeration value="secondColumnSubheading"/>
      <enumeration value="thirdColumnSubheading"/>
      <enumeration value="firstRowSubheading"/>
      <enumeration value="secondRowSubheading"/>
      <enumeration value="thirdRowSubheading"/>
      <enumeration value="pageFieldLabels"/>
      <enumeration value="pageFieldValues"/>
   </restriction>
</simpleType>
```

3.18.80 ST_TableType (Table Type)

An enumeration that specifies what the table data is based on.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
queryTable (Query Table)	A table based on an external data query.
worksheet (Worksheet)	A table based on a worksheet data range.
xml (XML)	A table based on an XML mapping.

Referenced By

Referenced By
table@tableType (§3.5.1.2)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TableType">
   <restriction base="xsd:string">
      <enumeration value="worksheet"/>
      <enumeration value="xml"/>
      <enumeration value="queryTable"/>
   </restriction>
</simpleType>
```

3.18.81 ST_TargetScreenSize (Target Screen Size Types)

This simple type defines the collection of screen resolutions that are supported for this workbook.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
1024x768 (1024 x 768 Resolution)	Sets the target screen resolution to 1024x768 pixels.
1152x882 (1152 x 882 Resolution)	Sets the target screen resolution to 1152x882 pixels.
1152x900 (1152 x 900 Resolution)	Sets the target screen resolution to 1152x900 pixels
1280x1024 (1280 x 1024 Resolution)	Sets the target screen resolution to 1280x1024 pixels.
1600x1200 (1600 x 1200 Resolution)	Sets the target screen resolution to 1600x1200 pixels.
1800x1440 (1800 x 1440 Resolution)	Sets the target screen resolution to 1800x1440 pixels.
1920x1200 (1920 x 1200 Resolution)	Sets the target screen resolution to 1920x1200 pixels.
544x376 (544 x 376 Resolution)	Sets the target screen resolution to 544x376 pixels.
640x480 (640 x 480 Resolution)	Sets the target screen resolution to 640x480 pixels.
720x512 (720 x 512 Resolution)	Sets the target screen resolution to 720x512 pixels.
800x600 (800 x 600 Resolution)	Sets the target screen resolution to 800x600 pixels.

Referenced By
webPublishing@targetScreenSize (§3.2.24)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TargetScreenSize">
   <restriction base="xsd:string">
      <enumeration value="544x376"/>
      <enumeration value="640x480"/>
      <enumeration value="720x512"/>
      <enumeration value="800x600"/>
      <enumeration value="1024x768"/>
      <enumeration value="1152x882"/>
      <enumeration value="1152x900"/>
      <enumeration value="1280x1024"/>
      <enumeration value="1600x1200"/>
      <enumeration value="1800x1440"/>
      <enumeration value="1920x1200"/>
   </restriction>
</simpleType>
```

3.18.82 ST_TimePeriod (Time Period Types)

Used in a "contains dates" conditional formatting rule. These are dynamic time periods, which change based on the date the conditional formatting is refreshed / applied.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
last7Days (Last 7 Days)	A date in the last seven days.
lastMonth (Last Month)	A date occuring in the last calendar month.
lastWeek (Last Week)	A date occuring last week.
nextMonth (Next Month)	A date occuring in the next calendar month.
nextWeek (Next Week)	A date occuring next week.
thisMonth (This Month)	A date occuring in this calendar month.
thisWeek (This Week)	A date occuring this week.
today (Today)	Today's date.
tomorrow (Tomorrow)	Tomorrow's date.
yesterday (Yesterday)	Yesterday's date.

Referenced By
cfRule@timePeriod (§3.3.1.9)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TimePeriod">
   <restriction base="xsd:string">
      <enumeration value="today"/>
      <enumeration value="yesterday"/>
      <enumeration value="tomorrow"/>
      <enumeration value="last7Days"/>
      <enumeration value="thisMonth"/>
      <enumeration value="lastMonth"/>
      <enumeration value="nextMonth"/>
      <enumeration value="thisWeek"/>
      <enumeration value="lastWeek"/>
      <enumeration value="nextWeek"/>
   </restriction>
</simpleType>
```

3.18.83 ST_TotalsRowFunction (Totals Row Function Types)

An enumeration that specifies what function is used to aggregate the data in a column before it is displayed in the totals row.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
average (Average)	Represents the arithmetic mean.
count (Non Empty Cell Count)	Represents a count of the number of non-empty cells.
countNums (Count Numbers)	Represents the number of cells that contain numbers.
custom (Custom Formula)	Represents the formula provided in `totalsRowFormula`.
max (Maximum)	Represents the largest value.
min (Minimum)	Represents the smallest value.
none (None)	No total row.
stdDev (StdDev)	Represents the estimated standard deviation.
sum (Sum)	Represents the arithmetic sum.
var (Var)	Represents the estimated variance.

Referenced By
tableColumn@totalsRowFunction (§3.5.1.3)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TotalsRowFunction">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="sum"/>
      <enumeration value="min"/>
      <enumeration value="max"/>
      <enumeration value="average"/>
      <enumeration value="count"/>
      <enumeration value="countNums"/>
      <enumeration value="stdDev"/>
      <enumeration value="var"/>
      <enumeration value="custom"/>
   </restriction>
</simpleType>
```

3.18.84 ST_Type (Top N Evaluation Type)

This simple type defines the values for the Top N conditional formatting evaluation for the PivotTable. For more information on Top N conditional formatting, see the Sheet (§3.3.1) reference material.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
all (All)	Indicates that Top N conditional formatting is evaluated across the entire scope range.
column (Column Top N)	Indicates that Top N conditional formatting is evaluated for each column.
none (Top N None)	Indicates that Top N conditional formatting is not evaluated
row (Row Top N)	Indicates that Top N conditional formatting is evaluated for each row.

Referenced By
conditionalFormat@type (§3.10.1.18)

The following XML Schema fragment defines the contents of this simple type:

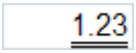
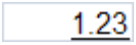
```
<simpleType name="ST_Type">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="all"/>
      <enumeration value="row"/>
      <enumeration value="column"/>
   </restriction>
</simpleType>
```

3.18.85 ST_UnderlineValues (Underline Types)

Represents the different types of possible underline formatting.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
double (Double Underline)	Double-line underlining under each character in the cell. underlines are drawn through the descenders of characters such as g and p. 
doubleAccounting (Accounting Double Underline)	Double-line accounting underlining under each character in the cell. The underlines are drawn under the descenders of characters such as g and p. 
none (None)	No underline.
single (Single Underline)	Single-line underlining under each character in the cell. The underline is drawn through the descenders of characters such as g and p. 
singleAccounting (Accounting Single Underline)	Single-line accounting underlining under each character in the cell. The underline is drawn under the descenders of characters such as g and p. 

Referenced By
u@val (§3.4.13)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_UnderlineValues">
   <restriction base="xsd:string">
      <enumeration value="single"/>
      <enumeration value="double"/>
      <enumeration value="singleAccounting"/>
      <enumeration value="doubleAccounting"/>
      <enumeration value="none"/>
   </restriction>
</simpleType>
```

3.18.86 ST_UnsignedIntHex (Hex Unsigned Integer)

This simple type represents the Hex representation of an unsigned integer.

This simple type's contents are a restriction of the XML Schema hexBinary datatype.

This simple type also specifies the following restrictions:

- This simple type's contents must have a length of exactly 4 characters.

Referenced By
bgColor@rgb (§3.8.3); color@rgb (§3.3.1.14); e@bc (§3.10.1.27); e@fc (§3.10.1.27); fgColor@rgb (§3.8.18); m@bc (§3.10.1.50); m@fc (§3.10.1.50); n@bc (§3.10.1.60); n@fc (§3.10.1.60); rgbColor@rgb (§3.8.34); s@bc (§3.10.1.85); s@fc (§3.10.1.85); t@bc (§3.9.16); t@fc (§3.9.16); tabColor@rgb (§3.3.1.90)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_UnsignedIntHex">
   <restriction base="xsd:hexBinary">
      <length value="4"/>
   </restriction>
</simpleType>
```

3.18.87 ST_UnsignedShortHex (Unsigned Short Hex)

This simple type defines the Hex representation of an unsigned short.

This simple type's contents are a restriction of the XML Schema hexBinary datatype.

This simple type also specifies the following restrictions:

- This simple type's contents must have a length of exactly 2 characters.

Referenced By
fileSharing@reservationPassword (§3.2.12); protectedRange@password (§3.3.1.69); sheetProtection@password (§3.3.1.81); sheetProtection@password (§3.3.1.82); workbookProtection@revisionsPassword (§3.2.29); workbookProtection@workbookPassword (§3.2.29)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_UnsignedShortHex">
   <restriction base="xsd:hexBinary">
      <length value="2"/>
   </restriction>
</simpleType>
```

3.18.88 ST_UpdateLinks (Update Links Behavior Types)

This simple type defines when the application updates links to other workbooks when the workbook is opened.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description

Enumeration Value	Description
always (Always Update Links)	Indicates that links to other workbooks are always updated when the workbook is opened. The application will not display an alert in the user interface (UI).
never (Never Update Links)	Indicates that links to other workbooks are never updated when the workbook is opened. The application will not display an alert in the user interface.
userSet (User Set)	Indicates that the end-user specified whether they receive an alert to update links to other workbooks when the workbook is opened. For example, the application may expose this option in an application settings dialog.

Referenced By
workbookPr@updateLinks (§3.2.28)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_UpdateLinks">
   <restriction base="xsd:string">
      <enumeration value="userSet"/>
      <enumeration value="never"/>
      <enumeration value="always"/>
   </restriction>
</simpleType>
```

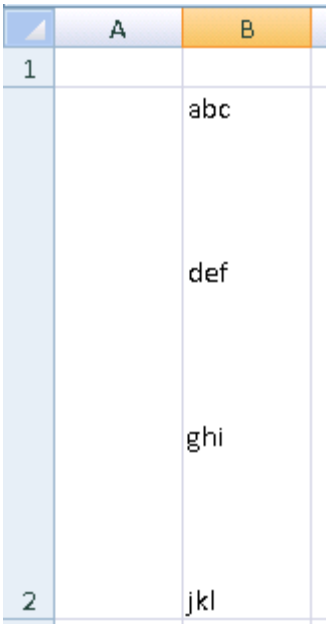
3.18.89 ST_VerticalAlignment (Vertical Alignment Types)

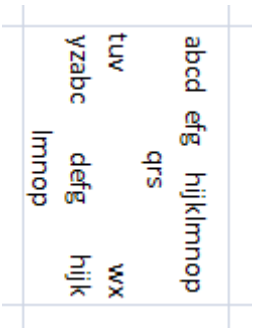
This enumeration value indicates the type of vertical alignment for a cell, i.e., whether it is aligned top, bottom, vertically centered, justified or distributed.

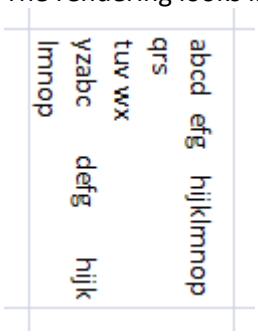
This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
bottom (Aligned To Bottom)	The vertical alignment is aligned-to-bottom.
center (Centered Vertical Alignment)	The vertical alignment is centered across the height of the cell.
distributed (Distributed Vertical Alignment)	When text direction is horizontal: the vertical alignment of lines of text is distributed vertically, where each line of text inside the cell is evenly distributed across the height of the cell, with flush top

Enumeration Value	Description
	and bottom margins. When text direction is vertical: behaves exactly as distributed horizontal alignment. The first words in a line of text (appearing at the top of the cell) are flush with the top edge of the cell, and the last words of a line of text are flush with the bottom edge of the cell, and the line of text is distributed evenly from top to bottom. [*Example*: Horizontal text: this first example shows four lines of text (read horizontally from left to right) distributed vertically across the height of the cell. The first line is "abc", the second line is "def", the third line is "ghi" and the fourth line is "jkl".  Vertical text: this second example shows three lines of text (read vertically from top to bottom) distributed vertically across the height of the cell. The lines of text are: abcd efg hijklmnop qrs tuv wx yzabc defg hijk lmnop The rendering looks like this:

Enumeration Value	Description
	 *end example*]
justify (Justified Vertically)	When text direction is horizontal: the vertical alignment of lines of text is distributed vertically, where each line of text inside the cell is evenly distributed across the height of the cell, with flush top and bottom margins. When text direction is vertical: similar behavior as horizontal justification. The alignment is justified (flush top and bottom in this case). For each line of text, each line of the wrapped text in a cell is aligned to the top and bottom (except the last line). If no single line of text wraps in the cell, then the text is not justified. [*Example*: Horizontal text: this first example shows four lines of text (read horizontally from left to right) justified vertically across the height of the cell. The first line is "abc", the second line is "def", the third line is "ghi" and the fourth line is "jkl".

Enumeration Value	Description
	 Vertical text: this second example shows three lines of text (read vertically from top to bottom) distributed vertically across the height of the cell. The lines of text are: abcd efg hijklmnop qrs tuv wx yzabc defg hijk lmnop The rendering looks like this:  *end example*]
top (Align Top)	The vertical alignment is aligned-to-top.

Referenced By
alignment@vertical (§3.8.1)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_VerticalAlignment">
   <restriction base="xsd:string">
      <enumeration value="top"/>
      <enumeration value="center"/>
      <enumeration value="bottom"/>
      <enumeration value="justify"/>
      <enumeration value="distributed"/>
   </restriction>
</simpleType>
```

3.18.90 ST_VerticalAlignRun (Vertical Alignment Run Types)

Defines the possible settings for vertical alignment of a run of text. This is used to get superscript or subscript text without altering the font size properties of the rest of the text run.

[*Example*:

```
<rPr>
  <vertAlign val="subscript"/>
</rPr>
```

end example]

The above example shows a run which shall be positioning as subscript when displaying its contents. The resulting run is positioned as `subscript`, therefore it is rendered in a smaller size below the default baseline location for the contents of the run.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
baseline (Baseline)	Returns the text in this run to the baseline, default, alignment, and returns it to the original font size.
subscript (Subscript)	Specifies that this text should be subscript. Lowers the text in this run below the baseline and changes it to a smaller size, if a smaller size is available.
superscript (Superscript)	Specifies that this text should be superscript. Raises the text in this run above the baseline and changes it to a smaller size, if a smaller size is available.

Referenced By
vertAlign@val (§3.4.14)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_VerticalAlignRun">
   <restriction base="xsd:string">
      <enumeration value="baseline"/>
      <enumeration value="superscript"/>
      <enumeration value="subscript"/>
   </restriction>
</simpleType>
```

3.18.91 ST_Visibility (Visibility Types)

This simple type defines the possible states for sheet visibility.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
hidden (Hidden)	Indicates the book window is hidden, but can be shown by the user via the user interface.
veryHidden (Very Hidden)	Indicates the sheet is hidden and cannot be shown in the user interface (UI). This state is only available programmatically.
visible (Visible)	Indicates the workbook window is visible.

Referenced By
workbookView@visibility (§3.2.30)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Visibility">
   <restriction base="xsd:string">
      <enumeration value="visible"/>
      <enumeration value="hidden"/>
      <enumeration value="veryHidden"/>
   </restriction>
</simpleType>
```

3.18.92 ST_VolDepType (Volatile Dependency Types)

This simple type defines the dependency types available for this workbook.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
olapFunctions (OLAP Formulas)	Indicates that the type is Cube Functions.
realTimeData (Real Time Data)	Indicates that the type is Real Time Data (RTD).

Referenced By
volType@type (§3.15.5)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_VolDepType">
   <restriction base="xsd:string">
      <enumeration value="realTimeData"/>
      <enumeration value="olapFunctions"/>
   </restriction>
</simpleType>
```

3.18.93 ST_VolValueType (Volatile Dependency Value Types)

This simple type defines the data type of the values in the dependency cache.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
b (Boolean)	Indicates topic value is a boolean.
e (Error)	Indicates topic value is an error.
n (Real Number)	Indicates topic value is a real number.
s (String)	Indicates topic value is a string.

Referenced By
tp@t (§3.15.3)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_VolValueType">
   <restriction base="xsd:string">
      <enumeration value="b"/>
      <enumeration value="n"/>
      <enumeration value="e"/>
      <enumeration value="s"/>
   </restriction>
</simpleType>
```

3.18.94 ST_WebSourceType (Web Source Type)

This is an enumeration of types of objects which can be selected from the workbook to be published as HTML. For example, the entire sheet can be published, or a narrower set of objects on the sheet can be published, like a chart or a range.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
autoFilter (AutoFilter)	Auto filter
chart (Chart)	Chart
label (Label)	Label
pivotTable (PivotTable)	PivotTable
printArea (Print Area)	Print area
query (QueryTable)	Query Table
range (Range)	Range of cells
sheet (All Sheet Content)	All content of a sheet

Referenced By
webPublishItem@sourceType (§3.3.1.94)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_WebSourceType">
   <restriction base="xsd:string">
      <enumeration value="sheet"/>
      <enumeration value="printArea"/>
      <enumeration value="autoFilter"/>
      <enumeration value="range"/>
      <enumeration value="chart"/>
      <enumeration value="pivotTable"/>
      <enumeration value="query"/>
      <enumeration value="label"/>
   </restriction>
</simpleType>
```

3.18.95 ST_XmlDataType (XML Data Types)

Represents a W3C XML built-in datatype name (http://www.w3.org/TR/xmlschema-2/)

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
anyType (Any Type)	See http//www.w3.org/2001/XMLSchema
anyURI (Any URI)	See http//www.w3.org/2001/XMLSchema
base64Binary (Base 64 Encoded Binary)	See http//www.w3.org/2001/XMLSchema
boolean (Boolean)	See http//www.w3.org/2001/XMLSchema
byte (Byte)	See http//www.w3.org/2001/XMLSchema
date (Date)	See http//www.w3.org/2001/XMLSchema
dateTime (Date Time)	See http//www.w3.org/2001/XMLSchema
decimal (Decimal)	See http//www.w3.org/2001/XMLSchema
double (Double)	See http//www.w3.org/2001/XMLSchema
duration (Duration)	See http//www.w3.org/2001/XMLSchema
ENTITIES (ENTITIES)	See http//www.w3.org/2001/XMLSchema
ENTITY (ENTITY)	See http//www.w3.org/2001/XMLSchema
float (Float)	See http//www.w3.org/2001/XMLSchema
gDay (gDay)	See http//www.w3.org/2001/XMLSchema
gMonth (gMonth)	See http//www.w3.org/2001/XMLSchema
gMonthDay (gMonthDays)	See http//www.w3.org/2001/XMLSchema
gYear (gYear)	See http//www.w3.org/2001/XMLSchema
gYearMonth (gYearMonth)	See http//www.w3.org/2001/XMLSchema
hexBinary (Hex Binary)	See http//www.w3.org/2001/XMLSchema
ID (ID)	See http//www.w3.org/2001/XMLSchema
IDREF (IDREF)	See http//www.w3.org/2001/XMLSchema
IDREFS (IDREFS)	See http//www.w3.org/2001/XMLSchema
int (Integer)	See http//www.w3.org/2001/XMLSchema
integer (Integer)	See http//www.w3.org/2001/XMLSchema
language (Language)	See http//www.w3.org/2001/XMLSchema
long (Long)	See http//www.w3.org/2001/XMLSchema
Name (Name)	See http//www.w3.org/2001/XMLSchema
NCName (NCName)	See http//www.w3.org/2001/XMLSchema
negativeInteger (Negative Integer)	See http//www.w3.org/2001/XMLSchema
NMTOKEN (NMTOKEN)	See http//www.w3.org/2001/XMLSchema
NMTOKENS (NMTOKENS)	See http//www.w3.org/2001/XMLSchema
nonNegativeInteger (Non Negative Integer)	See http//www.w3.org/2001/XMLSchema
nonPositiveInteger (Non Positive Integer)	See http//www.w3.org/2001/XMLSchema
normalizedString (Normalized String)	See http//www.w3.org/2001/XMLSchema

Enumeration Value	Description
NOTATION (Notation)	See http//www.w3.org/2001/XMLSchema
positiveInteger (Positive Integer)	See http//www.w3.org/2001/XMLSchema
QName (Qname)	See http//www.w3.org/2001/XMLSchema
short (Short)	See http//www.w3.org/2001/XMLSchema
string (String)	See http//www.w3.org/2001/XMLSchema
time (Time)	See http//www.w3.org/2001/XMLSchema
token (Token)	See http//www.w3.org/2001/XMLSchema
unsignedByte (Unsigned Byte)	See http//www.w3.org/2001/XMLSchema
unsignedInt (Unsigned Integer)	See http//www.w3.org/2001/XMLSchema
unsignedLong (Unsigned Long)	See http//www.w3.org/2001/XMLSchema
unsignedShort (Unsigned Short)	See http//www.w3.org/2001/XMLSchema

Referenced By
xmlColumnPr@xmlDataType (§3.5.1.7); xmlPr@xmlDataType (§3.5.2.4)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_XmlDataType">
   <restriction base="xsd:string">
      <enumeration value="string"/>
      <enumeration value="normalizedString"/>
      <enumeration value="token"/>
      <enumeration value="byte"/>
      <enumeration value="unsignedByte"/>
      <enumeration value="base64Binary"/>
      <enumeration value="hexBinary"/>
      <enumeration value="integer"/>
      <enumeration value="positiveInteger"/>
      <enumeration value="negativeInteger"/>
      <enumeration value="nonPositiveInteger"/>
      <enumeration value="nonNegativeInteger"/>
      <enumeration value="int"/>
      <enumeration value="unsignedInt"/>
      <enumeration value="long"/>
      <enumeration value="unsignedLong"/>
      <enumeration value="short"/>
      <enumeration value="unsignedShort"/>
      <enumeration value="decimal"/>
      <enumeration value="float"/>
      <enumeration value="double"/>
      <enumeration value="boolean"/>
      <enumeration value="time"/>
      <enumeration value="dateTime"/>
      <enumeration value="duration"/>
      <enumeration value="date"/>
      <enumeration value="gMonth"/>
      <enumeration value="gYear"/>
      <enumeration value="gYearMonth"/>
      <enumeration value="gDay"/>
      <enumeration value="gMonthDay"/>
      <enumeration value="Name"/>
      <enumeration value="QName"/>
      <enumeration value="NCName"/>
      <enumeration value="anyURI"/>
      <enumeration value="language"/>
      <enumeration value="ID"/>
      <enumeration value="IDREF"/>
      <enumeration value="IDREFS"/>
      <enumeration value="ENTITY"/>
      <enumeration value="ENTITIES"/>
      <enumeration value="NOTATION"/>
      <enumeration value="NMTOKEN"/>
      <enumeration value="NMTOKENS"/>
      <enumeration value="anyType"/>
   </restriction>
</simpleType>
```

3.18.96 ST_Xstring (Escaped String)

String of characters with support for escaped invalid-XML characters.

For all characters which cannot be represented in XML as defined by the XML 1.0 specification, the characters are escaped using the Unicode numerical character representation escape character format _xHHHH_, where H represents a hexadecimal character in the character's value. [*Example*: The Unicode character 8 is invalid in an XML 1.0 document, so it shall be escaped as _x0008_. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

Referenced By
author (§3.7.1); b@c (§3.10.1.2); cacheField@caption (§3.10.1.3); cacheField@formula (§3.10.1.3); cacheField@name (§3.10.1.3); cacheField@propertyName (§3.10.1.3); cacheHierarchy@allCaption (§3.10.1.6); cacheHierarchy@allUniqueName (§3.10.1.6); cacheHierarchy@caption (§3.10.1.6); cacheHierarchy@defaultMemberUniqueName (§3.10.1.6); cacheHierarchy@dimensionUniqueName (§3.10.1.6); cacheHierarchy@displayFolder (§3.10.1.6); cacheHierarchy@measureGroup (§3.10.1.6); cacheHierarchy@uniqueName (§3.10.1.6); calculatedItem@formula (§3.10.1.8); calculatedMember@hierarchy (§3.10.1.10); calculatedMember@mdx (§3.10.1.10); calculatedMember@memberName (§3.10.1.10); calculatedMember@name (§3.10.1.10); calculatedMember@parent (§3.10.1.10); cellSmartTagPr@key (§3.3.1.5); cellSmartTagPr@val (§3.3.1.5); cellStyle@name (§3.8.7); cfvo@val (§3.3.1.10); connection@description (§3.13.1); connection@name (§3.13.1); connection@odcFile (§3.13.1); connection@singleSignOnId (§3.13.1); connection@sourceFile (§3.13.1); customFilter@val (§3.3.2.2); customPr@name (§3.3.1.20); customWorkbookView@name (§3.2.3); d@c (§3.10.1.21); dataField@name (§3.10.1.22); dataRef@name (§3.3.1.28); dataRef@sheet (§3.3.1.28); dataValidation@error (§3.3.1.30); dataValidation@errorTitle (§3.3.1.30); dataValidation@prompt (§3.3.1.30); dataValidation@promptTitle (§3.3.1.30); dbPr@command (§3.13.3); dbPr@connection (§3.13.3); dbPr@serverCommand (§3.13.3); ddeItem@name (§3.14.2); ddeLink@ddeService (§3.14.4); ddeLink@ddeTopic (§3.14.4); definedName@name (§3.14.5); definedName@refersTo (§3.14.5); deletedField@name (§3.12.1); dimension@caption (§3.10.1.24); dimension@name (§3.10.1.24); dimension@uniqueName (§3.10.1.24); e@c (§3.10.1.27); e@v (§3.10.1.27); evenFooter (§3.3.1.35); evenHeader (§3.3.1.36); fileSharing@userName (§3.2.12); filter@description (§3.10.1.33); filter@name (§3.10.1.33); filter@stringValue1 (§3.10.1.33); filter@stringValue2 (§3.10.1.33); filter@val (§3.3.2.6); firstFooter (§3.3.1.38); firstHeader (§3.3.1.39); functionGroup@name (§3.2.14); futureMetadata@name (§3.9.4); group@caption (§3.10.1.37); group@name (§3.10.1.37); group@uniqueName (§3.10.1.37); group@uniqueParent (§3.10.1.37); groupLevel@caption (§3.10.1.39); groupLevel@uniqueName (§3.10.1.39); groupMember@uniqueName (§3.10.1.41); header@userName (§3.11.1.1); hyperlink@display (§3.3.1.44); hyperlink@location (§3.3.1.44); hyperlink@tooltip (§3.3.1.44); inputCells@val (§3.3.1.49); item@n (§3.10.1.45); kpi@caption (§3.10.1.47); kpi@displayFolder (§3.10.1.47); kpi@goal (§3.10.1.47); kpi@measureGroup (§3.10.1.47); kpi@parent (§3.10.1.47); kpi@status (§3.10.1.47); kpi@time (§3.10.1.47); kpi@trend (§3.10.1.47); kpi@uniqueName (§3.10.1.47); kpi@value (§3.10.1.47); kpi@weight (§3.10.1.47); m@c (§3.10.1.50); main@first (§3.15.1); measureGroup@caption (§3.10.1.53); measureGroup@name (§3.10.1.53); member@name (§3.10.1.55); metadataType@name (§3.9.10); mp@name (§3.10.1.57); n@c (§3.10.1.60); name@val (§3.8.29); numFmt@formatCode (§3.8.30); oddFooter (§3.3.1.55); oddHeader (§3.3.1.56); olapPr@localConnection (§3.13.5); oleItem@name (§3.14.9); oleLink@progId (§3.14.11); oleObject@link (§3.3.1.57); pageField@cap (§3.10.1.62); pageField@name (§3.10.1.62); pageItem@name (§3.10.1.64); parameter@cell (§3.13.6); parameter@name (§3.13.6); parameter@prompt (§3.13.6);

Referenced By
parameter@string (§3.13.6); pivotCacheDefinition@refreshedBy (§3.10.1.67); pivotField@name (§3.10.1.69); pivotField@subtotalCaption (§3.10.1.69); pivotField@uniqueMemberProperty (§3.10.1.69); pivotHierarchy@caption (§3.10.1.72); pivotTableDefinition@colHeaderCaption (§3.10.1.73); pivotTableDefinition@dataCaption (§3.10.1.73); pivotTableDefinition@errorCaption (§3.10.1.73); pivotTableDefinition@grandTotalCaption (§3.10.1.73); pivotTableDefinition@missingCaption (§3.10.1.73); pivotTableDefinition@name (§3.10.1.73); pivotTableDefinition@pageStyle (§3.10.1.73); pivotTableDefinition@pivotTableStyle (§3.10.1.73); pivotTableDefinition@rowHeaderCaption (§3.10.1.73); pivotTableDefinition@tag (§3.10.1.73); pivotTableDefinition@vacatedStyle (§3.10.1.73); protectedRange@name (§3.3.1.69); query@mdx (§3.10.1.75); queryTable@name (§3.12.2); queryTableField@name (§3.12.4); rangeSet@name (§3.10.1.79); rangeSet@sheet (§3.10.1.79); rcmt@author (§3.11.1.11); rdn@comment (§3.11.1.13); rdn@customMenu (§3.11.1.13); rdn@description (§3.11.1.13); rdn@help (§3.11.1.13); rdn@name (§3.11.1.13); rdn@oldComment (§3.11.1.13); rdn@oldCustomMenu (§3.11.1.13); rdn@oldDescription (§3.11.1.13); rdn@oldHelp (§3.11.1.13); rdn@oldStatusBar (§3.11.1.13); rdn@statusBar (§3.11.1.13); rFont@val (§3.4.5); ris@name (§3.11.1.18); rsnm@newName (§3.11.1.22); rsnm@oldName (§3.11.1.22); s@c (§3.10.1.85); s@v (§3.13.8); s@v (§3.10.1.85); scenario@comment (§3.3.1.73); scenario@name (§3.3.1.73); scenario@user (§3.3.1.73); serverFormat@culture (§3.10.1.86); serverFormat@format (§3.10.1.86); set@setDefinition (§3.10.1.88); sheet@name (§3.2.19); sheetName@val (§3.14.15); smartTagType@name (§3.2.22); smartTagType@namespaceUri (§3.2.22); smartTagType@url (§3.2.22); sortCondition@customList (§3.3.1.88); ST_Formula (§3.18.36); stp (§3.15.2); t (§3.4.12); t@ct (§3.9.16); table@comment (§3.5.1.2); table@dataCellStyle (§3.5.1.2); table@displayName (§3.5.1.2); table@headerRowCellStyle (§3.5.1.2); table@name (§3.5.1.2); table@totalsRowCellStyle (§3.5.1.2); tableColumn@dataCellStyle (§3.5.1.3); tableColumn@headerRowCellStyle (§3.5.1.3); tableColumn@name (§3.5.1.3); tableColumn@totalsRowCellStyle (§3.5.1.3); tableColumn@totalsRowLabel (§3.5.1.3); tableColumn@uniqueName (§3.5.1.3); tableStyleInfo@name (§3.5.1.5); textPr@decimal (§3.13.12); textPr@delimiter (§3.13.12); textPr@sourceFile (§3.13.12); textPr@thousands (§3.13.12); undo@dn (§3.11.1.25); userInfo@name (§3.11.2.1); v (§3.3.1.93); val (§3.14.17); webPr@editPage (§3.13.13); webPr@post (§3.13.13); webPr@url (§3.13.13); webPublishItem@destinationFile (§3.3.1.94); webPublishItem@divId (§3.3.1.94); webPublishItem@sourceObject (§3.3.1.94); webPublishItem@title (§3.3.1.94); webPublishObject@destinationFile (§3.2.25); webPublishObject@divId (§3.2.25); webPublishObject@sourceObject (§3.2.25); webPublishObject@title (§3.2.25); worksheetSource@name (§3.10.1.95); worksheetSource@sheet (§3.10.1.95); xmlCellPr@uniqueName (§3.5.2.3); xmlColumnPr@xpath (§3.5.1.7); xmlPr@xpath (§3.5.2.4)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Xstring">
    <restriction base="xsd:string"/>
</simpleType>
```

4. PresentationML Reference Material

The subordinate subclauses specify the semantics for the XML markup comprising a PresentationML document, as defined by Part 1 of this Standard.

4.1 Table of Contents

This subclause is informative.

End of informative text.

4.2 Basics

The Basics portion of the PresentationML framework defines all commonly used elements for a PresentationML document. That is, those elements that do not fall into one of the following subclauses for PresentationML content as defined within this section.

4.2.1 custData (Customer Data)

This element specifies customer data which allows for the specifying and persistence of customer specific data within the presentation.

Parent Elements
custDataLst (§4.4.1.16)

Attributes	Description
id (Relationship ID)	This attribute specifies the relationship id for referencing other resources outside the scope of the current PresentationML file.

Attributes	Description
Namespace: .../officeDocument/2006/relationships	The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustomerData">
   <attribute ref="r:id" use="required"/>
</complexType>
```

4.2.2 custShow (Custom Show)

This element specifies a custom show which is an ordered list of a group of slides that are contained within the presentation. The custom show element allows for the specification of a presentation order that is different from the order in which the slides themselves are stored.

[Example: Consider the following custom show list that outlines a couple custom shows for a given set of slides.

```
<p:custShowLst>
  <p:custShow name="Custom Show 1" id="0">
    <p:sldLst>
      <p:sld r:id="rId4"/>
      <p:sld r:id="rId3"/>
      <p:sld r:id="rId2"/>
      <p:sld r:id="rId5"/>
    </p:sldLst>
  </p:custShow>
  <p:custShow name="Custom Show 2" id="1">
    <p:sldLst>
      <p:sld r:id="rId4"/>
      <p:sld r:id="rId5"/>
    </p:sldLst>
  </p:custShow>
</p:custShowLst>
```

In the above example there are two custom shows specified. The first specifies to present the slides in the order of 4, 3, 2 then 5 while the second specifies to play only slide 4 then 5. End example]

Parent Elements
htmlPubPr (§4.3.1.13); showPr (§4.3.1.28)

Attributes	Description
id (Custom Show Identifier)	This attribute specifies the custom show identification number. This is a number given that should be unique within the presentation document. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustomShowId">
   <attribute name="id" type="xsd:unsignedInt" use="required"/>
</complexType>
```

4.2.3 ext (Extension)

This element specifies an extension that is used for future extensions to the current version of DrawingML. This allows for the specifying of currently unknown elements in the future that will be used for later versions of generating applications.

Parent Elements
extLst (§4.2.4); extLst (§4.2.5)

Child Elements	Subclause
Any element from any namespace	n/a

Attributes	Description
uri (Uniform Resource Identifier)	This attribute specifies the URI, or uniform resource identifier that represents the data stored under this tag. The URI is used to identify the correct 'server' that can process the contents of this tag. The possible values for this attribute are defined by the XML Schema token datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Extension">
   <sequence>
      <any processContents="lax"/>
   </sequence>
   <attribute name="uri" type="xsd:token"/>
</complexType>
```

4.2.4 extLst (Extension List with Modification Flag)

This element specifies the extension list with modification ability within which all future extensions of type ext will be defined. The extension list along with corresponding future extensions is used to extend the storage

capabilities of the PresentationML framework. This allows for various new types of data to be stored natively within the framework.

[*Note*: Using this extLst element allows the generating application to store whether this extension property has been modified. *end note*]

Parent Elements
cm (§4.5.1); cxnSp (§4.4.1.17); graphicFrame (§4.4.1.18); grpSp (§4.4.1.19); handoutMaster (§4.4.1.21); hf (§4.4.1.22); notes (§4.4.1.23); notesMaster (§4.4.1.24); ph (§4.4.1.33); pic (§4.4.1.34); sld (§4.4.1.35); sldLayout (§4.4.1.36); sldMaster (§4.4.1.39); sp (§4.4.1.40); spTree (§4.4.1.42); timing (§4.4.1.44); transition (§4.4.1.46)

Child Elements	Subclause
ext (Extension)	§4.2.3

Attributes	Description
mod (Modify)	This attribute specifies whether the data contained within this element has been modified and should thus be processed again by the generating application. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ExtensionListModify">
   <sequence>
      <group ref="EG_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="mod" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.2.5 extLst (Extension List)

This element specifies the extension list within which all future extensions of type ext will be defined. The extension list along with corresponding future extensions is used to extend the storage capabilities of the PresentationML framework. This allows for various new types of data to be stored natively within the framework.

Parent Elements
bgPr (§4.4.1.2); cmAuthor (§4.5.2); control (§4.4.2.1); cSld (§4.4.1.15); custShow (§4.3.1.5); embed (§4.4.2.2); handoutMasterId (§4.3.1.11); htmlPubPr (§4.3.1.13); link (§4.4.2.3); normalViewPr (§4.3.2.6); notesMasterId (§4.3.1.18); notesTextViewPr (§4.3.2.7); notesViewPr (§4.3.2.8); nvPr (§4.4.1.30); outlineViewPr (§4.3.2.10); photoAlbum (§4.3.1.22); presentation (§4.3.1.24); presentationPr (§4.3.1.25); prnPr (§4.3.1.26); showPr (§4.3.1.28); sldId (§4.3.1.29); sldLayoutId (§4.4.1.37); sldMasterId (§4.3.1.32); sldSyncPr (§4.7.1); slideViewPr (§4.3.2.16); sorterViewPr (§4.3.2.17); txStyles (§4.4.1.48); viewPr (§4.3.2.18); webPr (§4.3.1.36)

Child Elements	Subclause
ext (Extension)	§4.2.3

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ExtensionList">
   <sequence>
      <group ref="EG_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.2.6 sld (Presentation Slide)

This element specifies a slide within a slide list. The slide list is used to specify an ordering of slides.

[Example: Consider the following custom show with an ordering of slides.

```
<p:custShowLst>
  <p:custShow name="Custom Show 1" id="0">
    <p:sldLst>
      <p:sld r:id="rId4"/>
      <p:sld r:id="rId3"/>
      <p:sld r:id="rId2"/>
      <p:sld r:id="rId5"/>
    </p:sldLst>
  </p:custShow>
</p:custShowLst>
```

In the above example the order specified to present the slides is slide 4, then 3, 2 and finally 5. End example]

Parent Elements
sldLst (§4.3.1.31)

Attributes	Description
id (Relationship ID) Namespace: .../officeDocument/2006/relationships	This attribute specifies the relationship id that is used to reference to the actual slide XML file that contains all the information to the slide listed within the slide list. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SlideRelationshipListEntry">
   <attribute ref="r:id" use="required"/>
</complexType>
```

4.2.7 sldAll (All Slides)

This attribute specifies all slides instead of a given range of slides for use within the html publishing properties as well as the show properties.

Parent Elements
htmlPubPr (§4.3.1.13); showPr (§4.3.1.28)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

4.2.8 sldRg (Slide Range)

This element specifies a slide range for use within the html publishing properties as well as the show properties.

[*Note*: The indexes used here correlate directly with the presentation slide numbers which they reference to. That is the slide range must be greater than or equal to 1 and also less than or equal to the number of slides in the presentation document. *end note*]

Parent Elements
htmlPubPr (§4.3.1.13); showPr (§4.3.1.28)

Attributes	Description
end (End)	This attribute defines the end of the index range. The possible values for this attribute are defined by the ST_Index simple type (§4.8.7).
st (Start)	This attribute defines the start of the index range. The possible values for this attribute are defined by the ST_Index simple type (§4.8.7).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_IndexRange">
   <attribute name="st" type="ST_Index" use="required"/>
   <attribute name="end" type="ST_Index" use="required"/>
</complexType>
```

4.2.9 tags (Customer Data Tags)

This element specifies the existence of customer data in the form of tags. This allows for the storage of customer data within the PresenationML framework. While this is similar to the ext tag in that it can be used store

information, this tag mainly focuses on referencing to other parts of the presentation document. This is accomplished via the relationship identification attribute that is required for all specified tags.

Parent Elements
custDataLst (§4.4.1.16)

Attributes	Description
id (Relationship ID) Namespace: .../officeDocument/2006/relationships	This attribute specifies the relationship identifier for the customer data tag. This allows for a link to a resource that is external from the current XML document but still contained within the presentation document. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TagsData">
   <attribute ref="r:id" use="required"/>
</complexType>
```

4.3 Presentation

The Presentation portion of the PresentationML framework houses a set of elements that describe the storing of presentation-wide and view-specific properties. The presentation-wide properties are those that pertain to the entire presentation. The view-specific properties assist the generating application and viewing application by storing parameters that pertain to the final delivery of the presentation.

4.3.1 Presentation Properties

This section contains all presentation-level properties that pertain to a presentation document:

4.3.1.1 bold (Bold Embedded Font)

This element specifies a bold embedded font that is linked to a parent typeface. Once specified, this bold version of the given typeface name is available for use within the presentation. The actual font data is referenced using a relationships file that contains links to all fonts available. This font data contains font information for each of the characters to be made available.

[*Example*: Consider the following embedded font with a bold version specified.

```
<p:embeddedFont>
  <p:font typeface="MyFont" pitchFamily="34" charset="0"/>
  <p:bold r:id="rId2"/>
</p:embeddedFont>
```

end example]

[*Note*: Not all characters for a typeface must be stored. It is up to the generating application to determine which characters are to be stored in the corresponding font data files. *end note*]

Parent Elements
embeddedFont (§4.3.1.8)

Attributes	Description
id (Relationship Identifier) Namespace: .../officeDocument/2006/relationships	Specifies the relationship identifier that is used in conjunction with a corresponding relationship file to resolve the location of this embedded font that is referenced in a presentation. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EmbeddedFontDataId">
   <attribute ref="r:id" use="required"/>
</complexType>
```

4.3.1.2 boldItalic (Bold Italic Embedded Font)

This element specifies a bold italic embedded font that is linked to a parent typeface. Once specified, this bold italic version of the given typeface name is available for use within the presentation. The actual font data is referenced using a relationships file that contains links to all fonts available. This font data contains font information for each of the characters to be made available.

[*Example*: Consider the following embedded font with a bold italic version specified.

```
<p:embeddedFont>
  <p:font typeface="MyFont" pitchFamily="34" charset="0"/>
  <p:boldItalic r:id="rId2"/>
</p:embeddedFont>
```

end example]

[*Note*: Not all characters for a typeface must be stored. It is up to the generating application to determine which characters are to be stored in the corresponding font data files. *end note*]

Parent Elements
embeddedFont (§4.3.1.8)

Attributes	Description

Attributes	Description
id (Relationship Identifier) Namespace: .../officeDocument/2006/relationships	Specifies the relationship identifier that is used in conjunction with a corresponding relationship file to resolve the location of this embedded font that is referenced in a presentation. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EmbeddedFontDataId">
   <attribute ref="r:id" use="required"/>
</complexType>
```

4.3.1.3 browse (Browse Slide Show Mode)

This element specifies that the presentation slide show should be viewed in a single window or browse mode, instead of full screen.

[*Example*: Consider the following presentation that is set to be viewed in a browse mode.

```
<p:presentationPr xmlns:a="" xmlns:r="" xmlns:p="">
  <p:showPr>
    ..
    <p:browse showScrollbar="0"/>
    ..
  </p:showPr>
</p:presentationPr>
```

end example]

Parent Elements
showPr (§4.3.1.28)

Attributes	Description
showScrollbar (Show Scroll Bar in Window)	Specifies whether to show the scroll bar in the viewing window. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShowInfoBrowse">
   <attribute name="showScrollbar" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

4.3.1.4 clrMru (Color MRU)

This specifies the most recently used user-selected colors within the presentation. This list contains custom user-selected colors outside the presentation's theme colors, enabling the application to expose these additional color choices for easy reuse. The first item in the list is the most recently used color.

[*Example*: Consider the following presentation with two user-selected colors in the color MRU list.

```
<p:presentationPr xmlns:a="" xmlns:r="" xmlns:p="">
  ..
  <p:clrMru>
    <a:srgbClr val="5361EB"/>
    <a:srgbClr val="CCECFF"/>
  </p:clrMru>
  ..
</p:presentationPr>
```

end example]

Parent Elements
presentationPr (§4.3.1.25)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ColorMRU">
   <sequence>
      <group ref="EG_ColorChoice" minOccurs="0" maxOccurs="10"/>
   </sequence>
</complexType>
```

4.3.1.5 custShow (Custom Show)

This element specifies a custom show that defines a specific slide sequence that the slides will be displayed in. This allows for many variants of the same set of slides to be presented.

[*Example*: Consider the following custom show using three slides.

```
<p:custShow name="Custom Show 1" id="0">
```

```
    <p:sldLst>
      <p:sld r:id="rId5"/>
      <p:sld r:id="rId2"/>
      <p:sld r:id="rId4"/>
    </p:sldLst>
  </p:custShow>
```

Notice here that the custom show specifies a show, or presentation, where slide 5 will be shown first, then slide 2 and finally slide 4. *end example*]

Parent Elements
custShowLst (§4.3.1.6)

Child Elements	Subclause
extLst (Extension List)	§4.2.5
sldLst (List of Presentation Slides)	§4.3.1.31

Attributes	Description
id (Custom Show ID)	Specifies the identification number for this custom show. This should be unique among all the custom shows within the corresponding presentation. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
name (Custom Show Name)	Specifies a name for the custom show. The possible values for this attribute are defined by the ST_Name simple type (§4.8.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustomShow">
   <sequence>
      <element name="sldLst" type="CT_SlideRelationshipList" minOccurs="1" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="name" type="ST_Name" use="required"/>
   <attribute name="id" type="xsd:unsignedInt" use="required"/>
</complexType>
```

4.3.1.6 custShowLst (List of Custom Shows)

This element specifies a list of all custom shows that are available within the corresponding presentation. A custom show is a defined slide sequence that allows for the displaying of the slides with the presentation in any arbitrary order.

Parent Elements
presentation (§4.3.1.24)

Child Elements	Subclause
custShow (Custom Show)	§4.3.1.5

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustomShowList">
   <sequence>
      <element name="custShow" type="CT_CustomShow" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

4.3.1.7 defaultTextStyle (Presentation Default Text Style)

This element specifies the default text styles that are to be used within the presentation. The text style defined here can be referenced when inserting a new slide if that slide is not associated with a master slide or if no styling information has been otherwise specified for the text within the presentation slide.

Parent Elements
presentation (§4.3.1.24)

Child Elements	Subclause
defPPr (Default Paragraph Style)	§5.1.5.2.2
extLst (Extension List)	§5.1.2.1.15
lvl1pPr (List Level 1 Text Style)	§5.1.5.4.13
lvl2pPr (List Level 2 Text Style)	§5.1.5.4.14
lvl3pPr (List Level 3 Text Style)	§5.1.5.4.15
lvl4pPr (List Level 4 Text Style)	§5.1.5.4.16
lvl5pPr (List Level 5 Text Style)	§5.1.5.4.17
lvl6pPr (List Level 6 Text Style)	§5.1.5.4.18
lvl7pPr (List Level 7 Text Style)	§5.1.5.4.19
lvl8pPr (List Level 8 Text Style)	§5.1.5.4.20
lvl9pPr (List Level 9 Text Style)	§5.1.5.4.21

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextListStyle">
   <sequence>
      <element name="defPPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl1pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl2pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl3pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl4pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl5pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl6pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl7pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl8pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl9pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.3.1.8 embeddedFont (Embedded Font)

This element specifies an embedded font. Once specified, this font is available for use within the presentation. Within a font specification there can be regular, bold, italic and boldItalic versions of the font specified. The actual font data for each of these is referenced using a relationships file that contains links to all available fonts. This font data contains font information for each of the characters to be made available in each version of the font.

[*Example*: Consider the following embedded font.

```
<p:embeddedFont>
  <p:font typeface="MyFont" pitchFamily="34" charset="0"/>
  <p:regular r:id="rId2"/>
</p:embeddedFont>
```

end example]

[*Note*: Not all characters for a typeface must be stored. It is up to the generating application to determine which characters are to be stored in the corresponding font data files. *end note*]

Parent Elements
embeddedFontLst (§4.3.1.9)

Child Elements	Subclause
bold (Bold Embedded Font)	§4.3.1.1
boldItalic (Bold Italic Embedded Font)	§4.3.1.2
font (Embedded Font Name)	§4.3.1.10
italic (Italic Embedded Font)	§4.3.1.14

Child Elements	Subclause
regular (Regular Embedded Font)	§4.3.1.27

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EmbeddedFontListEntry">
   <sequence>
      <element name="font" type="a:CT_TextFont" minOccurs="1" maxOccurs="1"/>
      <element name="regular" type="CT_EmbeddedFontDataId" minOccurs="0" maxOccurs="1"/>
      <element name="bold" type="CT_EmbeddedFontDataId" minOccurs="0" maxOccurs="1"/>
      <element name="italic" type="CT_EmbeddedFontDataId" minOccurs="0" maxOccurs="1"/>
      <element name="boldItalic" type="CT_EmbeddedFontDataId" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.3.1.9 embeddedFontLst (Embedded Font List)

This element specifies a list of fonts that are embedded within the corresponding presentation. The font data for these fonts is stored alongside the other document parts within the document container. The actual font data is referenced within the embeddedFont element.

Parent Elements
presentation (§4.3.1.24)

Child Elements	Subclause
embeddedFont (Embedded Font)	§4.3.1.8

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EmbeddedFontList">
   <sequence>
      <element name="embeddedFont" type="CT_EmbeddedFontListEntry" minOccurs="0"
        maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

4.3.1.10 font (Embedded Font Name)

This element specifies specific properties describing an embedded font. Once specified, this font is available for use within the presentation. Within a font specification there can be regular, bold, italic and boldItalic versions of the font specified. The actual font data for each of these is referenced using a relationships file that contains links to all available fonts. This font data contains font information for each of the characters to be made available in each version of the font.

[*Example*: Consider the following embedded font.

```
<p:embeddedFont>
  <p:font typeface="MyFont" pitchFamily="34" charset="0"/>
```

```
    <p:regular r:id="rId2"/>
  </p:embeddedFont>
```

end example]

[*Note*: Not all characters for a typeface must be stored. It is up to the generating application to determine which characters are to be stored in the corresponding font data files. *end note*]

Parent Elements
embeddedFont (§4.3.1.8)

Attributes	Description
charset (Similar Character Set) Namespace: .../drawingml/2006/main	Specifies the most similar character set to the one being used. This is useful if the generating application cannot use the current font and must choose a similar one. The possible values for this attribute are defined by the XML Schema byte datatype.
panose (Panose Setting) Namespace: .../drawingml/2006/main	Specifies the panose standard setting that will be used to determine the closest matching font by any generating application that employs this method. The possible values for this attribute are defined by the ST_Panose simple type (§5.1.12.37).
pitchFamily (Similar Font Family) Namespace: .../drawingml/2006/main	Specifies the most similar font family to the one being used. This is useful if the generating application cannot use the current font and must choose a similar one. The possible values for this attribute are defined by the XML Schema byte datatype.
typeface (Text Typeface) Namespace: .../drawingml/2006/main	Specifies the typeface, or name of the font that is to be used for a bullet. The typeface used here should be selected from the font list of the generating application. The possible values for this attribute are defined by the ST_TextTypeface simple type (§5.1.12.81).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextFont">
   <attribute name="typeface" type="ST_TextTypeface"/>
   <attribute name="panose" type="ST_Panose" use="optional"/>
   <attribute name="pitchFamily" type="xsd:byte" use="optional" default="0"/>
   <attribute name="charset" type="xsd:byte" use="optional" default="1"/>
</complexType>
```

4.3.1.11 handoutMasterId (Handout Master ID)

This element specifies a handout master that is available within the corresponding presentation. A handout master is a slide that is specifically designed for printing as a handout.

[*Example*: Consider the following specification of a handout master within a presentation

```
<p:presentation xmlns:a="" xmlns:r="" xmlns:p="" embedTrueTypeFonts="1">
  ..
  <p:handoutMasterIdLst>
    <p:handoutMasterId r:id="rId8"/>
  </p:handoutMasterIdLst>
  ..
</p:presentation>
```

end example]

Parent Elements
handoutMasterIdLst (§4.3.1.12)

Child Elements	Subclause
extLst (Extension List)	§4.2.5

Attributes	Description
id (Relationship Identifier) Namespace: .../officeDocument/2006/relationships	Specifies the relationship identifier that is used in conjunction with a corresponding relationship file to resolve the location within a presentation of the handoutMaster element defining this handout master. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HandoutMasterIdListEntry">
   <sequence>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute ref="r:id" use="required"/>
</complexType>
```

4.3.1.12 handoutMasterIdLst (List of Handout Master IDs)

This element specifies a list of identification information for the handout master slides that are available within the corresponding presentation. A handout master is a slide that is specifically designed for printing as a handout.

Parent Elements
presentation (§4.3.1.24)

Child Elements	Subclause
handoutMasterId (Handout Master ID)	§4.3.1.11

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HandoutMasterIdList">
    <sequence>
        <element name="handoutMasterId" type="CT_HandoutMasterIdListEntry" minOccurs="0"
          maxOccurs="1"/>
    </sequence>
</complexType>
```

4.3.1.13 htmlPubPr (HTML Publishing Properties)

This element specifies the publishing properties to be used when publishing this presentation document to the HTML file format.

Parent Elements
presentationPr (§4.3.1.25)

Child Elements	Subclause
custShow (Custom Show)	§4.2.2
extLst (Extension List)	§4.2.5
sldAll (All Slides)	§4.2.7
sldRg (Slide Range)	§4.2.8

Attributes	Description
id (Publish Path) Namespace: .../officeDocument/2006/relationships	Specifies the path that should be used when publishing. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
pubBrowser (Browser Support Target)	Specifies the web browser support that this publishing should be optimized for. The possible values for this attribute are defined by the ST_HtmlPublishWebBrowserSupport simple type (§4.8.6).
showSpeakerNotes (Show Speaker	Specifies whether to show speaker notes when publishing.

Attributes	Description
Notes)	The possible values for this attribute are defined by the XML Schema boolean datatype.
title (HTML Output Title)	Specifies a title for the HTML output file. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HtmlPublishProperties">
   <sequence>
      <group ref="EG_SlideListChoice" minOccurs="1" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="showSpeakerNotes" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="pubBrowser" type="ST_HtmlPublishWebBrowserSupport" use="optional"
     default="v3v4"/>
   <attribute name="title" type="xsd:string" use="optional" default=""/>
   <attribute ref="r:id" use="required"/>
</complexType>
```

4.3.1.14 italic (Italic Embedded Font)

This element specifies an italic embedded font that is linked to a parent typeface. Once specified, this italic version of the given typeface name is available for use within the presentation. The actual font data is referenced using a relationships file that contains links to all fonts available. This font data contains font information for each of the characters to be made available.

[*Example*: Consider the following embedded font with a italic version specified.

```
<p:embeddedFont>
  <p:font typeface="MyFont" pitchFamily="34" charset="0"/>
  <p:italic r:id="rId2"/>
</p:embeddedFont>
```

end example]

[*Note*: Not all characters for a typeface must be stored. It is up to the generating application to determine which characters are to be stored in the corresponding font data files. *end note*]

Parent Elements
embeddedFont (§4.3.1.8)

Attributes	Description
id (Relationship Identifier)	Specifies the relationship identifier that is used in conjunction with a corresponding relationship file to resolve the location of this embedded font that is referenced in a presentation.

Attributes	Description
Namespace: .../officeDocument/2006/relationships	The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EmbeddedFontDataId">
   <attribute ref="r:id" use="required"/>
</complexType>
```

4.3.1.15 kinsoku (Kinsoku Settings)

This element specifies the presentation-wide kinsoku settings that define the line breaking behaviour of East Asian text within the corresponding presentation.

Parent Elements
presentation (§4.3.1.24)

Attributes	Description
invalEndChars (Invalid Kinsoku End Characters)	Specifies the characters that are not valid for ending a line of text with. The possible values for this attribute are defined by the XML Schema string datatype.
invalStChars (Invalid Kinsoku Start Characters)	Specifies the characters that are not valid for starting a line of text with. The possible values for this attribute are defined by the XML Schema string datatype.
lang (Language)	Specifies the corresponding East Asian language that these settings apply to. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Kinsoku">
   <attribute name="lang" type="xsd:string" use="optional"/>
   <attribute name="invalStChars" type="xsd:string" use="required"/>
   <attribute name="invalEndChars" type="xsd:string" use="required"/>
</complexType>
```

4.3.1.16 kiosk (Kiosk Slide Show Mode)

This element specifies that the presentation slide show should be viewed in a full-screen kiosk mode. A presentation viewed in kiosk mode should have user input disabled and will restart after a specified interval.

[*Example*: Consider the following presentation that is set to be viewed in a looping kiosk mode.

```
<p:presentationPr xmlns:a="" xmlns:r="" xmlns:p="">
  <p:showPr loop="1" showNarration="1">
```

```
      ..
      <p:kiosk/>
      ..
    </p:showPr>
  </p:presentationPr>
```

end example]

Parent Elements
showPr (§4.3.1.28)

Attributes	Description
restart (Restart Show)	Specifies the time length that the presentation should run until it is to be restarted. That is, the presentation should loop back to the first slide specified in the presentation or custom show. This value is specified in 1/1000ths of a second and measured from the most recent time the presentation started or restarted. [*Note*: The counter is reset when a presentation is restarted due to automatic looping at the end of a show, if specified by the loop attribute of showPr. *end note*] The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShowInfoKiosk">
   <attribute name="restart" type="xsd:unsignedInt" use="optional" default="300000"/>
</complexType>
```

4.3.1.17 modifyVerifier (Modification Verifier)

This element specifies the write protection settings which have been applied to a PresentationML document. *Write protection* refers to a mode in which the document's contents cannot be modified, and the document cannot be resaved using the same file name.

When present, the application shall require a password to enable modifications to the document. If the supplied password does not match the hash value in this attribute, then write protection shall be enabled. If this element is omitted, then no write protection shall be applied to the current document. Since this protection does not encrypt the document, malicious applications may circumvent its use.

The password supplied to the algorithm is to be a Unicode string; strings longer than 255 characters are truncated to 255 characters. The attributes of this element specify the algorithm to be used to verify the password provided by the user.

[*Example*: Consider a PresentationML document that can only be opened in a write protected state unless a password is provided, in which case the file would be opened in an editable state. This requirement would be specified using the following PresentationML:

```
<p:documentProtection …
  p:cryptAlgorithmClass="hash" p:cryptAlgorithmType="typeAny"
  p:cryptAlgorithmSid="1" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" ... />
```

In order for the hosting application to enable edits to the document, the hosting application would have to be provided with a password that the hosting application would then hash using the algorithm specified by the algorithm attributes and compare to the value of the hashData attribute (`9oN7nWkCAyEZib1RomSJTjmPpCY=`). If the two values matched, the file would be opened in an editable state. *end example]*

Parent Elements
presentation (§4.3.1.24)

Attributes	Description
algIdExt (Cryptographic Algorithm Extensibility)	Specifies that a cryptographic algorithm which was not defined by this Office Open XML Standard has been used to generate the hash value stored with this document. This value, when present, shall be interpreted based on the value of the algIdExtSource attribute in order to determine the algorithm used, which shall be application-defined. [*Rationale*: This extensibility affords the fact that with exponentially increasing computing power, documents created in the future will likely need to utilize as yet undefined hashing algorithms in order to remain secure. *end rationale*] If this value is present, the cryptAlgorithmClass, cryptAlgorithmType, and cryptAlgorithmSid attribute values shall be ignored in favor of the algorithm defined by this attribute. [*Example*: Consider a PresentationML document with the following information stored in its protection element: `<p:… p:algIdExt="0000000A" p:algIdExtSource="futureCryptography" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />` The algIdExt attribute value of `0000000A` specifies that the algorithm with hex code `A` shall be used as defined by the `futureCryptography` application. *end example*] The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
algIdExtSource (Algorithm Extensibility Source)	Specifies the application which defined the algorithm value specified by the algIdExt attribute.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>[Example: Consider a PresentationML document with the following information stored in one its protection element:

<code><p:… p:algIdExt="0000000A"
p:algIdExtSource="futureCryptography"
p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" /></code>

The algIdExtSource attribute value of <code>futureCryptography</code> specifies that the algorithm used here was published by the <code>futureCryptography</code> application. end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>cryptAlgorithmClass (Cryptographic Algorithm Class)</td><td>Specifies the class of cryptographic algorithm used by this protection. [Note: The initial version of this Office Open XML Standard only supports a single version - <code>hash</code> - but future versions may expand this as necessary. end note]

[Example: Consider a PresentationML document with the following information stored in its protection element:

<code><p:… p:cryptAlgorithmClass="hash"
p:cryptAlgorithmType="typeAny"
p:cryptAlgorithmSid="1"
p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" /></code>

The cryptAlgorithmClass attribute value of <code>hash</code> specifies that the algorithm used for the password is a hashing algorithm. end example]

The possible values for this attribute are defined by the ST_AlgClass simple type (§4.8.1).</td></tr>
<tr><td>cryptAlgorithmSid (Cryptographic Hashing Algorithm)</td><td>Specifies the specific cryptographic hashing algorithm which shall be used along with the saltData attribute and user-supplied password in order to compute a hash value for comparison.

The possible values for this attribute shall be interpreted as follows:
<table>
<tr><th>Value</th><th>Algorithm</th></tr>
<tr><td>1</td><td>MD2</td></tr>
<tr><td>2</td><td>MD4</td></tr>
<tr><td>3</td><td>MD5</td></tr>
<tr><td>4</td><td>SHA-1</td></tr>
<tr><td>5</td><td>MAC</td></tr>
<tr><td>6</td><td>RIPEMD</td></tr>
<tr><td>7</td><td>RIPEMD-160</td></tr>
<tr><td>8</td><td>Undefined. Shall not be used.</td></tr>
<tr><td>9</td><td>HMAC</td></tr>
</table>
</td></tr>
</table>

Attributes	Description
	<table><tr><td>10</td><td>Undefined. Shall not be used.</td></tr><tr><td>11</td><td>Undefined. Shall not be used.</td></tr><tr><td>12</td><td>SHA-256</td></tr><tr><td>13</td><td>SHA-384</td></tr><tr><td>14</td><td>SHA-512</td></tr><tr><td>Any other value</td><td>Undefined. Shall not be used.</td></tr></table> [*Example*: Consider a PresentationML document with the following information stored in its protection element: `<p:… p:cryptAlgorithmClass="hash" p:cryptAlgorithmType="typeAny" p:cryptAlgorithmSid="1" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />` The cryptAlgorithmSid attribute value of 1 specifies that the SHA-1 hashing algorithm shall be used to generate a hash from the user-defined password. *end example*] The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
cryptAlgorithmType (Cryptographic Algorithm Type)	Specifies the type of cryptographic algorithm used by this protection. [*Note*: The initial version of this Office Open XML Standard only supports a single type - typeAny - but future versions may expand this as necessary. *end note*] [*Example*: Consider a PresentationML document with the following information stored in its protection element: `<p:… p:cryptAlgorithmClass="hash" p:cryptAlgorithmType="typeAny" p:cryptAlgorithmSid="1" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />` The cryptAlgorithmType attribute value of typeAny specifies that any type of algorithm may have been used for the password. *end example*] The possible values for this attribute are defined by the ST_AlgType simple type (§4.8.2).
cryptProvider (Cryptographic Provider)	Specifies the cryptographic provider which was used to generate the hash value stored in this document. If the user provided a cryptographic provider which was not the system's built-in provider, then that provider shall be stored here so it can subsequently be used if available. If this attribute is omitted, then the built-in cryptographic provider on the system shall be

Attributes	Description
	used. [*Example*: Consider a PresentationML document with the following information stored in its protection element: `<p:… p:cryptProvider="Krista'sProvider" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />` The cryptProvider attribute value of `Krista'sProvider` specifies that the cryptographic provider with name "Krista's Provider" shall be used if available. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
cryptProviderType (Cryptographic Provider Type)	Specifies the type of cryptographic provider to be used. [*Example*: Consider a PresentationML document with the following information stored in its protection element: `<p:… p:cryptProviderType="rsaAES" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />` The cryptProviderType attribute value of `rsaAES` specifies that the cryptographic provider type shall be an Advanced Encryption Standard provider. *end example*] The possible values for this attribute are defined by the ST_CryptProv simple type (§4.8.4).
cryptProviderTypeExt (Cryptographic Provider Type Extensibility)	Specifies that a cryptographic provider type which was not defined by this Office Open XML Standard has been used to generate the hash value stored with this document. This value, when present, shall be interpreted based on the value of the cryptProviderTypeExtSource attribute in order to determine the provider type used, which shall be application-defined. [*Rationale*: This extensibility affords the fact that with exponentially increasing computing power, documents created in the future will likely need to utilize as yet undefined cryptographic provider types in order to remain secure. *end rationale*] If this value is present, the cryptProviderType attribute value shall be ignored in favor of the provider type defined by this attribute. [*Example*: Consider a PresentationML document with the following information stored in its protection element: `<p:… p:cryptProviderTypeExt="00A5691D" p:cryptProvideTypeExtSource="futureCryptography" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />` The cryptProviderTypeExt attribute value of `00A5691D` specifies that the provider type

Attributes	Description
	associated with hex code A5691D shall be used as defined by the futureCryptography application. *end example*] The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
cryptProviderTypeExtSource (Provider Type Extensibility Source)	Specifies the application which defined the provider type value specified by the cryptProviderTypeExt attribute. [*Example*: Consider a PresentationML document with the following information stored in its protection element: `<p:… p:cryptProviderTypeExt="00A5691D" p:cryptProvideTypeExtSource="futureCryptography" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />` The cryptProvideTypeExtSource attribute value of futureCryptography specifies that the provider type used here was published by the futureCryptography application. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
hashData (Password Hash)	Specifies the hash value for the password stored with this document. This value shall be compared with the resulting hash value after hashing the user-supplied password using the algorithm specified by the preceding attributes and parent XML element, and if the two values match, the protection shall no longer be enforced. If this value is omitted, then no password shall be associated with the protection, and it may be turned off without supplying any password. [*Example*: Consider a PresentationML document with the following information stored in its protection element: `<p:… p:cryptAlgorithmClass="hash" p:cryptAlgorithmType="typeAny" p:cryptAlgorithmSid="1" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />` The hashData attribute value of 9oN7nWkCAyEZib1RomSJTjmPpCY= specifies that the user-supplied password shall be hashed using the pre-processing defined by the parent element (if any) followed by the SHA-1 algorithm (specified via the cryptAlgorithmSid attribute value of 1) and that the resulting has value must be 9oN7nWkCAyEZib1RomSJTjmPpCY= for the protection to be disabled. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
saltData (Salt for	Specifies the salt which was prepended to the user-supplied password before it was

Attributes	Description
Password Verifier)	hashed using the hashing algorithm defined by the preceding attribute values to generate the hashData attribute, and which shall also be prepended to the user-supplied password before attempting to generate a hash value for comparison. A *salt* is a random string which is added to a user-supplied password before it is hashed in order to prevent a malicious party from pre-calculating all possible password/hash combinations and simply using those precalculated values (often referred to as a "dictionary attack"). If this attribute is omitted, then no salt shall be prepended to the user-supplied password before it is hashed for comparison with the stored hash value. [*Example*: Consider a PresentationML document with the following information stored in its protection element: `<p:… p:saltData="ZUdHa+D8F/OAKP3I7ssUnQ==" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />` The saltData attribute value of `ZUdHa+D8F/OAKP3I7ssUnQ==` specifies that the user-supplied password shall have this value prepended before it is run through the specified hashing algorithm to generate a resulting hash value for comparison. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
spinCount (Iterations to Run Hashing Algorithm)	Specifies the number of times the hashing function shall be iteratively run (using each iteration's result as the input for the next iteration) when attempting to compare a user-supplied password with the value stored in the hashData attribute. [*Rationale*: Running the algorithm many times increases the cost of exhaustive search attacks correspondingly. Storing this value allows for the number of iterations to be increased over time to accommodate faster hardware (and hence the ability to run more iterations in less time). *end rationale*] [*Example*: Consider a PresentationML document with the following information stored in its protection element: `<p:… p:spinCount="100000" p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />` The spinCount attribute value of `100000` specifies that the hashing function shall be run one hundred thousand times to generate a hash value for comparison with the hash attribute. *end example*] The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ModifyVerifier">
   <attribute name="cryptProviderType" type="ST_CryptProv" use="required"/>
   <attribute name="cryptAlgorithmClass" type="ST_AlgClass" use="required"/>
   <attribute name="cryptAlgorithmType" type="ST_AlgType" use="required"/>
   <attribute name="cryptAlgorithmSid" type="xsd:unsignedInt" use="required"/>
   <attribute name="spinCount" type="xsd:unsignedInt" use="required"/>
   <attribute name="saltData" type="xsd:string" use="required"/>
   <attribute name="hashData" type="xsd:string" use="required"/>
   <attribute name="cryptProvider" type="xsd:string" use="optional"/>
   <attribute name="algIdExt" type="xsd:unsignedInt" use="optional"/>
   <attribute name="algIdExtSource" type="xsd:string" use="optional"/>
   <attribute name="cryptProviderTypeExt" type="xsd:unsignedInt" use="optional"/>
   <attribute name="cryptProviderTypeExtSource" type="xsd:string" use="optional"/>
</complexType>
```

4.3.1.18 notesMasterId (Notes Master ID)

This element specifies a notes master that is available within the corresponding presentation. A notes master is a slide that is specifically designed for the printing of the slide along with any attached notes.

[*Example*: Consider the following specification of a notes master within a presentation

```
<p:presentation xmlns:a="" xmlns:r="" xmlns:p="" embedTrueTypeFonts="1">
  ..
  <p:notesMasterIdLst>
    <p:notesMasterId r:id="rId8"/>
  </p:notesMasterIdLst>
  ..
</p:presentation>
```

end example]

Parent Elements
notesMasterIdLst (§4.3.1.19)

Child Elements	Subclause
extLst (Extension List)	§4.2.5

Attributes	Description
id (Relationship Identifier) Namespace: .../officeDocument	Specifies the relationship identifier that is used in conjunction with a corresponding relationship file to resolve the location within a presentation of the notesMaster element defining this notes master. The possible values for this attribute are defined by the ST_RelationshipId simple type

Attributes	Description
/2006/relationships	(§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NotesMasterIdListEntry">
   <sequence>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute ref="r:id" use="required"/>
</complexType>
```

4.3.1.19 notesMasterIdLst (List of Notes Master IDs)

This element specifies a list of identification information for the notes master slides that are available within the corresponding presentation. A notes master is a slide that is specifically designed for the printing of the slide along with any attached notes.

Parent Elements
presentation (§4.3.1.24)

Child Elements	Subclause
notesMasterId (Notes Master ID)	§4.3.1.18

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NotesMasterIdList">
   <sequence>
      <element name="notesMasterId" type="CT_NotesMasterIdListEntry" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.3.1.20 notesSz (Notes Slide Size)

This element specifies the size of slide surface used for notes slides and handout slides. Objects within a notes slide can be specified outside these extents, but the notes slide will have a background surface of the specified size when presented or printed. This element is intended to specify the region to which content is fitted in any special type of printout the application may choose to generate, such as an outline handout.

[*Example*: Consider the following specifying of the size of a notes slide.

```
<p:presentation xmlns:a="" xmlns:r="" xmlns:p="" embedTrueTypeFonts="1">
  ..
  <p:notesSz cx="9144000" cy="6858000"/>
  ..
</p:presentation>
```

end example]

Parent Elements
presentation (§4.3.1.24)

Attributes	Description
cx (Extent Length) Namespace: .../drawingml/2006/main	Specifies the length of the extents rectangle in EMUs. This rectangle shall dictate the size of the object as displayed (the result of any scaling to the original object). [*Example*: Consider a DrawingML object specified as follows: `<… cx="1828800" cy="200000"/>` The cx attributes specifies that this object has a height of 1828800 EMUs (English Metric Units). *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).
cy (Extent Width) Namespace: .../drawingml/2006/main	Specifies the width of the extents rectangle in EMUs. This rectangle shall dictate the size of the object as displayed (the result of any scaling to the original object). [*Example*: Consider a DrawingML object specified as follows: `< … cx="1828800" cy="200000"/>` The cy attribute specifies that this object has a width of 200000 EMUs (English Metric Units). *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PositiveSize2D">
   <attribute name="cx" type="ST_PositiveCoordinate" use="required"/>
   <attribute name="cy" type="ST_PositiveCoordinate" use="required"/>
</complexType>
```

4.3.1.21 penClr (Pen Color for Slide Show)

This element specifies the pen color that should be used to make markings on the slides while in a presentation.

Parent Elements
showPr (§4.3.1.28)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

4.3.1.22 photoAlbum (Photo Album Information)

This element specifies that the corresponding presentation contains a photo album. A photo album specifies a list of images within the presentation that spread across one or more slides, all of which share a consistent layout. Each image in the album is formatted with a consistent style. This functionality enables the application to manage all of the images together and modify their ordering, layout, and formatting as a set.

This element does not enforce the specified properties on individual photo album images; rather, it specifies common settings that should be applied by default to all photo album images and their containing slides. Images that are part of the photo album are identified by the presence of the isPhoto element in the definition of the picture.

[*Example*: Consider the following presentation that has been specified as a photo album

```
<p:presentation xmlns:a="" xmlns:r="" xmlns:p="" embedTrueTypeFonts="1">
  ..
  <p:photoAlbum bw="1" layout="2pic"/>
  ..
</p:presentation>
```

end example]

Parent Elements
presentation (§4.3.1.24)

Child Elements	Subclause
extLst (Extension List)	§4.2.5

Attributes	Description
bw (Black and White)	Specifies whether all pictures in the photo album are to be displayed as black and white. The possible values for this attribute are defined by the XML Schema boolean datatype.
frame (Frame Type)	Specifies the frame type that is to be used on all the pictures in the photo album. The possible values for this attribute are defined by the ST_PhotoAlbumFrameShape simple type (§4.8.11).
layout (Photo Album Layout)	Specifies the layout that is to be used to arrange the pictures in the photo album on individual slides. The possible values for this attribute are defined by the ST_PhotoAlbumLayout simple type (§4.8.12).
showCaptions (Show/Hide Captions)	Specifies whether to show captions for pictures in the photo album. Captions are text boxes grouped with each image, with the group set to not allow ungrouping. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PhotoAlbum">
   <sequence>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="bw" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="showCaptions" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="layout" type="ST_PhotoAlbumLayout" use="optional" default="fitToSlide"/>
   <attribute name="frame" type="ST_PhotoAlbumFrameShape" use="optional" default="frameStyle1"/>
</complexType>
```

4.3.1.23 present (Presenter Slide Show Mode)

This element specifies that the presentation slide show should be viewed in a full-screen presenter mode. In this mode, the presentation is displayed on one monitor while a different monitor displays notes and provides navigation controls intended to be viewed only by the presenter.

[*Example*: Consider the following presentation that is set to be viewed in a present mode.

```
<p:presentationPr xmlns:a="" xmlns:r="" xmlns:p="">
  <p:showPr>
    ..
    <p:present/>
    ..
  </p:showPr>
</p:presentationPr>
```

end example]

Parent Elements
showPr (§4.3.1.28)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

4.3.1.24 presentation (Presentation)

This element specifies within it fundamental presentation-wide properties.

[*Example*: Consider the following presentation with a single slide master and two slides. In addition to these commonly used elements there can also be the specification of other properties such as slide size, notes size and default text styles.

```
<p:presentation xmlns:a="" xmlns:r="" xmlns:p="">
    <p:sldMasterIdLst>
        <p:sldMasterId id="2147483648" r:id="rId1"/>
    </p:sldMasterIdLst>
    <p:sldIdLst>
        <p:sldId id="256" r:id="rId3"/>
        <p:sldId id="257" r:id="rId4"/>
    </p:sldIdLst>
    <p:sldSz cx="9144000" cy="6858000" type="screen4x3"/>
    <p:notesSz cx="6858000" cy="9144000"/>
    <p:defaultTextStyle>
      ...
    </p:defaultTextStyle>
</p:presentation>
```

end example]

Parent Elements
Root element of PresentationML Presentation part

Child Elements	Subclause
custDataLst (Customer Data List)	§4.4.1.16
custShowLst (List of Custom Shows)	§4.3.1.6
defaultTextStyle (Presentation Default Text Style)	§4.3.1.7
embeddedFontLst (Embedded Font List)	§4.3.1.9
extLst (Extension List)	§4.2.5
handoutMasterIdLst (List of Handout Master IDs)	§4.3.1.12
kinsoku (Kinsoku Settings)	§4.3.1.15

Child Elements	Subclause
modifyVerifier (Modification Verifier)	§4.3.1.17
notesMasterIdLst (List of Notes Master IDs)	§4.3.1.19
notesSz (Notes Slide Size)	§4.3.1.20
photoAlbum (Photo Album Information)	§4.3.1.22
sldIdLst (List of Slide IDs)	§4.3.1.30
sldMasterIdLst (List of Slide Master IDs)	§4.3.1.33
sldSz (Presentation Slide Size)	§4.3.1.34
smartTags (Smart Tags)	§4.3.1.35

Attributes	Description
autoCompressPictures (Automatically Compress Pictures)	Specifies whether the generating application should automatically compress all pictures for this presentation. The possible values for this attribute are defined by the XML Schema boolean datatype.
bookmarkIdSeed (Bookmark ID Seed)	Specifies a seed for generating bookmark IDs to ensure IDs remain unique across the document. This value specifies the number to be used as the ID for the next new bookmark created. The possible values for this attribute are defined by the ST_BookmarkIdSeed simple type (§4.8.3).
compatMode (Compatibility Mode)	Specifies whether the generating application is to be in a compatibility mode which serves to inform the user of any loss of content or functionality when working with older formats. The possible values for this attribute are defined by the XML Schema boolean datatype.
embedTrueTypeFonts (Embed True Type Fonts)	Specifies whether the generating application should automatically embed true type fonts or not. The possible values for this attribute are defined by the XML Schema boolean datatype.
firstSlideNum (First Slide Number)	Specifies the first slide number in the presentation. The possible values for this attribute are defined by the XML Schema int datatype.
removePersonalInfoOnSave (Remove Personal Information on Save)	Specifies whether to automatically remove personal information when the presentation document is saved. The possible values for this attribute are defined by the XML Schema boolean datatype.
rtl (Right-To-Left Views)	Specifies if the current view of the user interface is oriented right-to-left or left-to-right. The view is right-to-left is this value is set to true, and left-to-right otherwise.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema boolean datatype.
saveSubsetFonts (Save Subset Fonts)	Specifies to save only the subset of characters used in the presentation when a font is embedded. The possible values for this attribute are defined by the XML Schema boolean datatype.
serverZoom (Server Zoom)	Specifies the scaling to be used when the presentation is embedded in another document. The embedded slides are to be scaled by this percentage. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
showSpecialPlsOnTitleSld (Show Header and Footer Placeholders on Titles)	Specifies whether to show the header and footer placeholders on the title slides. The possible values for this attribute are defined by the XML Schema boolean datatype.
strictFirstAndLastChars (Strict First and Last Characters)	Specifies whether to use strict characters for starting and ending lines of Japanese text. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Presentation">
   <sequence>
      <element name="sldMasterIdLst" type="CT_SlideMasterIdList" minOccurs="0" maxOccurs="1"/>
      <element name="notesMasterIdLst" type="CT_NotesMasterIdList" minOccurs="0" maxOccurs="1"/>
      <element name="handoutMasterIdLst" type="CT_HandoutMasterIdList" minOccurs="0" maxOccurs="1"/>
      <element name="sldIdLst" type="CT_SlideIdList" minOccurs="0" maxOccurs="1"/>
      <element name="sldSz" type="CT_SlideSize" minOccurs="0" maxOccurs="1"/>
      <element name="notesSz" type="a:CT_PositiveSize2D" minOccurs="1" maxOccurs="1"/>
      <element name="smartTags" type="CT_SmartTags" minOccurs="0" maxOccurs="1"/>
      <element name="embeddedFontLst" type="CT_EmbeddedFontList" minOccurs="0" maxOccurs="1"/>
      <element name="custShowLst" type="CT_CustomShowList" minOccurs="0" maxOccurs="1"/>
      <element name="photoAlbum" type="CT_PhotoAlbum" minOccurs="0" maxOccurs="1"/>
      <element name="custDataLst" type="CT_CustomerDataList" minOccurs="0" maxOccurs="1"/>
      <element name="kinsoku" type="CT_Kinsoku" minOccurs="0"/>
      <element name="defaultTextStyle" type="a:CT_TextListStyle" minOccurs="0" maxOccurs="1"/>
      <element name="modifyVerifier" type="CT_ModifyVerifier" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="serverZoom" type="a:ST_Percentage" use="optional" default="50000"/>
   <attribute name="firstSlideNum" type="xsd:int" use="optional" default="1"/>
   <attribute name="showSpecialPlsOnTitleSld" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="rtl" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="removePersonalInfoOnSave" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="compatMode" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="strictFirstAndLastChars" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="embedTrueTypeFonts" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="saveSubsetFonts" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="autoCompressPictures" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="bookmarkIdSeed" type="ST_BookmarkIdSeed" use="optional" default="1"/>
</complexType>
```

4.3.1.25 presentationPr (Presentation-wide Properties)

This element functions as a parent element within which additional presentation-wide document properties are contained. All properties and their corresponding settings are defined within the child elements.

Parent Elements
Root element of PresentationML Presentation Properties part

Child Elements	Subclause
clrMru (Color MRU)	§4.3.1.4
extLst (Extension List)	§4.2.5
htmlPubPr (HTML Publishing Properties)	§4.3.1.13
prnPr (Printing Properties)	§4.3.1.26
showPr (Presentation-wide Show Properties)	§4.3.1.28
webPr (Web Properties)	§4.3.1.36

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PresentationProperties">
   <sequence>
      <element name="htmlPubPr" type="CT_HtmlPublishProperties" minOccurs="0" maxOccurs="1"/>
      <element name="webPr" type="CT_WebProperties" minOccurs="0" maxOccurs="1"/>
      <element name="prnPr" type="CT_PrintProperties" minOccurs="0" maxOccurs="1"/>
      <element name="showPr" type="CT_ShowProperties" minOccurs="0" maxOccurs="1"/>
      <element name="clrMru" type="a:CT_ColorMRU" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.3.1.26 prnPr (Printing Properties)

This element specifies the default printing properties associated with this presentation document.

Parent Elements
presentationPr (§4.3.1.25)

Child Elements	Subclause
extLst (Extension List)	§4.2.5

Attributes	Description
clrMode (Print Color Mode)	Specifies the color mode to be used when printing. The possible values for this attribute are defined by the ST_PrintColorMode simple type (§4.8.15).
frameSlides (Frame slides when printing)	Specifies whether slides should be framed when printing. When framed, an outline border is printed for each slide. The possible values for this attribute are defined by the XML Schema boolean datatype.
hiddenSlides (Print Hidden Slides)	Specifies whether hidden slides should be printed. The possible values for this attribute are defined by the XML Schema boolean datatype.
prnWhat (Print Output)	Specifies what the default print output will be in terms of content layout. The possible values for this attribute are defined by the ST_PrintWhat simple type (§4.8.16).
scaleToFitPaper (Scale to Fit Paper when printing)	Specifies whether the print output should be scaled to fit the paper being used. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PrintProperties">
   <sequence>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="prnWhat" type="ST_PrintWhat" use="optional" default="slides"/>
   <attribute name="clrMode" type="ST_PrintColorMode" use="optional" default="clr"/>
   <attribute name="hiddenSlides" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="scaleToFitPaper" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="frameSlides" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.3.1.27 regular (Regular Embedded Font)

This element specifies a regular embedded font that is linked to a parent typeface. Once specified, this regular version of the given typeface name is available for use within the presentation. The actual font data is referenced using a relationships file that contains links to all fonts available. This font data contains font information for each of the characters to be made available.

[*Example*: Consider the following embedded font with a regular version specified.

```
<p:embeddedFont>
  <p:font typeface="MyFont" pitchFamily="34" charset="0"/>
  <p:regular r:id="rId2"/>
</p:embeddedFont>
```

end example]

[*Note*: Not all characters for a typeface must be stored. It is up to the generating application to determine which characters are to be stored in the corresponding font data files. *end note*]

Parent Elements
embeddedFont (§4.3.1.8)

Attributes	Description
id (Relationship Identifier) Namespace: .../officeDocument/2006/relationships	Specifies the relationship identifier that is used in conjunction with a corresponding relationship file to resolve the location of this embedded font that is referenced in a presentation. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EmbeddedFontDataId">
   <attribute ref="r:id" use="required"/>
</complexType>
```

4.3.1.28 showPr (Presentation-wide Show Properties)

This element functions as a parent element within which all presentation-wide show properties are contained. All properties and their corresponding settings are defined within the child elements.

Parent Elements
presentationPr (§4.3.1.25)

Child Elements	Subclause
browse (Browse Slide Show Mode)	§4.3.1.3
custShow (Custom Show)	§4.2.2
extLst (Extension List)	§4.2.5
kiosk (Kiosk Slide Show Mode)	§4.3.1.16
penClr (Pen Color for Slide Show)	§4.3.1.21
present (Presenter Slide Show Mode)	§4.3.1.23
sldAll (All Slides)	§4.2.7
sldRg (Slide Range)	§4.2.8

Attributes	Description
loop (Loop Slide Show)	Specifies whether the slide show should be set to loop at the end. The possible values for this attribute are defined by the XML Schema boolean datatype.
showAnimation (Show Animation in Slide Show)	Specifies whether slide show animation should be shown when presenting. The possible values for this attribute are defined by the XML Schema boolean datatype.
showNarration (Show Narration in Slide Show)	Specifies whether slide show narration should be played when presenting. The possible values for this attribute are defined by the XML Schema boolean datatype.
useTimings (Use Timings in Slide Show)	Specifies whether slide transition timings should be used to advance slides when presenting. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShowProperties">
   <sequence minOccurs="0" maxOccurs="1">
      <group ref="EG_ShowType" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_SlideListChoice" minOccurs="0" maxOccurs="1"/>
      <element name="penClr" type="a:CT_Color" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="loop" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="showNarration" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="showAnimation" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="useTimings" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

4.3.1.29 sldId (Slide ID)

This element specifies a presentation slide that is available within the corresponding presentation. A slide contains the information that is specific to a single slide such as slide-specific shape and text information.

[*Example*: Consider the following specification of a slide master within a presentation

```
<p:presentation xmlns:a="" xmlns:r="" xmlns:p="" embedTrueTypeFonts="1">
  ..
  <p:sldIdLst>
    <p:sldId id="256" r:id="rId3"/>
    <p:sldId id="257" r:id="rId4"/>
    <p:sldId id="258" r:id="rId5"/>
    <p:sldId id="259" r:id="rId6"/>
    <p:sldId id="260" r:id="rId7"/>
  </p:sldIdLst>
  ..
</p:presentation>
```

end example]

Parent Elements
sldIdLst (§4.3.1.30)

Child Elements	Subclause
extLst (Extension List)	§4.2.5

Attributes	Description
id (Relationship Identifier)	Specifies the relationship identifier that is used in conjunction with a corresponding relationship file to resolve the location within a presentation of the sld element defining this slide.

Attributes	Description
Namespace: .../officeDocument/2006/relationships	The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
id (Slide Identifier)	Specifies the slide identifier that is to contain a value that is unique throughout the presentation. The possible values for this attribute are defined by the ST_SlideId simple type (§4.8.17).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SlideIdListEntry">
   <sequence>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="id" type="ST_SlideId" use="required"/>
   <attribute ref="r:id" use="required"/>
</complexType>
```

4.3.1.30 sldIdLst (List of Slide IDs)

This element specifies a list of identification information for the slides that are available within the corresponding presentation. A slide contains the information that is specific to a single slide such as slide-specific shape and text information.

Parent Elements
presentation (§4.3.1.24)

Child Elements	Subclause
sldId (Slide ID)	§4.3.1.29

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SlideIdList">
   <sequence>
      <element name="sldId" type="CT_SlideIdListEntry" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

4.3.1.31 sldLst (List of Presentation Slides)

This element specifies a list of presentation slides. A presentation slide contains the information that is specific to a single slide such as slide-specific shape and text information.

Parent Elements
custShow (§4.3.1.5)

Child Elements	Subclause
sld (Presentation Slide)	§4.2.6

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SlideRelationshipList">
   <sequence>
      <element name="sld" type="CT_SlideRelationshipListEntry" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

4.3.1.32 sldMasterId (Slide Master ID)

This element specifies a slide master that is available within the corresponding presentation. A slide master is a slide that is specifically designed to be a template for all related child layout slides.

[*Example*: Consider the following specification of a slide master within a presentation

```
<p:presentation xmlns:a="" xmlns:r="" xmlns:p="" embedTrueTypeFonts="1">
  ..
  <p:sldMasterIdLst>
    <p:sldMasterId id="2147483648" r:id="rId1"/>
  </p:sldMasterIdLst>
  ..
</p:presentation>
```

end example]

Parent Elements
sldMasterIdLst (§4.3.1.33)

Child Elements	Subclause
extLst (Extension List)	§4.2.5

Attributes	Description
id (Slide Master Identifier)	Specifies the slide master identifier that is to contain a value that is unique throughout the presentation. The possible values for this attribute are defined by the ST_SlideMasterId simple type (§4.8.20).
id (Relationship Identifier) Namespace:	Specifies the relationship identifier that is used in conjunction with a corresponding relationship file to resolve the location within a presentation of the sldMaster element defining this slide master.

Attributes	Description
.../officeDocument/2006/relationships	The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SlideMasterIdListEntry">
   <sequence>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="id" type="ST_SlideMasterId" use="optional"/>
   <attribute ref="r:id" use="required"/>
</complexType>
```

4.3.1.33 sldMasterIdLst (List of Slide Master IDs)

This element specifies a list of identification information for the slide master slides that are available within the corresponding presentation. A slide master is a slide that is specifically designed to be a template for all related child layout slides.

Parent Elements
presentation (§4.3.1.24)

Child Elements	Subclause
sldMasterId (Slide Master ID)	§4.3.1.32

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SlideMasterIdList">
   <sequence>
      <element name="sldMasterId" type="CT_SlideMasterIdListEntry" minOccurs="0"
        maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

4.3.1.34 sldSz (Presentation Slide Size)

This element specifies the size of the presentation slide surface. Objects within a presentation slide can be specified outside these extents, but this is the size of background surface that will be shown when the slide is presented or printed..

[*Example*: Consider the following specifying of the size of a presentation slide.

```
<p:presentation xmlns:a="" xmlns:r="" xmlns:p="" embedTrueTypeFonts="1">
  ..
  <p:sldSz cx="9144000" cy="6858000" type="screen4x3"/>
  ..
```

```
</p:presentation>
```

end example]

Parent Elements
presentation (§4.3.1.24)

Attributes	Description
cx (Extent Length)	Specifies the length of the extents rectangle in EMUs. This rectangle shall dictate the size of the object as displayed (the result of any scaling to the original object). [*Example*: Consider a DrawingML object specified as follows: `<… cx="1828800" cy="200000"/>` The cx attributes specifies that this object has a height of 1828800 EMUs (English Metric Units). *end example*] The possible values for this attribute are defined by the ST_SlideSizeCoordinate simple type (§4.8.21).
cy (Extent Width)	Specifies the width of the extents rectangle in EMUs. This rectangle shall dictate the size of the object as displayed (the result of any scaling to the original object). [*Example*: Consider a DrawingML object specified as follows: `< … cx="1828800" cy="200000"/>` The cy attribute specifies that this object has a width of 200000 EMUs (English Metric Units). *end example*] The possible values for this attribute are defined by the ST_SlideSizeCoordinate simple type (§4.8.21).
type (Type of Size)	Specifies the type of slide size that should be used. This identifies in particular the expected delivery platform for this presentation. The possible values for this attribute are defined by the ST_SlideSizeType simple type (§4.8.22).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SlideSize">
   <attribute name="cx" type="ST_SlideSizeCoordinate" use="required"/>
   <attribute name="cy" type="ST_SlideSizeCoordinate" use="required"/>
   <attribute name="type" type="ST_SlideSizeType" use="optional" default="custom"/>
</complexType>
```

4.3.1.35 smartTags (Smart Tags)

This element specifies the existence of smart tags within the corresponding presentation.

Parent Elements
presentation (§4.3.1.24)

Attributes	Description
id (Relationship Identifier) Namespace: .../officeDocument/2006/relationships	Specifies the relationship identifier that is used in conjunction with a corresponding relationship file to resolve the location of this smart tag. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SmartTags">
   <attribute ref="r:id" use="required"/>
</complexType>
```

4.3.1.36 webPr (Web Properties)

This element specifies all general output properties that pertain to generating a web format version of the presentation document.

Parent Elements
presentationPr (§4.3.1.25)

Child Elements	Subclause
extLst (Extension List)	§4.2.5

Attributes	Description
allowPng (Allow PNG in HTML output)	Specifies whether to allow the output of PNG format pictures in the HTML document. The possible values for this attribute are defined by the XML Schema boolean datatype.
clr (Slide Navigation Colors for HTML output)	Specifies the color constraints that are to be used when generating HTML output. The possible values for this attribute are defined by the ST_WebColorType simple type (§4.8.60).
encoding (Encoding for HTML output)	Specifies the particular HTML character set encoding that should be used when generating output.

Attributes	Description
	The possible values for this attribute are defined by the ST_WebEncoding simple type (§4.8.61).
imgSz (Image size for HTML output)	Specifies the screen size for which the images in the HTML output should be optimized. The possible values for this attribute are defined by the ST_WebScreenSize simple type (§4.8.62).
organizeInFolders (Organize HTML output in folders)	Specifies whether the supporting output files should be automatically organized into a folder. The possible values for this attribute are defined by the XML Schema boolean datatype.
relyOnVml (Rely on VML for HTML output)	Specifies whether graphics should be output in VML within the HTML. The possible values for this attribute are defined by the XML Schema boolean datatype.
resizeGraphics (Resize graphics in HTML output)	Specifies whether to resize graphics to fit within the browser window when generating the HTML output. The possible values for this attribute are defined by the XML Schema boolean datatype.
showAnimation (Show animation in HTML output)	Specifies whether to show presentation animation in the HTML output file. The possible values for this attribute are defined by the XML Schema boolean datatype.
useLongFilenames (Use long file names in HTML output)	Specifies whether to allow the use of long file names when generating the HTML output. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_WebProperties">
   <sequence>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="showAnimation" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="resizeGraphics" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="allowPng" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="relyOnVml" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="organizeInFolders" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="useLongFilenames" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="imgSz" type="ST_WebScreenSize" use="optional" default="800x600"/>
   <attribute name="encoding" type="ST_WebEncoding" use="optional" default=""/>
   <attribute name="clr" type="ST_WebColorType" use="optional" default="whiteTextOnBlack"/>
</complexType>
```

4.3.2 View Properties

This section contains all properties that pertain to the viewing of the presentation.

4.3.2.1 cSldViewPr (Common Slide View Properties)

This element functions as a container for slide view properties that are common across multiple view property elements. The specific properties and associated values for these view properties reside within the child elements and attributes.

Parent Elements
notesViewPr (§4.3.2.8); slideViewPr (§4.3.2.16)

Child Elements	Subclause
cViewPr (Common View Properties)	§4.3.2.2
guideLst (List of Guides)	§4.3.2.5

Attributes	Description
showGuides (Show Guides in View)	Specifies whether to show guides when editing the presentation. The possible values for this attribute are defined by the XML Schema boolean datatype.
snapToGrid (Snap Objects to Grid)	Specifies whether objects should snap to underlying presentation grid when editing. The possible values for this attribute are defined by the XML Schema boolean datatype.
snapToObjects (Snap Objects to Objects)	Specifies whether objects should snap to other objects when editing the presentation. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CommonSlideViewProperties">
   <sequence>
      <element name="cViewPr" type="CT_CommonViewProperties" minOccurs="1" maxOccurs="1"/>
      <element name="guideLst" type="CT_GuideList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="snapToGrid" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="snapToObjects" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="showGuides" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.3.2.2 cViewPr (Common View Properties)

This element specifies the view properties that are common across multiple view property elements.

Parent Elements
cSldViewPr (§4.3.2.1); notesTextViewPr (§4.3.2.7); outlineViewPr (§4.3.2.10); sorterViewPr (§4.3.2.17)

Child Elements	Subclause
origin (View Origin)	§4.3.2.9
scale (View Scale)	§4.3.2.13

Attributes	Description
varScale (Variable Scale)	Specifies that the view content should automatically scale to best fit the current window size. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CommonViewProperties">
   <sequence>
      <element name="scale" type="a:CT_Scale2D" minOccurs="1" maxOccurs="1"/>
      <element name="origin" type="a:CT_Point2D" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="varScale" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.3.2.3 gridSpacing (Grid Spacing)

This element specifies the grid spacing that should be used for the grid underlying the presentation document. The grid may be used to align objects on the slide and to display visual positioning cues.

Parent Elements
viewPr (§4.3.2.18)

Attributes	Description
cx (Extent Length) Namespace: .../drawingml/2006/main	Specifies the length of the extents rectangle in EMUs. This rectangle shall dictate the size of the object as displayed (the result of any scaling to the original object). [*Example*: Consider a DrawingML object specified as follows: <… cx="1828800" cy="200000"/> The cx attributes specifies that this object has a height of 1828800 EMUs (English Metric Units). *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).
cy (Extent Width) Namespace:	Specifies the width of the extents rectangle in EMUs. This rectangle shall dictate the size of the object as displayed (the result of any scaling to the original object).

Attributes	Description
.../drawingml/2006/main	[*Example*: Consider a DrawingML object specified as follows: `< … cx="1828800" cy="200000"/>` The cy attribute specifies that this object has a width of `200000` EMUs (English Metric Units). *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PositiveSize2D">
   <attribute name="cx" type="ST_PositiveCoordinate" use="required"/>
   <attribute name="cy" type="ST_PositiveCoordinate" use="required"/>
</complexType>
```

4.3.2.4 guide (A Guide)

This element specifies a guide within the presentation. Guides are lines used for arranging layouts and content and never appear except as an aid in editing slides.

Parent Elements
guideLst (§4.3.2.5)

Attributes	Description
orient (Guide Orientation)	Specifies the orientation for a guide. The possible values for this attribute are defined by the ST_Direction simple type (§4.8.5).
pos (Guide Position)	Specifies the position information for a guide. The possible values for this attribute are defined by the ST_Coordinate32 simple type (§5.1.12.17).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Guide">
   <attribute name="orient" type="ST_Direction" use="optional" default="vert"/>
   <attribute name="pos" type="a:ST_Coordinate32" use="optional" default="0"/>
</complexType>
```

4.3.2.5 guideLst (List of Guides)

This element specifies a list of guides for a particular view of the presentation.

Parent Elements

Parent Elements
cSldViewPr (§4.3.2.1)

Child Elements	Subclause
guide (A Guide)	§4.3.2.4

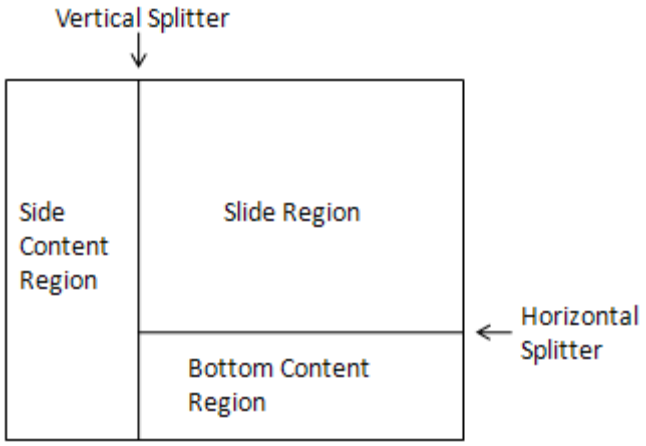
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GuideList">
   <sequence minOccurs="0" maxOccurs="1">
      <element name="guide" type="CT_Guide" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

4.3.2.6 normalViewPr (Normal View Properties)

This element specifies the view properties associated with the normal view mode. The normal view consists of three content regions: the slide itself, a side content region, and a bottom content region. The content of the side content region and bottom content region is determined by the generating application. Properties pertaining to the positioning of the different content regions are stored in this element. This information allows the application to save its view state to the file, so that when reopened the view is in the same state as when the presentation was last saved.

A vertical splitter bar separates the slide from the side content region. A horizontal splitter bar separates the slide from the content region below the slide. If the presentation is set to left-to-right, the side content region is to the left of the slide. If the presentation is set to right-to-left, the side content region is to the right of the slide.



Parent Elements
viewPr (§4.3.2.18)

Child Elements	Subclause
extLst (Extension List)	§4.2.5
restoredLeft (Normal View Restored Left Properties)	§4.3.2.11
restoredTop (Normal View Restored Top Properties)	§4.3.2.12

Attributes	Description
horzBarState (State of the Horizontal Splitter Bar)	Specifies the state that the horizontal splitter bar should be in when in normal view mode. The region to be maximized or minimized is the side content region. The possible values for this attribute are defined by the ST_SplitterBarState simple type (§4.8.23).
preferSingleView (Prefer Single View)	Specifies whether the user prefers to see a full-window single-content region over the standard normal view with three content regions. If enabled, the application may choose to display one of the content regions in the entire window. The possible values for this attribute are defined by the XML Schema boolean datatype.
showOutlineIcons (Show Outline Icons in Normal View)	Specifies whether the application should show icons if displaying outline content in any of the content regions of normal view mode. The possible values for this attribute are defined by the XML Schema boolean datatype.
snapVertSplitter (Snap Vertical Splitter)	Specifies whether the vertical splitter should snap to a minimized state when the side region is sufficiently small. The specific parameters of this behaviour are left to the generating application. The possible values for this attribute are defined by the XML Schema boolean datatype.
vertBarState (State of the Vertical Splitter Bar)	Specifies the state that the vertical splitter bar should be in when in normal view mode. The region to be maximized or minimized is the slide region. The possible values for this attribute are defined by the ST_SplitterBarState simple type (§4.8.23).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NormalViewProperties">
   <sequence>
      <element name="restoredLeft" type="CT_NormalViewPortion" minOccurs="1" maxOccurs="1"/>
      <element name="restoredTop" type="CT_NormalViewPortion" minOccurs="1" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="showOutlineIcons" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="snapVertSplitter" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="vertBarState" type="ST_SplitterBarState" use="optional" default="restored"/>
   <attribute name="horzBarState" type="ST_SplitterBarState" use="optional" default="restored"/>
   <attribute name="preferSingleView" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.3.2.7 notesTextViewPr (Notes Text View Properties)

This element functions as a parent element within which all properties associated with the notes text view are contained. All properties are defined within the child elements.

Parent Elements
viewPr (§4.3.2.18)

Child Elements	Subclause
cViewPr (Common View Properties)	§4.3.2.2
extLst (Extension List)	§4.2.5

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NotesTextViewProperties">
   <sequence minOccurs="1" maxOccurs="1">
      <element name="cViewPr" type="CT_CommonViewProperties" minOccurs="1" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.3.2.8 notesViewPr (Notes View Properties)

This element functions as a parent element within which all view properties associated with notes are contained. All properties are defined within the child elements.

Parent Elements
viewPr (§4.3.2.18)

Child Elements	Subclause
cSldViewPr (Common Slide View Properties)	§4.3.2.1
extLst (Extension List)	§4.2.5

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NotesViewProperties">
   <sequence>
      <element name="cSldViewPr" type="CT_CommonSlideViewProperties" minOccurs="1" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.3.2.9 origin (View Origin)

This element specifies the origin of the slide when it is being viewed with various scaling factors using the scale element.

Parent Elements
cViewPr (§4.3.2.2)

Attributes	Description
x (X-Axis Coordinate) Namespace: .../drawingml/2006/main	Specifies a coordinate on the x-axis. The origin point for this coordinate shall be specified by the parent XML element. [*Example*: Consider the following point on a basic wrapping polygon for a DrawingML object: `<wp:… x="0" y="100" />` The x attribute defines an x-coordinate of `0`. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).
y (Y-Axis Coordinate) Namespace: .../drawingml/2006/main	Specifies a coordinate on the x-axis. The origin point for this coordinate shall be specified by the parent XML element. [*Example*: Consider the following point on a basic wrapping polygon for a DrawingML object: `<wp:… x="0" y="100" />` The y attribute defines a y-coordinate of `100`. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Point2D">
   <attribute name="x" type="ST_Coordinate" use="required"/>
   <attribute name="y" type="ST_Coordinate" use="required"/>
</complexType>
```

4.3.2.10 outlineViewPr (Outline View Properties)

This element functions as a parent element within which all view properties associated with the outline view mode are contained. All properties are defined within the child elements.

Outline view displays only the textual content of a presentation. The presentation is formatted as an outline, with slide titles as the first level of the outline. Body text on slides is indented below the slide title.

Parent Elements
viewPr (§4.3.2.18)

Child Elements	Subclause
cViewPr (Common View Properties)	§4.3.2.2
extLst (Extension List)	§4.2.5
sldLst (List of Presentation Slides)	§4.3.2.15

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OutlineViewProperties">
   <sequence minOccurs="1" maxOccurs="1">
      <element name="cViewPr" type="CT_CommonViewProperties" minOccurs="1" maxOccurs="1"/>
      <element name="sldLst" type="CT_OutlineViewSlideList" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.3.2.11 restoredLeft (Normal View Restored Left Properties)

This element specifies the sizing of the side content region of the normal view, when the region is of a variable restored size (neither minimized nor maximized).

Parent Elements
normalViewPr (§4.3.2.6)

Attributes	Description
autoAdjust (Auto Adjust Normal View)	Specifies whether the size of the side content region should compensate for the new size when resizing the window containing the view within the application. The possible values for this attribute are defined by the XML Schema boolean datatype.
sz (Normal View Dimension Size)	Specifies the size of the slide region (width when a child of restoredTop, height when a child of restoredLeft). The possible values for this attribute are defined by the ST_PositiveFixedPercentage simple type (§5.1.12.45).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NormalViewPortion">
   <attribute name="sz" type="a:ST_PositiveFixedPercentage" use="required"/>
   <attribute name="autoAdjust" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

4.3.2.12 restoredTop (Normal View Restored Top Properties)

This element specifies the sizing of the top slide region of the normal view, when the region is of a variable restored size (neither minimized nor maximized).

Parent Elements
normalViewPr (§4.3.2.6)

Attributes	Description
autoAdjust (Auto Adjust Normal View)	Specifies whether the size of the side content region should compensate for the new size when resizing the window containing the view within the application. The possible values for this attribute are defined by the XML Schema boolean datatype.
sz (Normal View Dimension Size)	Specifies the size of the slide region (width when a child of restoredTop, height when a child of restoredLeft). The possible values for this attribute are defined by the ST_PositiveFixedPercentage simple type (§5.1.12.45).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NormalViewPortion">
   <attribute name="sz" type="a:ST_PositiveFixedPercentage" use="required"/>
   <attribute name="autoAdjust" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

4.3.2.13 scale (View Scale)

This element specifies the view scaling factors that the presentation was last viewed with.

Parent Elements
cViewPr (§4.3.2.2)

Child Elements	Subclause
sx (Horizontal Ratio)	§5.1.2.1.38
sy (Vertical Ratio)	§5.1.2.1.39

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Scale2D">
   <sequence>
      <element name="sx" type="CT_Ratio" minOccurs="1" maxOccurs="1"/>
      <element name="sy" type="CT_Ratio" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.3.2.14 sld (Presentation Slide)

This element specifies a presentation slide and properties specific to the slide's appearance in outline view.

[*Example*: Consider the following presentation slide that has been collapsed in outline view.

```
<p:viewPr xmlns:a="" xmlns:r="" xmlns:p="" lastView="outlineView">
  ..
  <p:outlineViewPr>
    ..
    <p:sldLst>
      <p:sld r:id="rId1" collapse="1"/>
    </p:sldLst>
    ..
  </p:outlineViewPr>
  ..
</p:viewPr>
```

end example]

Parent Elements
sldLst (§4.3.2.15)

Attributes	Description
collapse (Collapsed)	Specifies whether this presentation slide is to be shown as collapsed within outline view. That is, all text other than the slide title will not be shown to the user. The possible values for this attribute are defined by the XML Schema boolean datatype.
id (Relationship Identifier) Namespace: .../officeDocument/2006/relationships	Specifies the relationship identifier that is used in conjunction with a corresponding relationship file to resolve the location of this presentation slide within a presentation. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OutlineViewSlideEntry">
   <attribute ref="r:id" use="required"/>
   <attribute name="collapse" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.3.2.15 sldLst (List of Presentation Slides)

This element specifies a list of presentation slides. A presentation slide contains the information that is specific to a single slide such as slide-specific shape and text information.

Parent Elements
outlineViewPr (§4.3.2.10)

Child Elements	Subclause
sld (Presentation Slide)	§4.3.2.14

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OutlineViewSlideList">
   <sequence>
      <element name="sld" type="CT_OutlineViewSlideEntry" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

4.3.2.16 slideViewPr (Slide View Properties)

This element functions as a parent element within which all view properties associated with the slide view mode are contained. All properties are defined within the child elements.

Parent Elements
viewPr (§4.3.2.18)

Child Elements	Subclause
cSldViewPr (Common Slide View Properties)	§4.3.2.1
extLst (Extension List)	§4.2.5

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SlideViewProperties">
   <sequence>
      <element name="cSldViewPr" type="CT_CommonSlideViewProperties" minOccurs="1" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.3.2.17 sorterViewPr (Slide Sorter View Properties)

This element functions as a parent element within which all view properties associated with the slide sorter view mode are contained. All properties are defined within the child elements.

The slide sorter view displays thumbnails of multiple slides at once; the number of slides and size of thumbnails depends on the scaling factor of the view.

Parent Elements
viewPr (§4.3.2.18)

Child Elements	Subclause
cViewPr (Common View Properties)	§4.3.2.2

Child Elements	Subclause
extLst (Extension List)	§4.2.5

Attributes	Description
showFormatting (Show Formatting)	Specifies whether to show associated slide formatting when in slide sorter view mode. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SlideSorterViewProperties">
   <sequence minOccurs="1" maxOccurs="1">
      <element name="cViewPr" type="CT_CommonViewProperties" minOccurs="1" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="showFormatting" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

4.3.2.18 viewPr (Presentation-wide View Properties)

This element functions as a parent element within which all presentation-wide view properties are contained. All properties and their corresponding settings are defined within the child elements.

Parent Elements
Root element of PresentationML View Properties part

Child Elements	Subclause
extLst (Extension List)	§4.2.5
gridSpacing (Grid Spacing)	§4.3.2.3
normalViewPr (Normal View Properties)	§4.3.2.6
notesTextViewPr (Notes Text View Properties)	§4.3.2.7
notesViewPr (Notes View Properties)	§4.3.2.8
outlineViewPr (Outline View Properties)	§4.3.2.10
slideViewPr (Slide View Properties)	§4.3.2.16
sorterViewPr (Slide Sorter View Properties)	§4.3.2.17

Attributes	Description
lastView (Last View)	Specifies the view mode that was used when the presentation document was last saved. The possible values for this attribute are defined by the ST_ViewType simple type (§4.8.59).

Attributes	Description
showComments (Show Comments)	Specifies whether the slide comments should be shown. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ViewProperties">
   <sequence minOccurs="0" maxOccurs="1">
      <element name="normalViewPr" type="CT_NormalViewProperties" minOccurs="0" maxOccurs="1"/>
      <element name="slideViewPr" type="CT_SlideViewProperties" minOccurs="0" maxOccurs="1"/>
      <element name="outlineViewPr" type="CT_OutlineViewProperties" minOccurs="0" maxOccurs="1"/>
      <element name="notesTextViewPr" type="CT_NotesTextViewProperties" minOccurs="0"
        maxOccurs="1"/>
      <element name="sorterViewPr" type="CT_SlideSorterViewProperties" minOccurs="0" maxOccurs="1"/>
      <element name="notesViewPr" type="CT_NotesViewProperties" minOccurs="0" maxOccurs="1"/>
      <element name="gridSpacing" type="a:CT_PositiveSize2D" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="lastView" type="ST_ViewType" use="optional" default="sldView"/>
   <attribute name="showComments" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

4.4 Slides

The Slides portion of the PresentationML framework stores all information pertaining specifically to slides of various types. These slide types and corresponding parts can be broken down into three distinct parts, namely slides, embedded objects, and programmable tags.

4.4.1 Slides

Being the main segment of this section of PresentationML, the slides elements encompass all data that is to be contained within a slide. The best way to think of a slide is a container for all data that is to be on that slide. The specific shapes, images and relations within a slide will not come into play here. The elements here pertain to the six different types of slides that can be described within PresentationML, namely slide, slide layout, slide master, handout master, notes master and notes slide.

4.4.1.1 bg (Slide Background)

This element specifies the background appearance information for a slide. The slide background covers the entire slide and is visible where no objects exist and as the background for transparent objects.

Parent Elements
cSld (§4.4.1.15)

Child Elements	Subclause
bgPr (Background Properties)	§4.4.1.2

Child Elements	Subclause
bgRef (Background Style Reference)	§4.4.1.3

Attributes	Description
bwMode (Black and White Mode)	Specifies that the background should be rendered using only black and white coloring. That is, the coloring information for the background should be converted to either black or white when rendering the picture. [Note: No gray is to be used in rendering this background, only stark black and stark white. End note] The possible values for this attribute are defined by the ST_BlackWhiteMode simple type (§5.1.12.10).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Background">
   <sequence>
      <group ref="EG_Background"/>
   </sequence>
   <attribute name="bwMode" type="a:ST_BlackWhiteMode" use="optional" default="white"/>
</complexType>
```

4.4.1.2 bgPr (Background Properties)

This element specifies visual effects used to render the slide background. This includes any fill, image, or effects that are to make up the background of the slide.

Parent Elements
bg (§4.4.1.1)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
effectDag (Effect Container)	§5.1.10.25
effectLst (Effect Container)	§5.1.10.26
extLst (Extension List)	§4.2.5
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
solidFill (Solid Fill)	§5.1.10.54

Attributes	Description
shadeToTitle (Shade to Title)	Specifies whether the background of the slide is of a shade to title type. This type of gradient fill is on the slide background and changes based on the placement of the slide title placeholder. An example is shown below.  The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BackgroundProperties">
   <sequence>
      <group ref="a:EG_FillProperties" minOccurs="1" maxOccurs="1"/>
      <group ref="a:EG_EffectProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="shadeToTitle" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.4.1.3 bgRef (Background Style Reference)

This element specifies the slide background is to use a fill style defined in the style matrix. The idx attribute refers to the index of a background fill style or fill style within the presentation's style matrix, defined by the fmtScheme element. A value of 0 or 1000 indicates no background, values 1-999 refer to the index of a fill style within the fillStyleLst element, and values 1001 and above refer to the index of a background fill style within the bgFillStyleLst element. The value 1001 corresponds to the first background fill style, 1002 to the second background fill style, and so on.

[*Example*:

```
<p:bgRef idx="2">
  <a:schemeClr val="bg2"/>
</p:bgRef>
```

The above code indicates a slide background with the style's second fill style using the second background color of the color scheme.

end example]

[*Example*:

```
<p:bgRef idx="1001">
  <a:schemeClr val="bg2"/>
</p:bgRef>
```

The above code indicates a slide background with the style's first background fill style using the second background color of the color scheme.

end example]

Parent Elements
bg (§4.4.1.1)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

Attributes	Description
idx (Style Matrix Index) Namespace: .../drawingml/2006/main	Specifies the style matrix index of the style referred to. The possible values for this attribute are defined by the ST_StyleMatrixColumnIndex simple type (§5.1.12.57).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_StyleMatrixReference">
   <sequence>
      <group ref="EG_ColorChoice" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="idx" type="ST_StyleMatrixColumnIndex" use="required"/>
</complexType>
```

4.4.1.4 blipFill (Picture Fill)

This element specifies the type of picture fill that the picture object will have. Because a picture has a picture fill already by default, it is possible to have two fills specified for a picture object. An example of this is shown below.

[Example: Consider the picture below that has a blip fill applied to it. The image used to fill this picture object has transparent pixels instead of white pixels.

```
<p:pic>
  ..
  <p:blipFill>
    <a:blip r:embed="rId2"/>
    <a:stretch>
      <a:fillRect/>
    </a:stretch>
  </p:blipFill>
  ..
</p:pic>
```

The above picture object is shown as an example of this fill type. End example]

[Example: Consider now the same picture object but with an additional gradient fill applied within the shape properties portion of the picture.

```
<p:pic>
  ..
  <p:blipFill>
    <a:blip r:embed="rId2"/>
    <a:stretch>
      <a:fillRect/>
    </a:stretch>
  </p:blipFill>
```

```
  <p:spPr>
    <a:gradFill>
      <a:gsLst>
        <a:gs pos="0">
          <a:schemeClr val="tx2">
            <a:shade val="50000"/>
          </a:schemeClr>
        </a:gs>
        <a:gs pos="39999">
          <a:schemeClr val="tx2">
            <a:tint val="20000"/>
          </a:schemeClr>
        </a:gs>
        <a:gs pos="70000">
          <a:srgbClr val="C4D6EB"/>
        </a:gs>
        <a:gs pos="100000">
          <a:schemeClr val="bg1"/>
        </a:gs>
      </a:gsLst>
    </a:gradFill>
  </p:spPr>
  ..
</p:pic>
```

The above picture object is shown as an example of this double fill type. End example]

Parent Elements
pic (§4.4.1.34)

Child Elements	Subclause
blip (Blip)	§5.1.10.13
srcRect (Source Rectangle)	§5.1.10.55
stretch (Stretch)	§5.1.10.56
tile (Tile)	§5.1.10.58

Attributes	Description
dpi (DPI Setting) Namespace: .../drawingml/2006/main	Specifies the DPI (dots per inch) used to calculate the size of the blip. If not present or zero, the DPI in the blip is used. [*Note*: This attribute is primarily used to keep track of the picture quality within a document. There are different levels of quality needed for print than on-screen viewing and thus a need to track this information. *end note*] The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
rotWithShape (Rotate With Shape) Namespace: .../drawingml/2006/main	Specifies that the fill should rotate with the shape. That is, when the shape that has been filled with a picture and the containing shape (say a rectangle) is transformed with a rotation then the fill will be transformed with the same rotation. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BlipFillProperties">
   <sequence>
      <element name="blip" type="CT_Blip" minOccurs="0" maxOccurs="1"/>
      <element name="srcRect" type="CT_RelativeRect" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_FillModeProperties" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="dpi" type="xsd:unsignedInt" use="optional"/>
   <attribute name="rotWithShape" type="xsd:boolean" use="optional"/>
</complexType>
```

4.4.1.5 bodyStyle (Slide Master Body Text Style)

This element specifies the text formatting style for all body text within a master slide. This formatting will be used on all body text within presentation slides related to this master. The text formatting is specified by utilizing the DrawingML framework just as within a regular presentation slide. Within the bodyStyle element there may be many different types of styles defined as there are different types of text stored within the body of a slide.

Parent Elements

Parent Elements
txStyles (§4.4.1.48)

Child Elements	Subclause
defPPr (Default Paragraph Style)	§5.1.5.2.2
extLst (Extension List)	§5.1.2.1.15
lvl1pPr (List Level 1 Text Style)	§5.1.5.4.13
lvl2pPr (List Level 2 Text Style)	§5.1.5.4.14
lvl3pPr (List Level 3 Text Style)	§5.1.5.4.15
lvl4pPr (List Level 4 Text Style)	§5.1.5.4.16
lvl5pPr (List Level 5 Text Style)	§5.1.5.4.17
lvl6pPr (List Level 6 Text Style)	§5.1.5.4.18
lvl7pPr (List Level 7 Text Style)	§5.1.5.4.19
lvl8pPr (List Level 8 Text Style)	§5.1.5.4.20
lvl9pPr (List Level 9 Text Style)	§5.1.5.4.21

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextListStyle">
   <sequence>
      <element name="defPPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl1pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl2pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl3pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl4pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl5pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl6pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl7pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl8pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl9pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.4.1.6 clrMap (Color Scheme Map)

This element specifies the mapping layer that transforms one color scheme definition to another. Each attribute represents a color name that may be referenced in this master, and the value is the corresponding color in the theme.

[*Example*: Consider the following mapping of colors that applies to a slide master:

```
<p:clrMap bg1="dk1" tx1="lt1" bg2="dk2" tx2="lt2" accent1="accent1"
accent2="accent2" accent3="accent3" accent4="accent4" accent5="accent5"
accent6="accent6" hlink="hlink" folHlink="folHlink"/>
```

end example]

Parent Elements
handoutMaster (§4.4.1.21); notesMaster (§4.4.1.24); sldMaster (§4.4.1.39)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15

Attributes	Description
accent1 (Accent 1) Namespace: .../drawingml/2006/main	Specifies a color defined which is associated as the accent 1 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
accent2 (Accent 2) Namespace: .../drawingml/2006/main	Specifies a color defined which is associated as the accent 2 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
accent3 (Accent 3) Namespace: .../drawingml/2006/main	Specifies a color defined which is associated as the accent 3 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
accent4 (Accent 4) Namespace: .../drawingml/2006/main	Specifies a color defined which is associated as the accent 4 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
accent5 (Accent 5) Namespace: .../drawingml/2006/main	Specifies a color defined which is associated as the accent 5 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
accent6 (Accent 6) Namespace: .../drawingml/200	Specifies a color defined which is associated as the accent 6 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).

Attributes	Description
6/main	
bg1 (Background 1) Namespace: .../drawingml/2006/main	A color defined which is associated as the first background color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
bg2 (Background 2) Namespace: .../drawingml/2006/main	Specifies a color defined which is associated as the second background color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
folHlink (Followed Hyperlink) Namespace: .../drawingml/2006/main	Specifies a color defined which is associated as the color for a followed hyperlink. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
hlink (Hyperlink) Namespace: .../drawingml/2006/main	Specifies a color defined which is associated as the color for a hyperlink. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
tx1 (Text 1) Namespace: .../drawingml/2006/main	Specifies a color defined which is associated as the first text color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
tx2 (Text 2) Namespace: .../drawingml/2006/main	Specifies a color defined which is associated as the second text color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ColorMapping">
   <sequence>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="bg1" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="tx1" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="bg2" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="tx2" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent1" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent2" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent3" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent4" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent5" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent6" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="hlink" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="folHlink" type="ST_ColorSchemeIndex" use="required"/>
</complexType>
```

4.4.1.7 clrMapOvr (Color Scheme Map Override)

This element provides a mechanism with which to override the color schemes listed within the ClrMap element. If the masterClrMapping element is present, the color scheme defined by the master is used. If the overrideClrMapping element is present, it defines a new color scheme specific to the parent notes slide, presentation slide, or slide layout.

Parent Elements
notes (§4.4.1.23); sld (§4.4.1.35); sldLayout (§4.4.1.36)

Child Elements	Subclause
masterClrMapping (Master Color Mapping)	§5.1.8.6
overrideClrMapping (Override Color Mapping)	§5.1.8.8

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ColorMappingOverride">
   <sequence>
      <choice minOccurs="1" maxOccurs="1">
         <element name="masterClrMapping" type="CT_EmptyElement"/>
         <element name="overrideClrMapping" type="CT_ColorMapping"/>
      </choice>
   </sequence>
</complexType>
```

4.4.1.8 cNvCxnSpPr (Non-Visual Connector Shape Drawing Properties)

This element specifies the non-visual drawing properties specific to a connector shape. This includes information specifying the shapes to which the connector shape is connected.

Parent Elements
nvCxnSpPr (§4.4.1.26)

Child Elements	Subclause
cxnSpLocks (Connection Shape Locks)	§5.1.2.1.11
endCxn (Connection End)	§5.1.2.1.13
extLst (Extension List)	§5.1.2.1.15
stCxn (Connection Start)	§5.1.2.1.36

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualConnectorProperties">
   <sequence>
      <element name="cxnSpLocks" type="CT_ConnectorLocking" minOccurs="0" maxOccurs="1"/>
      <element name="stCxn" type="CT_Connection" minOccurs="0" maxOccurs="1"/>
      <element name="endCxn" type="CT_Connection" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.4.1.9 cNvGraphicFramePr (Non-Visual Graphic Frame Drawing Properties)

This element specifies the non-visual drawing properties for a graphic frame. These non-visual properties are properties that the generating application would utilize when rendering the slide surface.

Parent Elements
nvGraphicFramePr (§4.4.1.27)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
graphicFrameLocks (Graphic Frame Locks)	§5.1.2.1.19

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualGraphicFrameProperties">
   <sequence>
      <element name="graphicFrameLocks" type="CT_GraphicalObjectFrameLocking" minOccurs="0"
        maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.4.1.10 cNvGrpSpPr (Non-Visual Group Shape Drawing Properties)

This element specifies the non-visual drawing properties for a group shape. These non-visual properties are properties that the generating application would utilize when rendering the slide surface.

Parent Elements
nvGrpSpPr (§4.4.1.28)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
grpSpLocks (Group Shape Locks)	§5.1.2.1.21

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualGroupDrawingShapeProps">
   <sequence>
      <element name="grpSpLocks" type="CT_GroupLocking" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.4.1.11 cNvPicPr (Non-Visual Picture Drawing Properties)

This element specifies the non-visual properties for the picture canvas. These properties are to be used by the generating application to determine how certain properties are to be changed for the picture object in question.

[*Example*: Consider the following DrawingML.

```
<p:pic>
  ..
  <p:nvPicPr>
    <p:cNvPr id="4" name="Lilly_by_Lisher.jpg"/>
    <p:cNvPicPr>
      <a:picLocks noChangeAspect="1"/>
    </p:cNvPicPr>
    <p:nvPr/>
  </p:nvPicPr>
  ..
</p:pic>
```

end example]

Parent Elements
nvPicPr (§4.4.1.29)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
picLocks (Picture Locks)	§5.1.2.1.31

Attributes	Description
preferRelativeResize (Relative Resize Preferred) Namespace: .../drawingml/2006/main	Specifies if the user interface should show the resizing of the picture based on the picture's current size or its original size. If this attribute is set to true, then scaling will be relative to the original picture size as opposed to the current picture size. [*Example*: Consider the case where a picture has been resized within a document and is now 50% of the originally inserted picture size. Now if the user chooses to make a later adjustment to the size of this picture within the generating application, then the value of this attribute should be checked. If this attribute is set to true then a value of 50% will be shown. Similarly, if this attribute is set to false, then a value of 100% should be shown because the picture has not yet been resized from its current (smaller) size. *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualPictureProperties">
   <sequence>
      <element name="picLocks" type="CT_PictureLocking" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="preferRelativeResize" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

4.4.1.12 cNvPr (Non-Visual Drawing Properties)

This element specifies non-visual canvas properties. This allows for additional information that does not affect the appearance of the picture to be stored.

[Example: Consider the following DrawingML.

```
<p:pic>
  ..
  <p:nvPicPr>
    <p:cNvPr id="4" name="Lilly_by_Lisher.jpg"/>
  </p:nvPicPr>
  ..
</p:pic>
```

End example]

Parent Elements
nvCxnSpPr (§4.4.1.26); nvGraphicFramePr (§4.4.1.27); nvGrpSpPr (§4.4.1.28); nvPicPr (§4.4.1.29); nvSpPr (§4.4.1.31)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
hlinkClick (Click Hyperlink)	§5.1.5.3.5
hlinkHover (Hyperlink for Hover)	§5.1.2.1.23

Attributes	Description
descr (Alternative Text for Object) Namespace: .../drawingml/2006/main	Specifies alternative text for the current DrawingML object, for use by assistive technologies or applications which will not display the current object. If this element is omitted, then no alternative text is present for the parent object. [*Example*: Consider a DrawingML object defined as follows: `<… descr="A picture of a bowl of fruit">` The descr attribute contains alternative text which may be used in place of the actual DrawingML object. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
hidden (Hidden) Namespace: .../drawingml/2006/main	Specifies whether this DrawingML object shall be displayed. When a DrawingML object is displayed within a document, that object may be hidden (i.e., present, but not visible). This attribute shall determine whether the object shall be rendered or made hidden. [*Note*: An application may have settings which allow this object to be viewed. *end note*] If this attribute is omitted, then the parent DrawingML object shall be displayed (i.e., not hidden). [*Example*: Consider an inline DrawingML object which shall be hidden within the document's content. This setting would be specified as follows: `<… hidden="true" />` The hidden attribute has a value of `true`, which specifies that the DrawingML object is hidden and not displayed when the document is displayed. *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
id (Unique Identifier) Namespace: .../drawingml/2006/main	Specifies a unique identifier for the current DrawingML object within the current document. This ID may be used to assist in uniquely identifying this object so that it can be referred to by other parts of the document. If multiple objects within the same document share the same id attribute value, then the document shall be considered non-conformant. [*Example*: Consider a DrawingML object defined as follows:

Attributes	Description
	<… id="10" … > The id attribute has a value of 10, which is the unique identifier for this DrawingML object. *end example*] The possible values for this attribute are defined by the ST_DrawingElementId simple type (§5.1.12.19).
name (Name) Namespace: …/drawingml/2006/main	Specifies the name of the object. [*Note*: Typically, this will be used to store the original file name of a picture object. *end note*] [*Example*: Consider a DrawingML object defined as follows: < … name="foo.jpg" > The name attribute has a value of foo.jpg, which is the name of this DrawingML object. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualDrawingProps">
   <sequence>
      <element name="hlinkClick" type="CT_Hyperlink" minOccurs="0" maxOccurs="1"/>
      <element name="hlinkHover" type="CT_Hyperlink" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="id" type="ST_DrawingElementId" use="required"/>
   <attribute name="name" type="xsd:string" use="required"/>
   <attribute name="descr" type="xsd:string" use="optional" default=""/>
   <attribute name="hidden" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.4.1.13 cNvSpPr (Non-Visual Drawing Properties for a Shape)

This element specifies the non-visual drawing properties for a shape. These properties are to be used by the generating application to determine how the shape should be dealt with

[*Example*: Consider the shape that has a shape lock applied to it.

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="2" name="Rectangle 1"/>
    <p:cNvSpPr>
      <a:spLocks noGrp="1"/>
    </p:cNvSpPr>
  </p:nvSpPr>
```

```
    ..
</p:sp>
```

This shape lock is stored within the non-visual drawing properties for this shape. *End example*]

Parent Elements
nvSpPr (§4.4.1.31)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
spLocks (Shape Locks)	§5.1.2.1.34

Attributes	Description
txBox (Text Box) Namespace: .../drawingml/2006/main	Specifies that the corresponding shape is a text box and thus should be treated as such by the generating application. If this attribute is omitted then it is assumed that the corresponding shape is not specifically a text box. [*Note*: Because a shape is not specified to be a text box does not mean that it cannot have text attached to it. A text box is merely a specialized shape with specific properties. *end note*] The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualDrawingShapeProps">
   <sequence>
      <element name="spLocks" type="CT_ShapeLocking" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="txBox" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.4.1.14 controls (List of controls)

This element specifies a list of embedded controls for the corresponding slide. Custom embedded controls may be embedded on slides.

Parent Elements
cSld (§4.4.1.15)

Child Elements	Subclause
control (Embedded Control)	§4.4.2.1

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ControlList">
   <sequence>
      <element name="control" type="CT_Control" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

4.4.1.15 cSld (Common Slide Data)

This element specifies a container for the type of slide information that is relevant to all of the slide types. All slides share a common set of properties that is independent of the slide type; the description of these properties for any particular slide is stored within the slide's cSld container. Slide data specific to the slide type indicated by the parent element is stored elsewhere.

[*Note*: The actual data in cSld describe only the particular parent slide; it is only the type of information stored that is common across all slides. *end note*]

[*Example*: Consider the following PresentationML slide

```
<p:sld>
  <p:cSld>
    <p:spTree>
      ..
    </p:spTree>
  </p:cSld>
  ..
</p:sld>
```

As the above example shows, the shape tree of a slide (spTree) is a child element of cSld because all slide types may contain a shape tree. Other slide properties specific to the type of slide (such as transitions for sld slides) are specified elsewhere. *end example*]

Parent Elements
handoutMaster (§4.4.1.21); notes (§4.4.1.23); notesMaster (§4.4.1.24); sld (§4.4.1.35); sldLayout (§4.4.1.36); sldMaster (§4.4.1.39)

Child Elements	Subclause
bg (Slide Background)	§4.4.1.1
controls (List of controls)	§4.4.1.14
custDataLst (Customer Data List)	§4.4.1.16
extLst (Extension List)	§4.2.5
spTree (Shape Tree)	§4.4.1.42

Attributes	Description
name (Name)	Specifies the slide name property that is used to further identify this unique configuration of common slide data. This might be used to aid in distinguishing different slide layouts or various other slide types. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CommonSlideData">
   <sequence>
      <element name="bg" type="CT_Background" minOccurs="0" maxOccurs="1"/>
      <element name="spTree" type="CT_GroupShape" minOccurs="1" maxOccurs="1"/>
      <element name="custDataLst" type="CT_CustomerDataList" minOccurs="0" maxOccurs="1"/>
      <element name="controls" type="CT_ControlList" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="name" type="xsd:string" use="optional" default=""/>
</complexType>
```

4.4.1.16 custDataLst (Customer Data List)

This element allows for the specifying of customer defined data within the PresentationML framework. References to custom data or tags may be defined within this list.

Parent Elements
cSld (§4.4.1.15); nvPr (§4.4.1.30); presentation (§4.3.1.24)

Child Elements	Subclause
custData (Customer Data)	§4.2.1
tags (Customer Data Tags)	§4.2.9

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustomerDataList">
   <sequence minOccurs="0" maxOccurs="1">
      <element name="custData" type="CT_CustomerData" minOccurs="0" maxOccurs="unbounded"/>
      <element name="tags" type="CT_TagsData" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.4.1.17 cxnSp (Connection Shape)

This element specifies a connection shape that is used to connect two sp elements. Once a connection is specified using a cxnSp, it is left to the generating application to determine the exact path the connector will take. That is the connector routing algorithm is left up to the generating application as the desired path might be different depending on the specific needs of the application.

[Example: Consider the following connector shape that connects two regular shapes.

```
<p:spTree>
  ..
  <p:sp>
    <p:nvSpPr>
      <p:cNvPr id="1" name="Rectangle 1"/>
      <p:cNvSpPr/>
      <p:nvPr/>
    </p:nvSpPr>
    ..
  </p:sp>
  <p:sp>
    <p:nvSpPr>
      <p:cNvPr id="2" name="Rectangle 2"/>
      <p:cNvSpPr/>
      <p:nvPr/>
    </p:nvSpPr>
    ..
  </p:sp>
  <p:cxnSp>
    <p:nvCxnSpPr>
      <p:cNvPr id="3" name="Elbow Connector 3"/>
      <p:cNvCxnSpPr>
        <a:stCxn id="1" idx="3"/>
        <a:endCxn id="2" idx="1"/>
      </p:cNvCxnSpPr>
      <p:nvPr/>
    </p:nvCxnSpPr>
    ..
  </p:cxnSp>
</p:spTree>
```

End example]

Parent Elements
grpSp (§4.4.1.19); spTree (§4.4.1.42)

Child Elements	Subclause
extLst (Extension List with Modification Flag)	§4.2.4
nvCxnSpPr (Non-Visual Properties for a Connection Shape)	§4.4.1.26
spPr (Shape Properties)	§4.4.1.41
style (Shape Style)	§4.4.1.43

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Connector">
   <sequence>
      <element name="nvCxnSpPr" type="CT_ConnectorNonVisual" minOccurs="1" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="1" maxOccurs="1"/>
      <element name="style" type="a:CT_ShapeStyle" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionListModify" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.4.1.18 graphicFrame (Graphic Frame)

This element specifies the existence of a graphics frame. This frame contains a graphic that was generated by an external source and needs a container in which to be displayed on the slide surface.

Parent Elements
grpSp (§4.4.1.19); spTree (§4.4.1.42)

Child Elements	Subclause
extLst (Extension List with Modification Flag)	§4.2.4
graphic (Graphic Object)	§5.1.2.1.16
nvGraphicFramePr (Non-Visual Properties for a Graphic Frame)	§4.4.1.27
xfrm (2D Transform for Graphic Frame)	§4.4.1.49

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GraphicalObjectFrame">
   <sequence>
      <element name="nvGraphicFramePr" type="CT_GraphicalObjectFrameNonVisual" minOccurs="1"
        maxOccurs="1"/>
      <element name="xfrm" type="a:CT_Transform2D" minOccurs="1" maxOccurs="1"/>
      <element ref="a:graphic" minOccurs="1" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionListModify" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.4.1.19 grpSp (Group Shape)

This element specifies a group shape that represents many shapes grouped together. This shape is to be treated just as if it were a regular shape but instead of being described by a single geometry it is made up of all the

shape geometries encompassed within it. Within a group shape each of the shapes that make up the group are specified just as they normally would. The idea behind grouping elements however is that a single transform can apply to many shapes at the same time.

[Example: Consider the following group shape.

```
<p:grpSp>
  <p:nvGrpSpPr>
    <p:cNvPr id="10" name="Group 9"/>
    <p:cNvGrpSpPr/>
    <p:nvPr/>
  </p:nvGrpSpPr>
  <p:grpSpPr>
    <a:xfrm>
      <a:off x="838200" y="990600"/>
      <a:ext cx="2426208" cy="978408"/>
      <a:chOff x="838200" y="990600"/>
      <a:chExt cx="2426208" cy="978408"/>
   </a:xfrm>
  </p:grpSpPr>
  <p:sp>
  ..
  </p:sp>
  <p:sp>
  ..
  </p:sp>
  <p:sp>
  ..
  </p:sp>
</p:grpSp>
```

In the above example we see three shapes specified within a single group. These three shapes have their position and sizes specified just as they normally would within the shape tree. The generating application should apply the transformation after the bounding box for the group shape has been calculated. End example]

Parent Elements
grpSp (§4.4.1.19); spTree (§4.4.1.42)

Child Elements	Subclause
cxnSp (Connection Shape)	§4.4.1.17
extLst (Extension List with Modification Flag)	§4.2.4
graphicFrame (Graphic Frame)	§4.4.1.18

Child Elements	Subclause
grpSp (Group Shape)	§4.4.1.19
grpSpPr (Group Shape Properties)	§4.4.1.20
nvGrpSpPr (Non-Visual Properties for a Group Shape)	§4.4.1.28
pic (Picture)	§4.4.1.34
sp (Shape)	§4.4.1.40

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GroupShape">
   <sequence>
      <element name="nvGrpSpPr" type="CT_GroupShapeNonVisual" minOccurs="1" maxOccurs="1"/>
      <element name="grpSpPr" type="a:CT_GroupShapeProperties" minOccurs="1" maxOccurs="1"/>
      <choice minOccurs="0" maxOccurs="unbounded">
         <element name="sp" type="CT_Shape"/>
         <element name="grpSp" type="CT_GroupShape"/>
         <element name="graphicFrame" type="CT_GraphicalObjectFrame"/>
         <element name="cxnSp" type="CT_Connector"/>
         <element name="pic" type="CT_Picture"/>
      </choice>
      <element name="extLst" type="CT_ExtensionListModify" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.4.1.20 grpSpPr (Group Shape Properties)

This element specifies the properties that are to be common across all of the shapes within the corresponding group. If there are any conflicting properties within the group shape properties and the individual shape properties then the individual shape properties should take precedence.

Parent Elements
grpSp (§4.4.1.19); spTree (§4.4.1.42)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
effectDag (Effect Container)	§5.1.10.25
effectLst (Effect Container)	§5.1.10.26
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
scene3d (3D Scene Properties)	§5.1.4.1.26

Child Elements	Subclause
solidFill (Solid Fill)	§5.1.10.54
xfrm (2D Transform for Grouped Objects)	§5.1.9.5

Attributes	Description
bwMode (Black and White Mode) Namespace: .../drawingml/2006/main	Specifies that the group shape should be rendered using only black and white coloring. That is the coloring information for the group shape should be converted to either black or white when rendering the corresponding shapes. No gray is to be used in rendering this image, only stark black and stark white. [*Note*: This does not mean that the group shapes themselves are stored with only black and white color information. This attribute instead sets the rendering mode that the shapes will use when rendering. *end note*] The possible values for this attribute are defined by the ST_BlackWhiteMode simple type (§5.1.12.10).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GroupShapeProperties">
   <sequence>
      <element name="xfrm" type="CT_GroupTransform2D" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_FillProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_EffectProperties" minOccurs="0" maxOccurs="1"/>
      <element name="scene3d" type="CT_Scene3D" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="bwMode" type="ST_BlackWhiteMode" use="optional"/>
</complexType>
```

4.4.1.21 handoutMaster (Handout Master)

This element specifies an instance of a handout master slide. Within a handout master slide are contained all elements that describe the objects and their corresponding formatting for within a handout slide. Within a handout master slide the cSld element specifies the common slide elements such as shapes and their attached text bodies. There are other properties within a handout master slide but cSld encompasses the majority of the intended purpose for a handoutMaster slide.

Parent Elements
Root element of PresentationML Handout Master part

Child Elements	Subclause
clrMap (Color Scheme Map)	§4.4.1.6

Child Elements	Subclause
cSld (Common Slide Data)	§4.4.1.15
extLst (Extension List with Modification Flag)	§4.2.4
hf (Header/Footer information for a slide master)	§4.4.1.22

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HandoutMaster">
    <sequence>
        <element name="cSld" type="CT_CommonSlideData" minOccurs="1" maxOccurs="1"/>
        <group ref="EG_TopLevelSlide" minOccurs="1" maxOccurs="1"/>
        <element name="hf" type="CT_HeaderFooter" minOccurs="0" maxOccurs="1"/>
        <element name="extLst" type="CT_ExtensionListModify" minOccurs="0" maxOccurs="1"/>
    </sequence>
</complexType>
```

4.4.1.22 hf (Header/Footer information for a slide master)

This element specifies the header and footer information for a slide. Headers and footers consist of placeholders for text that should be consistent across all slides and slide types, such as a date and time, slide numbering, and custom header and footer text.

Parent Elements
handoutMaster (§4.4.1.21); notesMaster (§4.4.1.24); sldLayout (§4.4.1.36); sldMaster (§4.4.1.39)

Child Elements	Subclause
extLst (Extension List with Modification Flag)	§4.2.4

Attributes	Description
dt (Date/Time Placeholder)	Specifies whether the Date/Time placeholder is enabled for this master. If this attribute is not specified, a value of true should be assumed by the generating application. The possible values for this attribute are defined by the XML Schema boolean datatype.
ftr (Footer Placeholder)	Specifies whether the Footer placeholder is enabled for this master. If this attribute is not specified, a value of true should be assumed by the generating application. The possible values for this attribute are defined by the XML Schema boolean datatype.
hdr (Header Placeholder)	Specifies whether the Header placeholder is enabled for this master. If this attribute is not specified, a value of true should be assumed by the generating application. The possible values for this attribute are defined by the XML Schema boolean datatype.
sldNum (Slide Number	Specifies whether the slide number placeholder is enabled. If this attribute is not specified, a value of true should be assumed by the generating application.

Attributes	Description
Placeholder)	The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HeaderFooter">
   <sequence>
      <element name="extLst" type="CT_ExtensionListModify" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="sldNum" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="hdr" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="ftr" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="dt" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

4.4.1.23 notes (Notes Slide)

This element specifies the existence of a notes slide along with its corresponding data. Contained within a notes slide are all the common slide elements along with addition properties that are specific to the notes element.

[*Example*: Consider the following PresentationML notes slide

```
<p:notes>
  <p:cSld>
  ..
  </p:cSld>
  ..
</p:notes>
```

In the above example a notes element specifies the existence of a notes slide with all of its parts. Notice the cSld element, that specifies the common elements that can appear on any slide type and then any elements specify additional non-common properties for this notes slide. *end example*]

Parent Elements
Root element of PresentationML Notes Slide part

Child Elements	Subclause
clrMapOvr (Color Scheme Map Override)	§4.4.1.7
cSld (Common Slide Data)	§4.4.1.15
extLst (Extension List with Modification Flag)	§4.2.4

Attributes	Description

Attributes	Description
showMasterPhAnim (Show Master Placeholder Animations)	Specifies whether or not to display animations on placeholders from the master slide. The possible values for this attribute are defined by the XML Schema boolean datatype.
showMasterSp (Show Master Shapes)	Specifies if shapes on the master slide should be shown on slides or not. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NotesSlide">
   <sequence minOccurs="1" maxOccurs="1">
      <element name="cSld" type="CT_CommonSlideData" minOccurs="1" maxOccurs="1"/>
      <group ref="EG_ChildSlide" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionListModify" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attributeGroup ref="AG_ChildSlide"/>
</complexType>
```

4.4.1.24 notesMaster (Notes Master)

This element specifies an instance of a handout master slide. Within a handout master slide are contained all elements that describe the objects and their corresponding formatting for within a handout slide. Within a handout master slide the cSld element specifies the common slide elements such as shapes and their attached text bodies. There are other properties within a handout master slide but cSld encompasses the majority of the intended purpose for a handoutMaster slide.

Parent Elements
Root element of PresentationML Notes Master part

Child Elements	Subclause
clrMap (Color Scheme Map)	§4.4.1.6
cSld (Common Slide Data)	§4.4.1.15
extLst (Extension List with Modification Flag)	§4.2.4
hf (Header/Footer information for a slide master)	§4.4.1.22
notesStyle (Notes Text Style)	§4.4.1.25

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NotesMaster">
   <sequence>
      <element name="cSld" type="CT_CommonSlideData" minOccurs="1" maxOccurs="1"/>
      <group ref="EG_TopLevelSlide" minOccurs="1" maxOccurs="1"/>
      <element name="hf" type="CT_HeaderFooter" minOccurs="0" maxOccurs="1"/>
      <element name="notesStyle" type="a:CT_TextListStyle" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionListModify" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.4.1.25 notesStyle (Notes Text Style)

This element specifies the text formatting style for the all other text within a notes slide. This formatting will be used on all text within the corresponding notes slides. The text formatting is specified by utilizing the DrawingML framework just as within a regular presentation slide. Within the notesStyle element there may be many different types of styles defined as there are different types of text stored within a notes slide.

Parent Elements
notesMaster (§4.4.1.24)

Child Elements	Subclause
defPPr (Default Paragraph Style)	§5.1.5.2.2
extLst (Extension List)	§5.1.2.1.15
lvl1pPr (List Level 1 Text Style)	§5.1.5.4.13
lvl2pPr (List Level 2 Text Style)	§5.1.5.4.14
lvl3pPr (List Level 3 Text Style)	§5.1.5.4.15
lvl4pPr (List Level 4 Text Style)	§5.1.5.4.16
lvl5pPr (List Level 5 Text Style)	§5.1.5.4.17
lvl6pPr (List Level 6 Text Style)	§5.1.5.4.18
lvl7pPr (List Level 7 Text Style)	§5.1.5.4.19
lvl8pPr (List Level 8 Text Style)	§5.1.5.4.20
lvl9pPr (List Level 9 Text Style)	§5.1.5.4.21

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextListStyle">
   <sequence>
      <element name="defPPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl1pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl2pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl3pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl4pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl5pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl6pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl7pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl8pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl9pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.4.1.26 nvCxnSpPr (Non-Visual Properties for a Connection Shape)

This element specifies all non-visual properties for a connection shape. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a connection shape. This allows for additional information that does not affect the appearance of the connection shape to be stored.

Parent Elements
cxnSp (§4.4.1.17)

Child Elements	Subclause
cNvCxnSpPr (Non-Visual Connector Shape Drawing Properties)	§4.4.1.8
cNvPr (Non-Visual Drawing Properties)	§4.4.1.12
nvPr (Non-Visual Properties)	§4.4.1.30

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ConnectorNonVisual">
   <sequence>
      <element name="cNvPr" type="a:CT_NonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
      <element name="cNvCxnSpPr" type="a:CT_NonVisualConnectorProperties" minOccurs="1"
        maxOccurs="1"/>
      <element name="nvPr" type="CT_ApplicationNonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.4.1.27 nvGraphicFramePr (Non-Visual Properties for a Graphic Frame)

This element specifies all non-visual properties for a graphic frame. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a graphic

frame. This allows for additional information that does not affect the appearance of the graphic frame to be stored.

Parent Elements
graphicFrame (§4.4.1.18)

Child Elements	Subclause
cNvGraphicFramePr (Non-Visual Graphic Frame Drawing Properties)	§4.4.1.9
cNvPr (Non-Visual Drawing Properties)	§4.4.1.12
nvPr (Non-Visual Properties)	§4.4.1.30

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GraphicalObjectFrameNonVisual">
   <sequence>
      <element name="cNvPr" type="a:CT_NonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
      <element name="cNvGraphicFramePr" type="a:CT_NonVisualGraphicFrameProperties" minOccurs="1"
        maxOccurs="1"/>
      <element name="nvPr" type="CT_ApplicationNonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.4.1.28 nvGrpSpPr (Non-Visual Properties for a Group Shape)

This element specifies all non-visual properties for a group shape. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a group shape. This allows for additional information that does not affect the appearance of the group shape to be stored.

Parent Elements
grpSp (§4.4.1.19); spTree (§4.4.1.42)

Child Elements	Subclause
cNvGrpSpPr (Non-Visual Group Shape Drawing Properties)	§4.4.1.10
cNvPr (Non-Visual Drawing Properties)	§4.4.1.12
nvPr (Non-Visual Properties)	§4.4.1.30

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GroupShapeNonVisual">
   <sequence>
      <element name="cNvPr" type="a:CT_NonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
      <element name="cNvGrpSpPr" type="a:CT_NonVisualGroupDrawingShapeProps" minOccurs="1"
        maxOccurs="1"/>
      <element name="nvPr" type="CT_ApplicationNonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.4.1.29 nvPicPr (Non-Visual Properties for a Picture)

This element specifies all non-visual properties for a picture. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a picture. This allows for additional information that does not affect the appearance of the picture to be stored.

[*Example*: Consider the following PresentationML.

```
<p:pic>
  ..
  <p:nvPicPr>
    ..
  </p:nvPicPr>
  ..
</p:pic>
```

end example]

Parent Elements
pic (§4.4.1.34)

Child Elements	Subclause
cNvPicPr (Non-Visual Picture Drawing Properties)	§4.4.1.11
cNvPr (Non-Visual Drawing Properties)	§4.4.1.12
nvPr (Non-Visual Properties)	§4.4.1.30

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PictureNonVisual">
   <sequence>
      <element name="cNvPr" type="a:CT_NonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
      <element name="cNvPicPr" type="a:CT_NonVisualPictureProperties" minOccurs="1" maxOccurs="1"/>
      <element name="nvPr" type="CT_ApplicationNonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.4.1.30 nvPr (Non-Visual Properties)

This element specifies non-visual properties for objects. These properties include multimedia content associated with an object and properties indicating how the object is to be used or displayed in different contexts.

Parent Elements
nvCxnSpPr (§4.4.1.26); nvGraphicFramePr (§4.4.1.27); nvGrpSpPr (§4.4.1.28); nvPicPr (§4.4.1.29); nvSpPr (§4.4.1.31)

Child Elements	Subclause
audioCd (Audio from CD)	§5.1.3.1
audioFile (Audio from File)	§5.1.3.2
custDataLst (Customer Data List)	§4.4.1.16
extLst (Extension List)	§4.2.5
ph (Placeholder Shape)	§4.4.1.33
quickTimeFile (QuickTime from File)	§5.1.3.4
videoFile (Video from File)	§5.1.3.6
wavAudioFile (Audio from WAV File)	§5.1.3.7

Attributes	Description
isPhoto (Is a Photo Album)	Specifies whether the picture belongs to a photo album and should thus be included when editing a photo album within the generating application. The possible values for this attribute are defined by the XML Schema boolean datatype.
userDrawn (Is User Drawn)	Specifies if the corresponding object has been drawn by the user and should thus not be deleted. This allows for the flagging of slides that contain user drawn data. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ApplicationNonVisualDrawingProps">
   <sequence>
      <element name="ph" type="CT_Placeholder" minOccurs="0" maxOccurs="1"/>
      <group ref="a:EG_Media" minOccurs="0" maxOccurs="1"/>
      <element name="custDataLst" type="CT_CustomerDataList" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="isPhoto" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="userDrawn" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.4.1.31 nvSpPr (Non-Visual Properties for a Shape)

This element specifies all non-visual properties for a shape. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a shape. This allows for additional information that does not affect the appearance of the shape to be stored.

Parent Elements
sp (§4.4.1.40)

Child Elements	Subclause
cNvPr (Non-Visual Drawing Properties)	§4.4.1.12
cNvSpPr (Non-Visual Drawing Properties for a Shape)	§4.4.1.13
nvPr (Non-Visual Properties)	§4.4.1.30

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShapeNonVisual">
   <sequence>
      <element name="cNvPr" type="a:CT_NonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
      <element name="cNvSpPr" type="a:CT_NonVisualDrawingShapeProps" minOccurs="1" maxOccurs="1"/>
      <element name="nvPr" type="CT_ApplicationNonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.4.1.32 otherStyle (Slide Master Other Text Style)

This element specifies the text formatting style for the all other text within a master slide. This formatting will be used on all text not covered by the titleStyle or bodyStyle elements within related presentation slides. The text formatting is specified by utilizing the DrawingML framework just as within a regular presentation slide. Within the otherStyle element there may be many different types of styles defined as there are different types of text stored within a slide.

[*Note*: The otherStyle element is to be used for specifying the text formatting of text within a slide shape but not within a text box. Text box styling is handled from within the bodyStyle element. *end note*]

Parent Elements
txStyles (§4.4.1.48)

Child Elements	Subclause
defPPr (Default Paragraph Style)	§5.1.5.2.2
extLst (Extension List)	§5.1.2.1.15
lvl1pPr (List Level 1 Text Style)	§5.1.5.4.13
lvl2pPr (List Level 2 Text Style)	§5.1.5.4.14

Child Elements	Subclause
lvl3pPr (List Level 3 Text Style)	§5.1.5.4.15
lvl4pPr (List Level 4 Text Style)	§5.1.5.4.16
lvl5pPr (List Level 5 Text Style)	§5.1.5.4.17
lvl6pPr (List Level 6 Text Style)	§5.1.5.4.18
lvl7pPr (List Level 7 Text Style)	§5.1.5.4.19
lvl8pPr (List Level 8 Text Style)	§5.1.5.4.20
lvl9pPr (List Level 9 Text Style)	§5.1.5.4.21

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextListStyle">
   <sequence>
      <element name="defPPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl1pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl2pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl3pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl4pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl5pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl6pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl7pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl8pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl9pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.4.1.33 ph (Placeholder Shape)

This element specifies that the corresponding shape should be represented by the generating application as a placeholder. When a shape is considered a placeholder by the generating application it can have special properties to alert the user that they may enter content into the shape. Different types of placeholders are allowed and can be specified by using the placeholder type attribute for this element.

Parent Elements
nvPr (§4.4.1.30)

Child Elements	Subclause
extLst (Extension List with Modification Flag)	§4.2.4

Attributes	Description
hasCustomPrompt (Placeholder has	Specifies whether the corresponding placeholder should have a custom prompt or not.

Attributes	Description
custom prompt)	The possible values for this attribute are defined by the XML Schema boolean datatype.
idx (Placeholder Index)	Specifies the placeholder index. This is used when applying templates or changing layouts to match a placeholder on one template/master to another. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
orient (Placeholder Orientation)	Specifies the orientation of a placeholder. The possible values for this attribute are defined by the ST_Direction simple type (§4.8.5).
sz (Placeholder Size)	Specifies the size of a placeholder. The possible values for this attribute are defined by the ST_PlaceholderSize simple type (§4.8.13).
type (Placeholder Type)	Specifies what type of content a placeholder is intended to contain. The possible values for this attribute are defined by the ST_PlaceholderType simple type (§4.8.14).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Placeholder">
   <sequence>
      <element name="extLst" type="CT_ExtensionListModify" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="type" type="ST_PlaceholderType" use="optional" default="obj"/>
   <attribute name="orient" type="ST_Direction" use="optional" default="horz"/>
   <attribute name="sz" type="ST_PlaceholderSize" use="optional" default="full"/>
   <attribute name="idx" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="hasCustomPrompt" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.4.1.34 pic (Picture)

This element specifies the existence of a picture object within the document.

[*Example*: Consider the following PresentationML that specifies the existence of a picture within a document. This picture can have non-visual properties, a picture fill as well as shape properties attached to it.

```
<p:pic>
  <p:nvPicPr>
    <p:cNvPr id="4" name="lake.JPG" descr="Picture of a Lake" />
    <p:cNvPicPr>
      <a:picLocks noChangeAspect="1"/>
    </p:cNvPicPr>
    <p:nvPr/>
```

```
    </p:nvPicPr>
    <p:blipFill>
    ...
    </p:blipFill>
    <p:spPr>
    ...
    </p:spPr>
  </p:pic>
```

end example]

Parent Elements
grpSp (§4.4.1.19); spTree (§4.4.1.42)

Child Elements	Subclause
blipFill (Picture Fill)	§4.4.1.4
extLst (Extension List with Modification Flag)	§4.2.4
nvPicPr (Non-Visual Properties for a Picture)	§4.4.1.29
spPr (Shape Properties)	§4.4.1.41
style (Shape Style)	§4.4.1.43

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Picture">
   <sequence>
      <element name="nvPicPr" type="CT_PictureNonVisual" minOccurs="1" maxOccurs="1"/>
      <element name="blipFill" type="a:CT_BlipFillProperties" minOccurs="1" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="1" maxOccurs="1"/>
      <element name="style" type="a:CT_ShapeStyle" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionListModify" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.4.1.35 sld (Presentation Slide)

This element specifies a slide within a slide list. The slide list is used to specify an ordering of slides.

[*Example*: Consider the following custom show with an ordering of slides.

```
  <p:custShowLst>
    <p:custShow name="Custom Show 1" id="0">
      <p:sldLst>
        <p:sld r:id="rId4"/>
        <p:sld r:id="rId3"/>
        <p:sld r:id="rId2"/>
```

```
        <p:sld r:id="rId5"/>
      </p:sldLst>
    </p:custShow>
  </p:custShowLst>
```

In the above example the order specified to present the slides is slide 4, then 3, 2 and finally 5. *end example*]

Parent Elements
Root element of PresentationML Slide part

Child Elements	Subclause
clrMapOvr (Color Scheme Map Override)	§4.4.1.7
cSld (Common Slide Data)	§4.4.1.15
extLst (Extension List with Modification Flag)	§4.2.4
timing (Slide Timing Information for a Slide Layout)	§4.4.1.44
transition (Slide Transition for a Slide Layout)	§4.4.1.46

Attributes	Description
show (Show Slide in Slide Show)	Specifies that the current slide should be shown in slide show. If this attribute is omitted then a value of true is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
showMasterPhAnim (Show Master Placeholder Animations)	Specifies whether or not to display animations on placeholders from the master slide. The possible values for this attribute are defined by the XML Schema boolean datatype.
showMasterSp (Show Master Shapes)	Specifies if shapes on the master slide should be shown on slides or not. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Slide">
   <sequence minOccurs="1" maxOccurs="1">
      <element name="cSld" type="CT_CommonSlideData" minOccurs="1" maxOccurs="1"/>
      <group ref="EG_ChildSlide" minOccurs="0" maxOccurs="1"/>
      <element name="transition" type="CT_SlideTransition" minOccurs="0" maxOccurs="1"/>
      <element name="timing" type="CT_SlideTiming" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionListModify" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attributeGroup ref="AG_ChildSlide"/>
   <attribute name="show" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

4.4.1.36 sldLayout (Slide Layout)

This element specifies an instance of a slide layout. The slide layout contains in essence a template slide design that can be applied to any existing slide. When applied to an existing slide all corresponding content should be mapped to the new slide layout.

Parent Elements
Root element of PresentationML Slide Layout part

Child Elements	Subclause
clrMapOvr (Color Scheme Map Override)	§4.4.1.7
cSld (Common Slide Data)	§4.4.1.15
extLst (Extension List with Modification Flag)	§4.2.4
hf (Header/Footer information for a slide master)	§4.4.1.22
timing (Slide Timing Information for a Slide Layout)	§4.4.1.44
transition (Slide Transition for a Slide Layout)	§4.4.1.46

Attributes	Description
matchingName (Matching Name)	Specifies a name to be used in place of the name attribute within the cSld element. This is used for layout matching in response to layout changes and template applications. The possible values for this attribute are defined by the XML Schema string datatype.
preserve (Preserve Slide Layout)	Specifies whether the corresponding slide layout will be deleted when all the slides that follow that layout are deleted. If this attribute is not specified then a value of false should be assumed by the generating application. This would mean that the slide would in fact be deleted if no slides within the presentation were related to it. The possible values for this attribute are defined by the XML Schema boolean datatype.
showMasterPhAnim (Show Master	Specifies whether or not to display animations on placeholders from the master slide.

Attributes	Description
Placeholder Animations)	The possible values for this attribute are defined by the XML Schema boolean datatype.
showMasterSp (Show Master Shapes)	Specifies if shapes on the master slide should be shown on slides or not. The possible values for this attribute are defined by the XML Schema boolean datatype.
type (Slide Layout Type)	Specifies the slide layout type that is used by this slide. The possible values for this attribute are defined by the ST_SlideLayoutType simple type (§4.8.19).
userDrawn (Is User Drawn)	Specifies if the corresponding object has been drawn by the user and should thus not be deleted. This allows for the flagging of slides that contain user drawn data. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SlideLayout">
   <sequence minOccurs="1" maxOccurs="1">
      <element name="cSld" type="CT_CommonSlideData" minOccurs="1" maxOccurs="1"/>
      <group ref="EG_ChildSlide" minOccurs="0" maxOccurs="1"/>
      <element name="transition" type="CT_SlideTransition" minOccurs="0" maxOccurs="1"/>
      <element name="timing" type="CT_SlideTiming" minOccurs="0" maxOccurs="1"/>
      <element name="hf" type="CT_HeaderFooter" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionListModify" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attributeGroup ref="AG_ChildSlide"/>
   <attribute name="matchingName" type="xsd:string" use="optional" default=""/>
   <attribute name="type" type="ST_SlideLayoutType" use="optional" default="cust"/>
   <attribute name="preserve" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="userDrawn" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.4.1.37 sldLayoutId (Slide Layout Id)

This element specifies the relationship information for each slide layout that is used within the slide master. The slide master has relationship identifiers that it uses internally for determining the slide layouts that should be used. Then, to resolve what these slide layouts should be the sldLayoutId elements in the sldLayoutIdLst are utilized.

Parent Elements
sldLayoutIdLst (§4.4.1.38)

Child Elements	Subclause
extLst (Extension List)	§4.2.5

Attributes	Description
id (ID Tag) Namespace: .../officeDocument/2006/relationships	Specifies the relationship id value that the generating application can use to resolve which slide layout will be used in the creation of the slide. This relationship id is used within the relationship file for the master slide to expose the location of the corresponding layout file within the presentation. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
id (ID Tag)	Specifies the identification number that uniquely identifies this slide layout within the presentation file. The possible values for this attribute are defined by the ST_SlideLayoutId simple type (§4.8.18).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SlideLayoutIdListEntry">
   <sequence>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="id" type="ST_SlideLayoutId" use="optional"/>
   <attribute ref="r:id" use="required"/>
</complexType>
```

4.4.1.38 sldLayoutIdLst (List of Slide Layouts)

This element specifies the existence of the slide layout identification list. This list is contained within the slide master and is used to determine which layouts are being used within the slide master file. Each layout within the list of slide layouts has its own identification number and relationship identifier that uniquely identifies it within both the presentation document and the particular master slide within which it is used.

Parent Elements
sldMaster (§4.4.1.39)

Child Elements	Subclause
sldLayoutId (Slide Layout Id)	§4.4.1.37

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SlideLayoutIdList">
   <sequence>
      <element name="sldLayoutId" type="CT_SlideLayoutIdListEntry" minOccurs="0"
        maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

4.4.1.39 sldMaster (Slide Master)

This element specifies an instance of a slide master slide. Within a slide master slide are contained all elements that describe the objects and their corresponding formatting for within a presentation slide. Within a slide master slide are two main elements. The cSld element specifies the common slide elements such as shapes and their attached text bodies. Then the txStyles element specifies the formatting for the text within each of these shapes. The other properties within a slide master slide specify other properties for within a presentation slide such as color information, headers and footers, as well as timing and transition information for all corresponding presentation slides.

Parent Elements
Root element of PresentationML Slide Master part

Child Elements	Subclause
clrMap (Color Scheme Map)	§4.4.1.6
cSld (Common Slide Data)	§4.4.1.15
extLst (Extension List with Modification Flag)	§4.2.4
hf (Header/Footer information for a slide master)	§4.4.1.22
sldLayoutIdLst (List of Slide Layouts)	§4.4.1.38
timing (Slide Timing Information for a Slide Layout)	§4.4.1.44
transition (Slide Transition for a Slide Layout)	§4.4.1.46
txStyles (Slide Master Text Styles)	§4.4.1.48

Attributes	Description
preserve (Preserve Slide Master)	Specifies whether the corresponding slide layout will be deleted when all the slides that follow that layout are deleted. If this attribute is not specified then a value of false should be assumed by the generating application. This would mean that the slide would in fact be deleted if no slides within the presentation were related to it. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SlideMaster">
   <sequence minOccurs="1" maxOccurs="1">
      <element name="cSld" type="CT_CommonSlideData" minOccurs="1" maxOccurs="1"/>
      <group ref="EG_TopLevelSlide" minOccurs="1" maxOccurs="1"/>
      <element name="sldLayoutIdLst" type="CT_SlideLayoutIdList" minOccurs="0" maxOccurs="1"/>
      <element name="transition" type="CT_SlideTransition" minOccurs="0" maxOccurs="1"/>
      <element name="timing" type="CT_SlideTiming" minOccurs="0" maxOccurs="1"/>
      <element name="hf" type="CT_HeaderFooter" minOccurs="0" maxOccurs="1"/>
      <element name="txStyles" type="CT_SlideMasterTextStyles" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionListModify" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="preserve" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.4.1.40 sp (Shape)

This element specifies the existence of a single shape. A shape can either be a preset or a custom geometry, defined using the DrawingML framework. In addition to a geometry each shape can have both visual and non-visual properties attached. Text and corresponding styling information can also be attached to a shape. This shape is specified along with all other shapes within either the shape tree or group shape elements.

[Note: Shapes are the preferred mechanism for specifying text on a slide. End note]

Parent Elements
grpSp (§4.4.1.19); spTree (§4.4.1.42)

Child Elements	Subclause
extLst (Extension List with Modification Flag)	§4.2.4
nvSpPr (Non-Visual Properties for a Shape)	§4.4.1.31
spPr (Shape Properties)	§4.4.1.41
style (Shape Style)	§4.4.1.43
txBody (Shape Text Body)	§4.4.1.47

Attributes	Description
useBgFill (Use Background Fill)	Specifies that the shape fill should be set to that of the slide background surface. [Note: This attribute does not set the fill of the shape to be transparent but instead sets it to be filled with the portion of the slide background that is directly behind it. End note] The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Shape">
   <sequence>
      <element name="nvSpPr" type="CT_ShapeNonVisual" minOccurs="1" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="1" maxOccurs="1"/>
      <element name="style" type="a:CT_ShapeStyle" minOccurs="0" maxOccurs="1"/>
      <element name="txBody" type="a:CT_TextBody" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionListModify" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="useBgFill" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.4.1.41 spPr (Shape Properties)

This element specifies the visual shape properties that can be applied to a shape. These properties include the shape fill, outline, geometry, effects, and 3D orientation.

Parent Elements
cxnSp (§4.4.1.17); pic (§4.4.1.34); sp (§4.4.1.40)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
custGeom (Custom Geometry)	§5.1.11.8
effectDag (Effect Container)	§5.1.10.25
effectLst (Effect Container)	§5.1.10.26
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35
ln (Outline)	§5.1.2.1.24
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
prstGeom (Preset geometry)	§5.1.11.18
scene3d (3D Scene Properties)	§5.1.4.1.26
solidFill (Solid Fill)	§5.1.10.54
sp3d (Apply 3D shape properties)	§5.1.7.12
xfrm (2D Transform for Individual Objects)	§5.1.9.6

Attributes	Description
bwMode (Black and White Mode)	Specifies that the picture should be rendered using only black and white coloring. That is the coloring information for the picture should be converted to either black or white

Attributes	Description
Namespace: …/drawingml/2006/main	when rendering the picture. No gray is to be used in rendering this image, only stark black and stark white. [*Note*: This does not mean that the picture itself that is stored within the file is necessarily a black and white picture. This attribute instead sets the rendering mode that the picture will have applied to when rendering. *end note*] The possible values for this attribute are defined by the ST_BlackWhiteMode simple type (§5.1.12.10).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShapeProperties">
   <sequence>
      <element name="xfrm" type="CT_Transform2D" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_Geometry" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_FillProperties" minOccurs="0" maxOccurs="1"/>
      <element name="ln" type="CT_LineProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_EffectProperties" minOccurs="0" maxOccurs="1"/>
      <element name="scene3d" type="CT_Scene3D" minOccurs="0" maxOccurs="1"/>
      <element name="sp3d" type="CT_Shape3D" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="bwMode" type="ST_BlackWhiteMode" use="optional"/>
</complexType>
```

4.4.1.42 spTree (Shape Tree)

This element specifies all shapes within a slide. Contained within here are all the shapes, either grouped or not, that can be referenced on a given slide. As most objects within a slide are shapes, this represents the majority of content within a slide. Text and effects are attached to shapes that are contained within the spTree element.

[*Example*: Consider the following PresentationML slide

```
<p:sld>
  <p:cSld>
    <p:spTree>
      <p:nvGrpSpPr>
      ..
      </p:nvGrpSpPr>
      <p:grpSpPr>
      ..
      </p:grpSpPr>
      <p:sp>
      ..
      </p:sp>
```

```
      </p:spTree>
    </p:cSld>
    ..
  </p:sld>
```

In the above example the shape tree specifies all the shape properties for this slide. *end example*]

Parent Elements
cSld (§4.4.1.15)

Child Elements	Subclause
cxnSp (Connection Shape)	§4.4.1.17
extLst (Extension List with Modification Flag)	§4.2.4
graphicFrame (Graphic Frame)	§4.4.1.18
grpSp (Group Shape)	§4.4.1.19
grpSpPr (Group Shape Properties)	§4.4.1.20
nvGrpSpPr (Non-Visual Properties for a Group Shape)	§4.4.1.28
pic (Picture)	§4.4.1.34
sp (Shape)	§4.4.1.40

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GroupShape">
   <sequence>
      <element name="nvGrpSpPr" type="CT_GroupShapeNonVisual" minOccurs="1" maxOccurs="1"/>
      <element name="grpSpPr" type="a:CT_GroupShapeProperties" minOccurs="1" maxOccurs="1"/>
      <choice minOccurs="0" maxOccurs="unbounded">
         <element name="sp" type="CT_Shape"/>
         <element name="grpSp" type="CT_GroupShape"/>
         <element name="graphicFrame" type="CT_GraphicalObjectFrame"/>
         <element name="cxnSp" type="CT_Connector"/>
         <element name="pic" type="CT_Picture"/>
      </choice>
      <element name="extLst" type="CT_ExtensionListModify" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.4.1.43 style (Shape Style)

This element specifies the style information for a shape. This is used to define a shape's appearance in terms of the preset styles defined by the style matrix for the theme.

[*Example*:

```
  <p:style>
    <a:lnRef idx="3">
```

```
      <a:schemeClr val="lt1"/>
    </a:lnRef>
    <a:fillRef idx="1">
      <a:schemeClr val="accent3"/>
    </a:fillRef>
    <a:effectRef idx="1">
      <a:schemeClr val="accent3"/>
    </a:effectRef>
    <a:fontRef idx="minor">
      <a:schemeClr val="lt1"/>
    </a:fontRef>
  </p:style>
```

The parent shape of the above code is to have an outline that uses the third line style defined by the theme, use the first fill defined by the scheme, and be rendered with the first effect defined by the theme. Text inside the shape is to use the minor font defined by the theme.

end example]

Parent Elements
cxnSp (§4.4.1.17); pic (§4.4.1.34); sp (§4.4.1.40)

Child Elements	Subclause
effectRef (Effect Reference)	§5.1.4.2.8
fillRef (Fill Reference)	§5.1.4.2.10
fontRef (Font Reference)	§5.1.4.1.17
lnRef (Line Reference)	§5.1.4.2.19

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShapeStyle">
   <sequence>
      <element name="lnRef" type="CT_StyleMatrixReference" minOccurs="1" maxOccurs="1"/>
      <element name="fillRef" type="CT_StyleMatrixReference" minOccurs="1" maxOccurs="1"/>
      <element name="effectRef" type="CT_StyleMatrixReference" minOccurs="1" maxOccurs="1"/>
      <element name="fontRef" type="CT_FontReference" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.4.1.44 timing (Slide Timing Information for a Slide Layout)

This element specifies the timing information for handling all animations and timed events within the corresponding slide. This information is tracked via time nodes within the timing element. More information on the specifics of these time nodes and how they are to be defined can be found within the Animation section of the PresentationML framework.

Parent Elements
sld (§4.4.1.35); sldLayout (§4.4.1.36); sldMaster (§4.4.1.39)

Child Elements	Subclause
bldLst (Build List)	§4.6.14
extLst (Extension List with Modification Flag)	§4.2.4
tnLst (Time Node List)	§4.6.87

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SlideTiming">
    <sequence>
        <element name="tnLst" type="CT_TimeNodeList" minOccurs="0" maxOccurs="1"/>
        <element name="bldLst" type="CT_BuildList" minOccurs="0" maxOccurs="1"/>
        <element name="extLst" type="CT_ExtensionListModify" minOccurs="0" maxOccurs="1"/>
    </sequence>
</complexType>
```

4.4.1.45 titleStyle (Slide Master Title Text Style)

This element specifies the text formatting style for the title text within a master slide. This formatting will be used on all title text within related presentation slides. The text formatting is specified by utilizing the DrawingML framework just as within a regular presentation slide. Within a title style there may be many different types of styles defined as there are different types of text stored within a slide title.

Parent Elements
txStyles (§4.4.1.48)

Child Elements	Subclause
defPPr (Default Paragraph Style)	§5.1.5.2.2
extLst (Extension List)	§5.1.2.1.15
lvl1pPr (List Level 1 Text Style)	§5.1.5.4.13
lvl2pPr (List Level 2 Text Style)	§5.1.5.4.14
lvl3pPr (List Level 3 Text Style)	§5.1.5.4.15
lvl4pPr (List Level 4 Text Style)	§5.1.5.4.16
lvl5pPr (List Level 5 Text Style)	§5.1.5.4.17
lvl6pPr (List Level 6 Text Style)	§5.1.5.4.18
lvl7pPr (List Level 7 Text Style)	§5.1.5.4.19
lvl8pPr (List Level 8 Text Style)	§5.1.5.4.20
lvl9pPr (List Level 9 Text Style)	§5.1.5.4.21

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextListStyle">
   <sequence>
      <element name="defPPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl1pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl2pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl3pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl4pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl5pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl6pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl7pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl8pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl9pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.4.1.46 transition (Slide Transition for a Slide Layout)

This element specifies the type of slide transition that should be used to transition to the current slide from the previous slide. That is, the transition information is stored on the slide that will appear after the transition is complete.

Parent Elements
sld (§4.4.1.35); sldLayout (§4.4.1.36); sldMaster (§4.4.1.39)

Child Elements	Subclause
blinds (Blinds Slide Transition)	§4.6.18
checker (Checker Slide Transition)	§4.6.24
circle (Circle Slide Transition)	§4.6.26
comb (Comb Slide Transition)	§4.6.30
cover (Cover Slide Transition)	§4.6.32
cut (Cut Slide Transition)	§4.6.34
diamond (Diamond Slide Transition)	§4.6.35
dissolve (Dissolve Slide Transition)	§4.6.36
extLst (Extension List with Modification Flag)	§4.2.4
fade (Fade Slide Transition)	§4.6.41
newsflash (Newsflash Slide Transition)	§4.6.50
plus (Plus Slide Transition)	§4.6.54
pull (Pull Slide Transition)	§4.6.58
push (Push Slide Transition)	§4.6.59
random (Random Slide Transition)	§4.6.60

Child Elements	Subclause
randomBar (Random Bar Slide Transition)	§4.6.61
sndAc (Sound Action)	§4.6.69
split (Split Slide Transition)	§4.6.71
strips (Strips Slide Transition)	§4.6.74
wedge (Wedge Slide Transition)	§4.6.94
wheel (Wheel Slide Transition)	§4.6.95
wipe (Wipe Slide Transition)	§4.6.96
zoom (Zoom Slide Transition)	§4.6.97

Attributes	Description
advClick (Advance on Click)	Specifies whether a mouse click will advance the slide or not. If this attribute is not specified then a value of true is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
advTm (Advance after time)	Specifies the time, in milliseconds, after which the transition should start. This setting may be used in conjunction with the advClick attribute. If this attribute is not specified then it is assumed that no auto-advance will occur. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
spd (Transition Speed)	Specifies the transition speed that is to be used when transitioning from the current slide to the next. The possible values for this attribute are defined by the ST_TransitionSpeed simple type (§4.8.58).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SlideTransition">
   <sequence>
      <choice minOccurs="0" maxOccurs="1">
         <element name="blinds" type="CT_OrientationTransition"/>
         <element name="checker" type="CT_OrientationTransition"/>
         <element name="circle" type="CT_Empty"/>
         <element name="dissolve" type="CT_Empty"/>
         <element name="comb" type="CT_OrientationTransition"/>
         <element name="cover" type="CT_EightDirectionTransition"/>
         <element name="cut" type="CT_OptionalBlackTransition"/>
         <element name="diamond" type="CT_Empty"/>
         <element name="fade" type="CT_OptionalBlackTransition"/>
         <element name="newsflash" type="CT_Empty"/>
         <element name="plus" type="CT_Empty"/>
         <element name="pull" type="CT_EightDirectionTransition"/>
         <element name="push" type="CT_SideDirectionTransition"/>
         <element name="random" type="CT_Empty"/>
         <element name="randomBar" type="CT_OrientationTransition"/>
         <element name="split" type="CT_SplitTransition"/>
         <element name="strips" type="CT_CornerDirectionTransition"/>
         <element name="wedge" type="CT_Empty"/>
         <element name="wheel" type="CT_WheelTransition"/>
         <element name="wipe" type="CT_SideDirectionTransition"/>
         <element name="zoom" type="CT_InOutTransition"/>
      </choice>
      <element name="sndAc" minOccurs="0" maxOccurs="1" type="CT_TransitionSoundAction"/>
      <element name="extLst" type="CT_ExtensionListModify" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="spd" type="ST_TransitionSpeed" use="optional" default="fast"/>
   <attribute name="advClick" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="advTm" type="xsd:unsignedInt" use="optional"/>
</complexType>
```

4.4.1.47 txBody (Shape Text Body)

This element specifies the existence of text to be contained within the corresponding shape. All visible text and visible text related properties are contained within this element. There can be multiple paragraphs and within paragraphs multiple runs of text.

Parent Elements
sp (§4.4.1.40)

Child Elements	Subclause
bodyPr (Body Properties)	§5.1.5.1.1
lstStyle (Text List Styles)	§5.1.5.4.12
p (Text Paragraphs)	§5.1.5.2.6

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextBody">
   <sequence>
      <element name="bodyPr" type="CT_TextBodyProperties" minOccurs="1" maxOccurs="1"/>
      <element name="lstStyle" type="CT_TextListStyle" minOccurs="0" maxOccurs="1"/>
      <element name="p" type="CT_TextParagraph" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

4.4.1.48 txStyles (Slide Master Text Styles)

This element specifies the text styles within a slide master. Within this element is the styling information for title text, the body text and other slide text as well. This element is only for use within the Slide Master and thus sets the text styles for the corresponding presentation slides.

[*Example*: Consider the case where we would like to specify the title text for a master slide.

```
<p:txStyles>
  <p:titleStyle>
    <a:lvl1pPr algn="ctr" rtl="0" latinLnBrk="0">
      <a:spcBef>
        <a:spcPct val="0"/>
      </a:spcBef>
      <a:buNone/>
      <a:defRPr sz="4400" kern="1200">
        <a:solidFill>
          <a:schemeClr val="tx1"/>
        </a:solidFill>
        <a:latin typeface="+mj-lt"/>
        <a:ea typeface="+mj-ea"/>
        <a:cs typeface="+mj-cs"/>
      </a:defRPr>
    </a:lvl1pPr>
  </p:titleStyle>
</p:txStyles>
```

In the above example the title text will be set according to the above formatting for all related slides within the presentation. *end example*]

Parent Elements
sldMaster (§4.4.1.39)

Child Elements	Subclause
bodyStyle (Slide Master Body Text Style)	§4.4.1.5

Child Elements	Subclause
extLst (Extension List)	§4.2.5
otherStyle (Slide Master Other Text Style)	§4.4.1.32
titleStyle (Slide Master Title Text Style)	§4.4.1.45

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SlideMasterTextStyles">
   <sequence>
      <element name="titleStyle" type="a:CT_TextListStyle" minOccurs="0" maxOccurs="1"/>
      <element name="bodyStyle" type="a:CT_TextListStyle" minOccurs="0" maxOccurs="1"/>
      <element name="otherStyle" type="a:CT_TextListStyle" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.4.1.49 xfrm (2D Transform for Graphic Frame)

This element specifies the transform to be applied to the corresponding graphic frame. This transformation will be applied to the graphic frame just as it would be for a shape or group shape.

Parent Elements
graphicFrame (§4.4.1.18)

Child Elements	Subclause
ext (Extents)	§5.1.9.3
off (Offset)	§5.1.9.4

Attributes	Description
flipH (Horizontal Flip) Namespace: .../drawingml/2006/main	Specifies a horizontal flip. When true, this attribute defines that the shape will be flipped horizontally about the center of its bounding box. [*Example*: The following illustrates the effect of a horizontal flip.  *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
flipV (Vertical Flip)	Specifies a vertical flip. When true, this attribute defines that the group will be flipped vertically about the center of its bounding box.

Attributes	Description
Namespace: .../drawingml/2006/main	[*Example*: The following illustrates the effect of a vertical flip.  *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
rot (Rotation) Namespace: .../drawingml/2006/main	Specifies the rotation of the Graphic Frame. The units for which this attribute is specified in reside within the simple type definition referenced below. The possible values for this attribute are defined by the ST_Angle simple type (§5.1.12.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Transform2D">
   <sequence>
      <element name="off" type="CT_Point2D" minOccurs="0" maxOccurs="1"/>
      <element name="ext" type="CT_PositiveSize2D" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="rot" type="ST_Angle" use="optional" default="0"/>
   <attribute name="flipH" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="flipV" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.4.2 Embedded Objects

Within the slides portion of PresentationML, there are the embedded elements. These are objects that can be embedded within a slide. As we defined a slide to be a container it can be seen that it does not just contain shapes, pictures and text but embedded objects as well that are not necessarily native to the PresentationML platform.

4.4.2.1 control (Embedded Control)

This element specifies the existence of an embedded control in the slide.

Parent Elements
controls (§4.4.1.14)

Child Elements	Subclause
extLst (Extension List)	§4.2.5

Attributes	Description
id (Relationship ID) Namespace: .../officeDocument/2006/relationships	Specifies the relationship id that is used to identify this Embedded object from within a slide. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
imgH (Image Height)	Specifies the height of the embedded control. The possible values for this attribute are defined by the ST_PositiveCoordinate32 simple type (§5.1.12.43).
imgW (Image Width)	Specifies the width of the embedded control. The possible values for this attribute are defined by the ST_PositiveCoordinate32 simple type (§5.1.12.43).
name (Embedded Object Name)	Specifies the identifying name class used by scripting languages. This name is also used to construct the clipboard name. The possible values for this attribute are defined by the XML Schema string datatype.
showAsIcon (Show Embedded Object As Icon)	Specifies whether the Embedded object will be shows as an icon or using its native representation. The possible values for this attribute are defined by the XML Schema boolean datatype.
spid (Embedded object Shape ID)	Specifies the identifier of the shape associated with this Embedded object. The shape will contain all coordinate anchoring information. The possible values for this attribute are defined by the ST_ShapeID simple type (§5.1.12.55).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Control">
   <sequence>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attributeGroup ref="AG_Ole"/>
</complexType>
```

4.4.2.2 embed (Embedded Object or Control)

This element specifies an Embedded object or Control that is embedded within the presentation.

Parent Elements
oleObj (§4.4.2.4)

Child Elements	Subclause
extLst (Extension List)	§4.2.5

Attributes	Description
followColorScheme (Color Scheme Properties for Embedded object)	Specifies the Color Scheme Properties for the corresponding Embedded object being specified. The possible values for this attribute are defined by the ST_OleObjectFollowColorScheme simple type (§4.8.10).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OleObjectEmbed">
   <sequence>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="followColorScheme" type="ST_OleObjectFollowColorScheme" use="optional"
     default="none"/>
</complexType>
```

4.4.2.3 link (Linked Object or Control)

This element specifies a link to an external Embedded object or Control.

Parent Elements
oleObj (§4.4.2.4)

Child Elements	Subclause
extLst (Extension List)	§4.2.5

Attributes	Description
updateAutomatic (Update Linked Embedded Objects Automatically)	This attribute determines if linked embedded objects will be automatically updated when the presentation is opened or printed. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OleObjectLink">
   <sequence>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="updateAutomatic" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.4.2.4 oleObj (Global Element for Embedded objects and Controls)

This element specifies a global element to be used for an Embedded object and Control.

Child Elements	Subclause
embed (Embedded Object or Control)	§4.4.2.2
link (Linked Object or Control)	§4.4.2.3

Attributes	Description
id (Relationship ID) Namespace: .../officeDocument/2006/relationships	Specifies the relationship id that is used to identify this Embedded object from within a slide. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
imgH (Image Height)	Specifies the height of the embedded control. The possible values for this attribute are defined by the ST_PositiveCoordinate32 simple type (§5.1.12.43).
imgW (Image Width)	Specifies the width of the embedded control. The possible values for this attribute are defined by the ST_PositiveCoordinate32 simple type (§5.1.12.43).
name (Embedded Object Name)	Specifies the identifying name class used by scripting languages. This name is also used to construct the clipboard name. The possible values for this attribute are defined by the XML Schema string datatype.
progId (Embedded Object ProgID)	Specifies the progid for an Embedded object. The possible values for this attribute are defined by the XML Schema string datatype.
showAsIcon (Show Embedded Object As Icon)	Specifies whether the Embedded object will be shows as an icon or using its native representation. The possible values for this attribute are defined by the XML Schema boolean datatype.
spid (Embedded object Shape ID)	Specifies the identifier of the shape associated with this Embedded object. The shape will contain all coordinate anchoring information. The possible values for this attribute are defined by the ST_ShapeID simple type (§5.1.12.55).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OleObject">
   <choice minOccurs="1" maxOccurs="1">
      <element name="embed" type="CT_OleObjectEmbed"/>
      <element name="link" type="CT_OleObjectLink"/>
   </choice>
   <attributeGroup ref="AG_Ole"/>
   <attribute name="progId" type="xsd:string" use="optional"/>
</complexType>
```

4.4.3 Programmable Tags

Within the slides portion of PresentationML there are the tag elements. These are extensibility names and values that assist in the storage of legacy variables from older file formats.

4.4.3.1 tag (Programmable Extensibility Tag)

This element specifies a programmable extensibility tag to be used for storage of legacy variables.

Parent Elements
tagLst (§4.4.3.2)

Attributes	Description
name (Name)	Specifies the name associated with this specific programmable tag. The possible values for this attribute are defined by the XML Schema string datatype.
val (Value)	Specifies the value associated with this specific programmable tag. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_StringTag">
   <attribute name="name" type="xsd:string"/>
   <attribute name="val" type="xsd:string"/>
</complexType>
```

4.4.3.2 tagLst (Programmable Tab List)

This element specifies the list of programmable extensibility tags that will be used to store variables from legacy file formats.

Parent Elements
Root element of PresentationML User-Defined Tags part

Child Elements	Subclause

Child Elements	Subclause
tag (Programmable Extensibility Tag)	§4.4.3.1

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TagList">
   <sequence>
      <element name="tag" type="CT_StringTag" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

4.5 Comments

A comment is a text note attached to a slide, with the primary purpose of allowing readers of a presentation to provide feedback to the presentation author. Each comment contains an unformatted text string and information about its author, and is attached to a particular location on a slide. Comments may be visible while editing the presentation, but do not appear when a slide show is given. The displaying application decides when to display comments and determines their visual appearance.

4.5.1 cm (Comment)

This element specifies a single comment attached to a slide. It contains the text of the comment, its position on the slide, and attributes referring to its author and date.

[*Example:*

```
<p:cm authorId="0" dt="2006-08-28T17:26:44.129" idx="1">
  <p:pos x="10" y="10"/>
  <p:text>Add diagram to clarify.</p:text>
</p:cm>
```

End example]

Parent Elements
cmLst (§4.5.4)

Child Elements	Subclause
extLst (Extension List with Modification Flag)	§4.2.4
pos (Comment Position)	§4.5.5
text (Comment's Text Content)	§4.5.6

Attributes	Description
authorId (Comment Author ID)	This attribute specifies the author of the comment. It refers to the ID of an author in the comment author list for the document.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
dt (Comment Date/Time)	This attribute specifies the date and time this comment was last modified. The possible values for this attribute are defined by the XML Schema dateTime datatype.
idx (Comment Index)	This attribute specifies an identifier for this comment that is unique within a list of all comments by this author in this document. An author's first comment in a document has index 1. [*Note:* Because the index is unique only for the comment author, a document may contain multiple comments with the same index created by different authors. *End note*] The possible values for this attribute are defined by the ST_Index simple type (§4.8.7).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Comment">
   <sequence>
      <element name="pos" type="a:CT_Point2D" minOccurs="1" maxOccurs="1"/>
      <element name="text" type="xsd:string" minOccurs="1" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionListModify" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="authorId" type="xsd:unsignedInt" use="required"/>
   <attribute name="dt" type="xsd:dateTime" use="optional"/>
   <attribute name="idx" type="ST_Index" use="required"/>
</complexType>
```

4.5.2 cmAuthor (Comment Author)

This element specifies a single author with comments in the document. It contains a unique author ID, the author's name and initials, the index of the author's last comment, and the index of a color associated with the author.

[*Example:*

```
<p:cmAuthor id="0" name="Julie Lee" initials="JL" lastIdx="1" clrIdx="0"/>
```

End example]

Parent Elements
cmAuthorLst (§4.5.3)

Child Elements	Subclause
extLst (Extension List)	§4.2.5

Attributes	Description
clrIdx (Comment Author Color Index)	This attribute specifies an index into the generating application's comments color table to allow for visual (color) differentiation of different author's comments. This color is used for all comments by this author. If more authors exist than there are entries in the color table, the color index wraps around to the beginning of the table. [*Note:* It is left entirely up to the generating application to determine the amount of colors used in the comments color table and in what order these will be used when rendering comments on a slide surface. *End note*] The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
id (Comment Author ID)	This attribute specifies a unique (within the document) zero-based identifier that refers to a single comment author. [*Note:* The method of generating an author id is determined by the application and need not be sequential, provided each id is unique within the list of comment authors for the document. *End note*] The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
initials (Comment Author Initials)	This attribute specifies a string that represents the initials of this particular author. The value is not necessarily unique. It is intended for use by the application as an abbreviated version of the comment author's name. The possible values for this attribute are defined by the ST_Name simple type (§4.8.9).
lastIdx (Index of Comment Author's last comment)	Index of the last comment added to this document by this author. New comments by this author are counted starting with the value one greater than this index. [*Note:* The index of a deleted comment is not reused; therefore, this value is not an accurate count of the total number of comments by the author. *End note*] The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
name (Comment Author Name)	This attribute specifies the full name of this particular author. As a string, it has no security or authentication data. This value is not guaranteed to be unique across all document authors. The possible values for this attribute are defined by the ST_Name simple type (§4.8.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CommentAuthor">
   <sequence>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="id" type="xsd:unsignedInt" use="required"/>
   <attribute name="name" type="ST_Name" use="required"/>
   <attribute name="initials" type="ST_Name" use="required"/>
   <attribute name="lastIdx" type="xsd:unsignedInt" use="required"/>
   <attribute name="clrIdx" type="xsd:unsignedInt" use="required"/>
</complexType>
```

4.5.3 cmAuthorLst (List of Comment Authors)

This element specifies a list of authors with comments in the current document. Each comment in a document must refer to an author in this list. To determine if a new author is in this list, the author's name and initials must both match; otherwise, the new author is considered unique and a separate cmAuthor element is added.

[*Example:* A document contains comments left by two authors.

```
<p:cmAuthorLst>
    <p:cmAuthor id="0" name="Julie Lee" initials="JL" lastIdx="1" clrIdx="0"/>
    <p:cmAuthor id="1" name="Fred Jones" initials="FJ" lastIdx="2" clrIdx="1"/>
</p:cmAuthorLst>
```

End example]

Parent Elements
Root element of PresentationML Comment Authors part

Child Elements	Subclause
cmAuthor (Comment Author)	§4.5.2

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CommentAuthorList">
   <sequence>
      <element name="cmAuthor" type="CT_CommentAuthor" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

4.5.4 cmLst (Comment List)

This element specifies a list of comments for a particular slide.

[*Example:* A slide contains two comments, each left by a different author. This example demonstrates that two comments may have the same index if they are created by different authors.

```
<p:cmLst>
    <p:cm authorId="0" dt="2006-08-28T17:26:44.129" idx="1">
        <p:pos x="10" y="10"/>
        <p:text>Add diagram to clarify.</p:text>
    </p:cm>
    <p:cm authorId="1" dt="2006-08-28T17:44:19.679" idx="1">
        <p:pos x="1426" y="660"/>
        <p:text>Clean up this text.</p:text>
    </p:cm>
</p:cmLst>
```

End example]

Parent Elements
Root element of PresentationML Comments part

Child Elements	Subclause
cm (Comment)	§4.5.1

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CommentList">
   <sequence>
      <element name="cm" type="CT_Comment" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

4.5.5 pos (Comment Position)

This element specifies the positioning information for the placement of a comment on a slide surface. In LTR versions of the generating application, this position information should refer to the upper left point of the comment shape. In RTL versions of the generating application, this position information should refer to the upper right point of the comment shape.

[*Note:* The anchoring point on the slide surface is unaffected by a right-to-left or left-to-right layout change. That is the anchoring point remains the same for all language versions. *End note*]

[*Note:* Because there is no specified size or formatting for comments, this UI widget used to display a comment can be any size and thus the lower right point of the comment shape is determined by how the viewing application chooses to display comments. *End note*]

[*Example:*

```
<p:pos x="1426" y="660"/>
```

End example]

Parent Elements
cm (§4.5.1)

Attributes	Description
x (X-Axis Coordinate) Namespace: .../drawingml/2006/main	Specifies a coordinate on the x-axis. The origin point for this coordinate shall be specified by the parent XML element. [*Example*: Consider the following point on a basic wrapping polygon for a DrawingML object: `<wp:… x="0" y="100" />` The x attribute defines an x-coordinate of `0`. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).
y (Y-Axis Coordinate) Namespace: .../drawingml/2006/main	Specifies a coordinate on the x-axis. The origin point for this coordinate shall be specified by the parent XML element. [*Example*: Consider the following point on a basic wrapping polygon for a DrawingML object: `<wp:… x="0" y="100" />` The y attribute defines a y-coordinate of `100`. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Point2D">
   <attribute name="x" type="ST_Coordinate" use="required"/>
   <attribute name="y" type="ST_Coordinate" use="required"/>
</complexType>
```

4.5.6 text (Comment's Text Content)

This element specifies the content of a comment. This is the text with which the author has annotated the slide.

[*Example:*

```
<p:text>Add diagram to clarify.</p:text>
```

End example]

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
cm (§4.5.1)

4.6 Animation

The Animation section of the PresentationML framework stores the movement and related information of objects.

This schema is loosely based on the syntax and concepts from the Synchronized Multimedia Integration Language (SMIL), a W3C Recommendation for describing multimedia presentations using XML.

The schema describes all the animations effects on that reside on a slide and also the animation that occurs when going from slide to slide (slide transition).

Animations on a slide are inherently time-based and consists of an animation effects on an object or text.. Slide transitions however do not follow this concept and always appear before any animation on a slide.

All elements described in this schema are contained within the slide XML file. More superficially they are in the <transition> and the <timing> element as shown below:

```
<p:sld>
  <p:cSld>...
  <p:clrMapOvr>...
  <p:transition>...
  <p:timing>...
</p:sld>
```

4.6.1 anim (Animate)

This element is a generic animation element that requires little or no semantic understanding of the attribute being animated. It can animate text within a shape or even the shape itself.

[*Example*: Consider trying to emphasize text within a shape by changing the size of its font by 150%. The <anim> element should be used as follows:

```
<p:anim to="1.5" calcmode="lin" valueType="num">
  <p:cBhvr override="childStyle">
    <p:cTn id="1" dur="2000" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="1">
        <p:txEl>
          <p:charRg st="1" end="4"/>
        </p:txEl>
      </p:spTgt>
    </p:tgtEl>
    <p:attrNameLst>
```

```
          <p:attrName>style.fontSize</p:attrName>
        </p:attrNameLst>
      </p:cBhvr>
    </p:anim>
```

end example]

Parent Elements
childTnLst (§4.6.25); subTnLst (§4.6.78); tnLst (§4.6.87)

Child Elements	Subclause
cBhvr (Common Behavior)	§4.6.22
tavLst (Time Animated Value List)	§4.6.80

Attributes	Description
by (By)	This attribute specifies a relative offset value for the animation with respect to its position before the start of the animation. The possible values for this attribute are defined by the XML Schema string datatype.
calcmode (Calculation Mode)	This attribute specifies the interpolation mode for the animation. The possible values for this attribute are defined by the ST_TLAnimateBehaviorCalcMode simple type (§4.8.24).
from (From)	This attribute specifies the starting value of the animation. The possible values for this attribute are defined by the XML Schema string datatype.
to (To)	This attribute specifies the ending value for the animation as a percentage. The possible values for this attribute are defined by the XML Schema string datatype.
valueType (Value Type)	This attribute specifies the type of property value. The possible values for this attribute are defined by the ST_TLAnimateBehaviorValueType simple type (§4.8.25).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLAnimateBehavior">
   <sequence>
      <element name="cBhvr" type="CT_TLCommonBehaviorData" minOccurs="1" maxOccurs="1"/>
      <element name="tavLst" type="CT_TLTimeAnimateValueList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="by" type="xsd:string" use="optional"/>
   <attribute name="from" type="xsd:string" use="optional"/>
   <attribute name="to" type="xsd:string" use="optional"/>
   <attribute name="calcmode" type="ST_TLAnimateBehaviorCalcMode" use="optional"/>
   <attribute name="valueType" type="ST_TLAnimateBehaviorValueType" use="optional"/>
</complexType>
```

4.6.2 animClr (Animate Color Behavior)

This animation element is responsible for animating the color of an object.

[*Example*: Consider trying to emphasize a shape by changing its fill color to scheme color accent2. The <animClr> element should be used as follows:

```
<p:animClr clrSpc="rgb">
  <p:cBhvr>
    <p:cTn id="1" dur="2000" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="1"/>
    </p:tgtEl>
    <p:attrNameLst>
      <p:attrName>fillcolor</p:attrName>
    </p:attrNameLst>
  </p:cBhvr>
  <p:to>
    <a:schemeClr val="accent2"/>
  </p:to>
</p:animClr>
```

end example]

Parent Elements
childTnLst (§4.6.25); subTnLst (§4.6.78); tnLst (§4.6.87)

Child Elements	Subclause
by (By)	§4.6.21
cBhvr (Common Behavior)	§4.6.22
from (From)	§4.6.44
to (To)	§4.6.90

Attributes	Description
clrSpc (Color Space)	This attribute specifies the color space in which to interpolate the animation. Valid values for example can be HSL & RGB. The values for from/to/by/etc. can still be specified in any supported color format without affecting the color space within which the animation happens. The RGB color space is best used for doing animations between two different colors since it doesn't require going through any other hues between the two colors specified. The HSL space is useful for animating through a rainbow of colors or for modifying just the saturation by 30% for example. The possible values for this attribute are defined by the ST_TLAnimateColorSpace simple type (§4.8.27).
dir (Direction)	This attribute specifies which direction to cycle the hue around the color wheel. Valid values are clockwise or counter clockwise. Default is clockwise. The possible values for this attribute are defined by the ST_TLAnimateColorDirection simple type (§4.8.26).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLAnimateColorBehavior">
   <sequence>
      <element name="cBhvr" type="CT_TLCommonBehaviorData" minOccurs="1" maxOccurs="1"/>
      <element name="by" type="CT_TLByAnimateColorTransform" minOccurs="0" maxOccurs="1"/>
      <element name="from" type="a:CT_Color" minOccurs="0" maxOccurs="1"/>
      <element name="to" type="a:CT_Color" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="clrSpc" type="ST_TLAnimateColorSpace" use="optional"/>
   <attribute name="dir" type="ST_TLAnimateColorDirection" use="optional"/>
</complexType>
```

4.6.3 animEffect (Animate Effect)

This animation behavior provides the ability to do image transform/filter effects on elements. Some visual effects are dynamic in nature and have a progress that animates from 0 to 1 over a period of time to do visual transitions between hidden and visible states. Other filters are static and apply a effects like a blur or drop-shadow which aren't inherently time-based.

[*Example*: Consider trying to emphasize a shape by creating an entrance animation using a "blinds" motion.

```
<p:animEffect transition="in" filter="blinds(horizontal)">
  <p:cBhvr>
    <p:cTn id="7" dur="500"/>
    <p:tgtEl>
      <p:spTgt spid="4"/>
```

```
      </p:tgtEl>
    </p:cBhvr>
  </p:animEffect>
```

end example]

Parent Elements
childTnLst (§4.6.25); subTnLst (§4.6.78); tnLst (§4.6.87)

Child Elements	Subclause
cBhvr (Common Behavior)	§4.6.22
progress (Progress)	§4.6.57

Attributes	Description
filter (Filter)	This attribute specifies the named transitions for the effect. It allows specifying multiple down-level transition types to use. The runtime will try to use the first type listed and if that one is not supported, will try the next until it either finds one it supports or there are no more in the list. The syntax used for the filter attribute value is as follows: "type(subtype);type(subtype)". Subtype may be a string value such as "fromLeft" or a numerical value depending on the type specified. The possible values for this attribute are defined by the XML Schema string datatype.
prLst (Property List)	This attribute describes a list of properties that coincide with the effect specified in the filter attribute. These properties can be set by providing a name:value pairs in a semicolon-separated list. When multiple types are listed in the filter attribute, the runtime will attempt to apply each property value even though some may not apply to it. The possible values for this attribute are defined by the XML Schema string datatype.
transition (Transition)	This attribute specifies whether to transition the element in or out or treat it as a static filter. The valid values are none, in and out and the default value is in. When a value of "in" is specified, the element will not be visible at the start of the animation and will become completely visible be the end of the duration. When "out" is specified, the element will be visible at the start and not visible at the end of the effect. This visibility is in addition to the effect of setting CSS visibility or display attributes. The possible values for this attribute are defined by the ST_TLAnimateEffectTransition simple type (§4.8.28).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLAnimateEffectBehavior">
   <sequence>
      <element name="cBhvr" type="CT_TLCommonBehaviorData" minOccurs="1" maxOccurs="1"/>
      <element name="progress" type="CT_TLAnimVariant" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="transition" type="ST_TLAnimateEffectTransition" use="optional"/>
   <attribute name="filter" type="xsd:string" use="optional"/>
   <attribute name="prLst" type="xsd:string" use="optional"/>
</complexType>
```

4.6.4 animMotion (Animate Motion)

Animate motion provides an abstracted way to move positioned elements. It provides the ability to specify from/to/by type motion as well as to use more detailed path descriptions for motion over polylines or bezier curves.

[Example: Consider animating a shape from its original position to the right.. The <animMotion> element should be used as follows:

```
<p:animMotion origin="layout" path="M 0 0 L 0.25 0 E" pathEditMode="relative">
  <p:cBhvr>
    <p:cTn id="1" dur="2000" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="1"/>
    </p:tgtEl>
    <p:attrNameLst>
      <p:attrName>ppt_x</p:attrName>
      <p:attrName>ppt_y</p:attrName>
    </p:attrNameLst>
  </p:cBhvr>
</p:animMotion>
```

End example]

Parent Elements
childTnLst (§4.6.25); subTnLst (§4.6.78); tnLst (§4.6.87)

Child Elements	Subclause
by (By)	§4.6.20
cBhvr (Common Behavior)	§4.6.22
from (From)	§4.6.43
rCtr (Rotation Center)	§4.6.62
to (To)	§4.6.88

Attributes	Description
origin (Origin)	Specifies what the origin of the motion path is relative to such as the layout of the slide, or the parent. The possible values for this attribute are defined by the ST_TLAnimateMotionBehaviorOrigin simple type (§4.8.29).
path (Path)	Specifies the path primitive followed by coordinates for the animation motion. The allowed values that are understood within a path are as follows: M = move to, L = line to, C = curve to, Z=close loop, E=end UPPERCASE = absolute coords, lowercase = relative coords Thus total allowed set = {M,L,C,Z,E,m,l,c,z,e) [*Example*: The following string is a sample path. path: "M 0 0 L 1 1 c 1 2 3 4 4 4 Z" *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
pathEditMode (Path Edit Mode)	This attribute specifies how the motion path moves when the target element is moved. The possible values for this attribute are defined by the ST_TLAnimateMotionPathEditMode simple type (§4.8.30).
ptsTypes (Points Types)	This attribute describes the type of the points in the path attribute. The allowed values that are understood for the ptsTypes attribute are as follows: A = Auto, F = Corner, T = Straight, S = Smooth UPPERCASE = Straight Line follows point, lowercase = curve follows point. Thus, the total allowed set = {A,F,T,S,a,f,t,s} The possible values for this attribute are defined by the XML Schema string datatype.
rAng (Relative Angle)	The attribute describes the relative angle of the motion path. The possible values for this attribute are defined by the ST_Angle simple type (§5.1.12.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLAnimateMotionBehavior">
   <sequence>
      <element name="cBhvr" type="CT_TLCommonBehaviorData" minOccurs="1" maxOccurs="1"/>
      <element name="by" type="CT_TLPoint" minOccurs="0" maxOccurs="1"/>
      <element name="from" type="CT_TLPoint" minOccurs="0" maxOccurs="1"/>
      <element name="to" type="CT_TLPoint" minOccurs="0" maxOccurs="1"/>
      <element name="rCtr" type="CT_TLPoint" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="origin" type="ST_TLAnimateMotionBehaviorOrigin" use="optional"/>
   <attribute name="path" type="xsd:string" use="optional"/>
   <attribute name="pathEditMode" type="ST_TLAnimateMotionPathEditMode" use="optional"/>
   <attribute name="rAng" type="a:ST_Angle" use="optional"/>
   <attribute name="ptsTypes" type="xsd:string" use="optional"/>
</complexType>
```

4.6.5 animRot (Animate Rotation)

This animation element is responsible for animating the rotation of an object. Rotation values set in the "by" , "to, and "from" attributes are specified in degrees measured to a 60,000th, i.e 1 degree is 60,000. Rotation values can be larger than 360°.

The sign of the rotation angle specifies the direction for rotation. A negative rotation specifies that the rotation should appear in the host to go counter-clockwise".

[Example: Consider trying to emphasize a shape by rotating it 360 degrees clockwise. The <animRot> element should be used as follows:

```
<p:animRot by="21600000">
  <p:cBhvr>
    <p:cTn id="6" dur="2000" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="5"/>
    </p:tgtEl>
    <p:attrNameLst>
      <p:attrName>r</p:attrName>
    </p:attrNameLst>
  </p:cBhvr>
</p:animRot>
```

End example]

Parent Elements
childTnLst (§4.6.25); subTnLst (§4.6.78); tnLst (§4.6.87)

Child Elements	Subclause

Child Elements	Subclause
cBhvr (Common Behavior)	§4.6.22

Attributes	Description
by (By)	This attribute describes the relative offset value for the animation. The possible values for this attribute are defined by the ST_Angle simple type (§5.1.12.3).
from (From)	This attribute describes the starting value for the animation. The possible values for this attribute are defined by the ST_Angle simple type (§5.1.12.3).
to (To)	This attribute describes the ending value for the animation. The possible values for this attribute are defined by the ST_Angle simple type (§5.1.12.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLAnimateRotationBehavior">
   <sequence>
      <element name="cBhvr" type="CT_TLCommonBehaviorData" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="by" type="a:ST_Angle" use="optional"/>
   <attribute name="from" type="a:ST_Angle" use="optional"/>
   <attribute name="to" type="a:ST_Angle" use="optional"/>
</complexType>
```

4.6.6 animScale (Animate Scale)

This animation element is responsible for animating the scale of an object. When animating the scale, the element must scale around the reference point of the element and the positioning system used should be consistent with the one used for motion paths. When animating the width and height of an element, all of the width/height animation values are calculated first then the scale animations are applied on top of that. So for example, an animation from 0 to 100 of the width with a concurrent scale from 100% to 200% would result in the element appearing to scale from 0 to 200.

[Example: Consider trying to emphasize a shape by scaling it larger by 150%. The <animScale> element should be used as follows:

```
<p:childTnLst>
  <p:animScale>
    <p:cBhvr>
      <p:cTn id="6" dur="2000" fill="hold"/>
      <p:tgtEl>
        <p:spTgt spid="5"/>
      </p:tgtEl>
    </p:cBhvr>
```

```
      <p:by x="150000" y="150000"/>
    </p:animScale>
  </p:childTnLst>
```

End example]

Parent Elements
childTnLst (§4.6.25); subTnLst (§4.6.78); tnLst (§4.6.87)

Child Elements	Subclause
by (By)	§4.6.20
cBhvr (Common Behavior)	§4.6.22
from (From)	§4.6.43
to (To)	§4.6.88

Attributes	Description
zoomContents (Zoom Content)	This attribute specifies whether to zoom the contents of an object when doing a scaling animation. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLAnimateScaleBehavior">
   <sequence>
      <element name="cBhvr" type="CT_TLCommonBehaviorData" minOccurs="1" maxOccurs="1"/>
      <element name="by" type="CT_TLPoint" minOccurs="0" maxOccurs="1"/>
      <element name="from" type="CT_TLPoint" minOccurs="0" maxOccurs="1"/>
      <element name="to" type="CT_TLPoint" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="zoomContents" type="xsd:boolean" use="optional"/>
</complexType>
```

4.6.7 attrName (Attribute Name)

This element is used to contain an attribute value for an Attribute Name List. This value defines the specific attribute that an animation should be applied to, such as fill, style, and shadow, etc. A specific property is defined by using a "property.sub-property" format which is often extended to multiple sub properties as seen in the allowed values below.

Allowed property values:

style.opacity, style.rotation, style.visibility, style.color, style.fontSize, style.fontWeight, style.fontStyle, style.fontFamily, style.textEffectEmboss, style.textShadow, style.textTransform, style.textDecorationUnderline, style.textEffectOutline, style.textDecorationLineThrough, style.sRotation, imageData.cropTop, imageData.cropBottom, imageData.cropLeft, imageData.cropRight, imageData.cropRight, imageData.gain, imageData.blacklevel, imageData.gamma, imageData.grayscale, imageData.chromakey, fill.on, fill.type, fill.color, fill.opacity, fill.color2, fill.method, fill.opacity2, fill.angle, fill.focus, fill.focusposition.x, fill.focusposition.y, fill.focussize.x, fill.focussize.y, stroke.on, stroke.color, stroke.weight, stroke.opacity, stroke.linestyle, stroke.dashstyle, stroke.filltype, stroke.src, stroke.color2, stroke.imagesize.x, stroke.imagesize.y, stroke.startArrow, stroke.endArrow, stroke.startArrowWidth, stroke.startArrowLength, stroke.endArrowWidth, stroke.endArrowLength, shadow.on, shadow.type, shadow.color, shadow.color2, shadow.opacity, shadow.offset.x, shadow.offset.y, shadow.offset2.x, shadow.offset2.y, shadow.origin.x, shadow.origin.y, shadow.matrix.xtox, shadow.matrix.ytox, shadow.matrix.xtox, shadow.matrix.ytoy, shadow.matrix.perspectiveX, shadow.matrix.perspectiveY, skew.on, skew.offset.x, skew.offset.y, skew.origin.x, skew.origin.y, skew.matrix.xtox, skew.matrix.ytox, skew.matrix.xtox, skew.matrix.ytoy, skew.matrix.perspectiveX, skew.matrix.perspectiveY, extrusion.on, extrusion.type, extrusion.render, extrusion.viewpointorigin.x, extrusion.viewpointorigin.y, extrusion.viewpoint.x, extrusion.viewpoint.y, extrusion.viewpoint.z, extrusion.plane, extrusion.skewangle, extrusion.skewamt, extrusion.backdepth, extrusion.foredepth, extrusion.orientation.x, extrusion.orientation.y, extrusion.orientation.z, extrusion.orientationangle, extrusion.color, extrusion.rotationangle.x, extrusion.rotationangle.y, extrusion.lockrotationcenter, extrusion.autorotationcenter, extrusion.rotationcenter.x, extrusion.rotationcenter.y, extrusion.rotationcenter.z, and extrusion.colormode.

[*Example*: Consider trying to emphasize the txt font size within the body of a shape. The attribute would be 'style.fontSize' and this can be done by doing the following:

```
<p:anim to="1.5" calcmode="lin" valueType="num">
  <p:cBhvr override="childStyle">
    <p:cTn id="6" dur="2000" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="3">
        <p:txEl>
          <p:charRg st="4294967295" end="4294967295"/>
        </p:txEl>
      </p:spTgt>
    </p:tgtEl>
```

```
      <p:attrNameLst>
        <p:attrName>style.fontSize</p:attrName>
      </p:attrNameLst>
    </p:cBhvr>
  </p:anim>
```

end example]

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
attrNameLst (§4.6.8)

4.6.8 attrNameLst (Attribute Name List)

This element is used to describe a list of attributes in which to apply an animation to.

[Example: Consider trying to emphasize the txt font size within the body of a shape. The attribute would be 'style.fontSize' and this can be done by doing the following:

```
<p:anim to="1.5" calcmode="lin" valueType="num">
  <p:cBhvr override="childStyle">
    <p:cTn id="6" dur="2000" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="3">
        <p:txEl>
          <p:charRg st="4294967295" end="4294967295"/>
        </p:txEl>
      </p:spTgt>
    </p:tgtEl>
    <p:attrNameLst>
      <p:attrName>style.fontSize</p:attrName>
    </p:attrNameLst>
  </p:cBhvr>
</p:anim>
```

End example]

Parent Elements
cBhvr (§4.6.22)

Child Elements	Subclause
attrName (Attribute Name)	§4.6.7

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLBehaviorAttributeNameList">
   <sequence>
      <element name="attrName" type="xsd:string" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

4.6.9 audio (Audio)

This element is used to include audio during an animation.

[Example: Consider adding applause sound to an animation sequence. The <audio> element is used as follows:

```
<p:cTn>
  <p:stCondLst>...
  <p:childTnLst>...
  <p:subTnLst>
    <p:audio>
      <p:cMediaNode vol="11000">...
    </p:audio>
  </p:subTnLst>
</p:cTn>
```

End example]

Parent Elements
childTnLst (§4.6.25); subTnLst (§4.6.78); tnLst (§4.6.87)

Child Elements	Subclause
cMediaNode (Common Media Node Properties)	§4.6.29

Attributes	Description
isNarration (Is Narration)	This attribute indicates whether the audio is a narration for the slide. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLMediaNodeAudio">
   <sequence>
      <element name="cMediaNode" type="CT_TLCommonMediaNodeData" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="isNarration" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.6.10 bg (Background)

This element is used to specify animating the background of an object.

[Example: Consider adding animation to the background of Shape Id 3. The <bg> tag can be used as follows:

```
<p:tgtEl>
  <p:spTgt spid="3">
    <p:bg/>
  </p:spTgt>
</p:tgtEl>
```

End example]

Parent Elements
spTgt (§4.6.72)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

4.6.11 bldAsOne (Build As One)

This element specifies in the build list to build the entire graphical object as one entity.

[Example: Consider having a graph appear as on entity as opposed to by category. The <bldAsOne> element should be used as follows:

```
<p:bldLst>
  <p:bldGraphic spid="4" grpId="0">
    <p:bldAsOne/>
  </p:bldGraphic>
</p:bldLst>
```

End example]

Parent Elements
bldGraphic (§4.6.13)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

4.6.12 bldDgm (Build Diagram)

This element specifies how to build the animation for a diagram.

[Example: Consider the following example where a chart is specified to be animated by category rather than as one entity. Thus, the bldChart element should be used as follows:

```
<p:bdldLst>
  <p:bldGraphic spid="4" grpId="0">
    <p:bldSub>
      <a:bldChart bld="category"/>
    </p:bldSub>
  </p:bldGraphic>
</p:bldLst>
```

End example]

Parent Elements
bldLst (§4.6.14)

Attributes	Description
bld (Diagram Build Types)	This attribute describes how the diagram will be built. The animation will animate the sub-elements in the container in the particular order defined by this attribute. The possible values for this attribute are defined by the ST_TLDiagramBuildType simple type (§4.8.37).
grpId (Group ID)	This attribute ties effects persisted in the animation to the build information. The attribute is used by the editor when changes to the build information are made. GroupIDs are unique for a given shape. They are not guaranteed to be unique IDs across all shapes on a slide. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
spid (Shape ID)	This attribute describes the shape to which the build applies. The possible values for this attribute are defined by the ST_ShapeID simple type (§5.1.12.55).
uiExpand (Expand UI)	This attribute describes the view option indicating if the build should be displayed expanded. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLBuildDiagram">
   <attributeGroup ref="AG_TLBuild"/>
   <attribute name="bld" type="ST_TLDiagramBuildType" use="optional" default="whole"/>
</complexType>
```

4.6.13 bldGraphic (Build Graphics)

This element specifies how to build a graphical element.

[Example: Consider having a chart graphical element appear as a whole as opposed to by a category. The <bldGraphic> element should be used as follows:

```
<p:bldLdst>
  <p:bldGraphic spid="3" grpId="0">
    <p:bldSub>
      <a:bldChart bld="category"/>
    </p:bldSub>
  </p:bldGraphic>
</p:bldLst>
```

End example]

Parent Elements
bldLst (§4.6.14)

Child Elements	Subclause
bldAsOne (Build As One)	§4.6.11
bldSub (Build Sub Elements)	§4.6.17

Attributes	Description
grpId (Group ID)	This attribute ties effects persisted in the animation to the build information. The attribute is used by the editor when changes to the build information are made. GroupIDs are unique for a given shape. They are not guaranteed to be unique IDs across all shapes on a slide. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
spid (Shape ID)	This attribute describes the shape to which the build applies. The possible values for this attribute are defined by the ST_ShapeID simple type (§5.1.12.55).
uiExpand (Expand UI)	This attribute describes the view option indicating if the build should be displayed expanded. The possible values for this attribute are defined by the XML Schema boolean datatype.

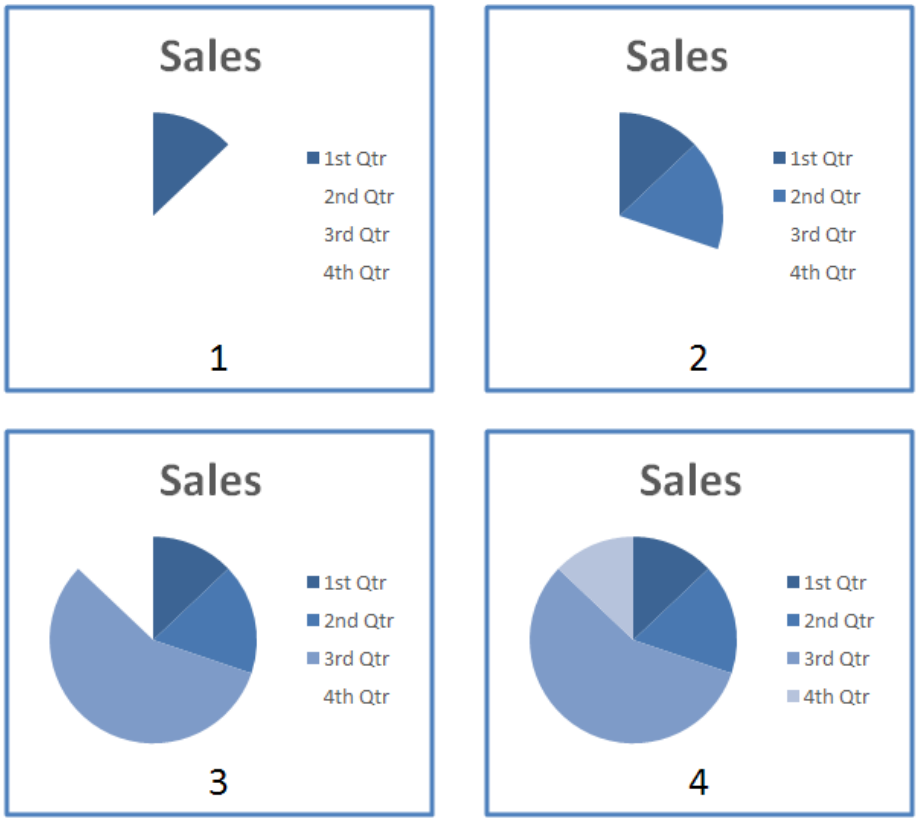
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLGraphicalObjectBuild">
   <choice minOccurs="1" maxOccurs="1">
      <element name="bldAsOne" type="CT_Empty"/>
      <element name="bldSub" type="a:CT_AnimationGraphicalObjectBuildProperties"/>
   </choice>
   <attributeGroup ref="AG_TLBuild"/>
</complexType>
```

4.6.14 bldLst (Build List)

This element specifies the list of graphic elements to build. This refers to how the different sub-shapes or sub-components of a object are displayed. The different objects that can have build properties are text, diagrams, and charts.

[Example: Consider animating a pie chart but based on category as shown below:



The <bldList> element should be used as follows:

```
<p:bldLst>
  <p:bldGraphic spid="1" grpId="0">
    <p:bldSub>
      <a:bldChart bld="category"/>
    </p:bldSub>
```

```
    </p:bldGraphic>
  </p:bldLst>
```

End example]

Parent Elements
timing (§4.4.1.44)

Child Elements	Subclause
bldDgm (Build Diagram)	§4.6.12
bldGraphic (Build Graphics)	§4.6.13
bldOleChart (Build Embedded Chart)	§4.6.15
bldP (Build Paragraph)	§4.6.16

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BuildList">
   <choice minOccurs="1" maxOccurs="unbounded">
      <element name="bldP" type="CT_TLBuildParagraph"/>
      <element name="bldDgm" type="CT_TLBuildDiagram"/>
      <element name="bldOleChart" type="CT_TLOleBuildChart"/>
      <element name="bldGraphic" type="CT_TLGraphicalObjectBuild"/>
   </choice>
</complexType>
```

4.6.15 bldOleChart (Build Embedded Chart)

This element describes animation an a embedded Chart.

[Example: Consider displaying animation on a embedded graphical chart. The <bldOleChart>element should be use as follows:

```
  <p:bldLst>
    <p:bldOleChart spid="1025" grpId="0"/>
  </p:bldLst>
```

End example]

Parent Elements
bldLst (§4.6.14)

Attributes	Description
animBg (Animate Background)	This attribute describes whether to animate the background of the shape.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema boolean datatype.
bld (Build)	This attribute describes how the diagram will be built. The animation will animate the sub-elements in the container in the particular order defined by this attribute. The possible values for this attribute are defined by the ST_TLOleChartBuildType simple type (§4.8.39).
grpId (Group ID)	This attribute ties effects persisted in the animation to the build information. The attribute is used by the editor when changes to the build information are made. GroupIDs are unique for a given shape. They are not guaranteed to be unique IDs across all shapes on a slide. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
spid (Shape ID)	This attribute describes the shape to which the build applies. The possible values for this attribute are defined by the ST_ShapeID simple type (§5.1.12.55).
uiExpand (Expand UI)	This attribute describes the view option indicating if the build should be displayed expanded. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLOleBuildChart">
   <attributeGroup ref="AG_TLBuild"/>
   <attribute name="bld" type="ST_TLOleChartBuildType" use="optional" default="allAtOnce"/>
   <attribute name="animBg" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

4.6.16 bldP (Build Paragraph)

This element specifies how to build paragraph level properties.

[Example: Consider having animation applied only to 1st level paragraphs. The <bldP> element should be used as follows:

```
<p:bldLst>
  <p:bldP spid="3" grpId="0" build="p"/>
</p:bldLst>
```

End example]

Parent Elements
bldLst (§4.6.14)

Child Elements	Subclause
tmplLst (Template effects)	§4.6.85

Attributes	Description
advAuto (Auto Advance Time)	This attribute specifies time after which to automatically advance the build to the next step. The possible values for this attribute are defined by the ST_TLTime simple type (§4.8.42).
animBg (Animate Background)	This attribute indicates whether to animate the background of the shape associated with the text. The possible values for this attribute are defined by the XML Schema boolean datatype.
autoUpdateAnimBg (Auto Update Animation Background)	This attribute indicates whether to automatically update the "animateBg" setting to true when the shape associated with the text has a fill or line. The possible values for this attribute are defined by the XML Schema boolean datatype.
bldLvl (Build Level)	This attribute describes the build level for the paragraph. It is only supported in paragraph type builds i.e the build attribute must also be set to "byParagraph" for this attribute to apply. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
build (Build Types)	This attribute describe the build types. The possible values for this attribute are defined by the ST_TLParaBuildType simple type (§4.8.40).
grpId (Group ID)	This attribute ties effects persisted in the animation to the build information. The attribute is used by the editor when changes to the build information are made. GroupIDs are unique for a given shape. They are not guaranteed to be unique IDs across all shapes on a slide. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
rev (Reverse)	This attribute is only supported in paragraph type builds. This specifies the direction of the build relative to the order of the elements in the container. When this is set to "true", the animations for the paragraphs will be persisted in reverse order to the order of the paragraphs themselves such that the last paragraph animates first. Default value is "false". The possible values for this attribute are defined by the XML Schema boolean datatype.
spid (Shape ID)	This attribute describes the shape to which the build applies.

Attributes	Description
	The possible values for this attribute are defined by the ST_ShapeID simple type (§5.1.12.55).
uiExpand (Expand UI)	This attribute describes the view option indicating if the build should be displayed expanded. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLBuildParagraph">
   <sequence>
      <element name="tmplLst" type="CT_TLTemplateList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attributeGroup ref="AG_TLBuild"/>
   <attribute name="build" type="ST_TLParaBuildType" use="optional" default="whole"/>
   <attribute name="bldLvl" type="xsd:unsignedInt" use="optional" default="1"/>
   <attribute name="animBg" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="autoUpdateAnimBg" type="xsd:boolean" default="true" use="optional"/>
   <attribute name="rev" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="advAuto" type="ST_TLTime" use="optional" default="indefinite"/>
</complexType>
```

4.6.17 bldSub (Build Sub Elements)

This element specifies the animation properties of a graphical object's sub-elements.

[Example: Consider applying animation to a graphical element consisting of a diagram. The <bldSub> element should be used as follows:

```
<p:bldLst>
  <p:bldGraphic spid="5" grpId="0">
    <p:bldSub>
      <a:bldDgm bld="one"/>
    </p:bldSub>
  </p:bldGraphic>
</p:bldLst>
```

End example]

Parent Elements
bldGraphic (§4.6.13)

Child Elements	Subclause
bldChart (Build Chart)	§5.1.2.1.1

Child Elements	Subclause
bldDgm (Build Diagram)	§5.1.2.1.2

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AnimationGraphicalObjectBuildProperties">
   <choice>
      <element name="bldDgm" type="CT_AnimationDgmBuildProperties"/>
      <element name="bldChart" type="CT_AnimationChartBuildProperties"/>
   </choice>
</complexType>
```

4.6.18 blinds (Blinds Slide Transition)

This element describes the a "Blinds" slide transition effect.

[Example: Consider we have a slide with a "blind" style transition. The <blinds> element should be used as follows:

```
<p:transition>
    <p:blinds/>
</p:transition>
```

End example]

Parent Elements
transition (§4.4.1.46)

Attributes	Description
dir (Transition Direction)	This attribute specifies a horizontal or vertical transition. The possible values for this attribute are defined by the ST_Direction simple type (§4.8.5).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OrientationTransition">
   <attribute name="dir" type="ST_Direction" use="optional" default="horz"/>
</complexType>
```

4.6.19 boolVal (Boolean Variant)

This element describes a Boolean Variant.

Parent Elements
progress (§4.6.57); to (§4.6.89); val (§4.6.92)

Attributes	Description
val (Value)	This attribute describes the boolean value for this element. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLAnimVariantBooleanVal">
   <attribute name="val" type="xsd:boolean" use="required"/>
</complexType>
```

4.6.20 by (By)

This element describes the relative offset value for the animation.

[Example: Consider a shape with an animation effect that scales the size of an object by 150%. The <by> element should be used as follows:

```
<p:animScale>
  <p:cBhvr>
    <p:cTn id="6" dur="2000" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="4"/>
    </p:tgtEl>
  </p:cBhvr>
  <p:by x="150000" y="150000"/>
</p:animScale>
```

End Example]

Parent Elements
animMotion (§4.6.4); animScale (§4.6.6)

Attributes	Description
x (X coordinate)	This attribute describes the X coordinate. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
y (Y coordinate)	This attribute describes the Y coordinate. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLPoint">
   <attribute name="x" type="a:ST_Percentage" use="required"/>
   <attribute name="y" type="a:ST_Percentage" use="required"/>
</complexType>
```

4.6.21 by (By)

This element describes the relative offset value for the color animation.

[Example: Consider a shape with a lightening emphasis animation applied to it. The <by> element should be used as follows:

```
<p:animClr clrSpc="hsl">
  <p:cBhvr>
    <p:cTn id="8" dur="500" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="4"/>
    </p:tgtEl>
    <p:attrNameLst>
      <p:attrName>stroke.color</p:attrName>
    </p:attrNameLst>
  </p:cBhvr>
  <p:by>
    <p:hsl h="0" s="0" l="0"/>
  </p:by>
</p:animClr>
```

End Example]

Parent Elements
animClr (§4.6.2)

Child Elements	Subclause
hsl (HSL)	§4.6.46
rgb (RGB)	§4.6.63

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLByAnimateColorTransform">
   <choice minOccurs="1" maxOccurs="1">
      <element name="rgb" type="CT_TLByRgbColorTransform"/>
      <element name="hsl" type="CT_TLByHslColorTransform"/>
   </choice>
</complexType>
```

4.6.22 cBhvr (Common Behavior)

This element describes the common behaviors of animations.

[Example: Consider trying to emphasize text within a shape by changing the size of its font. The <anim> element should be used as follows:

```
<p:anim to="1.5" calcmode="lin" valueType="num">
  <p:cBhvr override="childStyle">
    <p:cTn id="6" dur="2000" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="3">
        <p:txEl>
          <p:charRg st="4294967295" end="4294967295"/>
        </p:txEl>
      </p:spTgt>
    </p:tgtEl>
    <p:attrNameLst>
      <p:attrName>style.fontSize</p:attrName>
    </p:attrNameLst>
  </p:cBhvr>
</p:anim>
```

End example]

Parent Elements
anim (§4.6.1); animClr (§4.6.2); animEffect (§4.6.3); animMotion (§4.6.4); animRot (§4.6.5); animScale (§4.6.6); cmd (§4.6.28); set (§4.6.66)

Child Elements	Subclause
attrNameLst (Attribute Name List)	§4.6.8
cTn (Common Time Node Properties)	§4.6.33
tgtEl (Target Element)	§4.6.81

Attributes	Description
accumulate (Accumulate)	This attribute makes a repeating animation build with each iteration when set to "always." The possible values for this attribute are defined by the ST_TLBehaviorAccumulateType simple type (§4.8.31).
additive (Additive)	This attribute specifies how to apply the animation values to the original value for the property.

Attributes	Description
	The possible values for this attribute are defined by the ST_TLBehaviorAdditiveType simple type (§4.8.32).
by (By)	This attribute specifies a relative offset value for the animation.. The possible values for this attribute are defined by the XML Schema string datatype.
from (From)	This attribute specifies the starting value of the animation. The possible values for this attribute are defined by the XML Schema string datatype.
override (Override)	This attribute specifies how a behavior should override values of the attribute being animated on the target element. The "childStyle" will clear the attributes on the children contained inside the target element. The possible values for this attribute are defined by the ST_TLBehaviorOverrideType simple type (§4.8.33).
rctx (Runtime Context)	This attribute describes the runtime context of the animation. The currently-understood values are “PPT” and “IE.” This is used to specify the behavior used when animating in the PPT slideshow vs. IE HTML runtime. An example can be seen with the transparency effect. In IE, the transparency is animated as a bitmap, where in PPT, the style.opacity property of a shape is used to animate the transparency. The possible values for this attribute are defined by the XML Schema string datatype.
to (To)	This attribute specifies the ending value of the animation. The possible values for this attribute are defined by the XML Schema string datatype.
xfrmType (Transform Type)	This attribute specifies the type of transform to be used. The possible values for this attribute are defined by the ST_TLBehaviorTransformType simple type (§4.8.34).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLCommonBehaviorData">
   <sequence>
      <element name="cTn" type="CT_TLCommonTimeNodeData" minOccurs="1" maxOccurs="1"/>
      <element name="tgtEl" type="CT_TLTimeTargetElement" minOccurs="1" maxOccurs="1"/>
      <element name="attrNameLst" type="CT_TLBehaviorAttributeNameList" minOccurs="0"
        maxOccurs="1"/>
   </sequence>
   <attribute name="additive" type="ST_TLBehaviorAdditiveType" use="optional"/>
   <attribute name="accumulate" type="ST_TLBehaviorAccumulateType" use="optional"/>
   <attribute name="xfrmType" type="ST_TLBehaviorTransformType" use="optional"/>
   <attribute name="from" type="xsd:string" use="optional"/>
   <attribute name="to" type="xsd:string" use="optional"/>
   <attribute name="by" type="xsd:string" use="optional"/>
   <attribute name="rctx" type="xsd:string" use="optional"/>
   <attribute name="override" type="ST_TLBehaviorOverrideType" use="optional"/>
</complexType>
```

4.6.23 charRg (Character Range)

This element specifies animation on a character range defined by a start and end character position.

[Example: Consider animating the first word (characters 1 through 9) within a sentence. The <charRg> element should be used as follows:

```
<p:animMotion>
  <p:cBhvr>
    <p:cTn id="6" dur="2000" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="3">
        <p:txEl>
          <p:charRg st="0" end="9"/>
        </p:txEl>
      </p:spTgt>
    </p:tgtEl>
    <p:attrNameLst>
      <p:attrName>ppt_x</p:attrName>
      <p:attrName>ppt_y</p:attrName>
    </p:attrNameLst>
  </p:cBhvr>
</p:animMotion>
```

End example]

Parent Elements
txEl (§4.6.91)

Attributes	Description
end (End)	This attribute defines the end of the index range. The possible values for this attribute are defined by the ST_Index simple type (§4.8.7).
st (Start)	This attribute defines the start of the index range. The possible values for this attribute are defined by the ST_Index simple type (§4.8.7).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_IndexRange">
   <attribute name="st" type="ST_Index" use="required"/>
   <attribute name="end" type="ST_Index" use="required"/>
</complexType>
```

4.6.24 checker (Checker Slide Transition)

This element describes the Checker slide transition effect.

[Example: Consider we have a slide with a "checker" slide transition animated vertically. The <checker> element should be used as follows:

```
<p:transition>
    <p:checker dir="vert"/>
</p:transition>
```

End example]

Parent Elements
transition (§4.4.1.46)

Attributes	Description
dir (Transition Direction)	This attribute specifies a horizontal or vertical transition. The possible values for this attribute are defined by the ST_Direction simple type (§4.8.5).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OrientationTransition">
   <attribute name="dir" type="ST_Direction" use="optional" default="horz"/>
</complexType>
```

4.6.25 childTnLst (Children Time Node List)

This element describes the list of time nodes that have a fixed location in the timing tree based on their parent time node. The children's start time is defined relative to their parent time node’s start.

Parent Elements
cTn (§4.6.33)

Child Elements	Subclause
anim (Animate)	§4.6.1
animClr (Animate Color Behavior)	§4.6.2
animEffect (Animate Effect)	§4.6.3
animMotion (Animate Motion)	§4.6.4
animRot (Animate Rotation)	§4.6.5
animScale (Animate Scale)	§4.6.6
audio (Audio)	§4.6.9
cmd (Command)	§4.6.28
excl (Exclusive)	§4.6.40
par (Parallel Time Node)	§4.6.53
seq (Sequence Time Node)	§4.6.65
set (Set Time Node Behavior)	§4.6.66
video (Video)	§4.6.93

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TimeNodeList">
   <choice minOccurs="1" maxOccurs="unbounded">
      <element name="par" type="CT_TLTimeNodeParallel"/>
      <element name="seq" type="CT_TLTimeNodeSequence"/>
      <element name="excl" type="CT_TLTimeNodeExclusive"/>
      <element name="anim" type="CT_TLAnimateBehavior"/>
      <element name="animClr" type="CT_TLAnimateColorBehavior"/>
      <element name="animEffect" type="CT_TLAnimateEffectBehavior"/>
      <element name="animMotion" type="CT_TLAnimateMotionBehavior"/>
      <element name="animRot" type="CT_TLAnimateRotationBehavior"/>
      <element name="animScale" type="CT_TLAnimateScaleBehavior"/>
      <element name="cmd" type="CT_TLCommandBehavior"/>
      <element name="set" type="CT_TLSetBehavior"/>
      <element name="audio" type="CT_TLMediaNodeAudio"/>
      <element name="video" type="CT_TLMediaNodeVideo"/>
   </choice>
</complexType>
```

4.6.26 circle (Circle Slide Transition)

This element describes the Circle transition effect..

[Example: Consider we have a slide with a "circle" style transition. The <circle> element should be used as follows:

```
<p:transition>
    <p:circle/>
</p:transition>
```

End example]

Parent Elements
transition (§4.4.1.46)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

4.6.27 clrVal (Color Value)

This element describes the color variant. This is used to specify a color that we will use for animating the color property of an object.

[Example: Consider trying to emphasize text within a shape by changing the color its font.

```
<p:set>
  <p:cBhvr override="childStyle">
    ...
  </p:cBhvr>
  <p:to>
    <p:clrVal>
      <a:schemeClr val="accent2"/>
    </p:clrVal>
  </p:to>
</p:set>
```

End example]

Parent Elements
progress (§4.6.57); to (§4.6.89); val (§4.6.92)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

4.6.28 cmd (Command)

This element describes the several non-durational type of commands that can be executed within a timeline. This can be used to send events, call functions on elements, and send verbs to embedded objects. For example “Object Action” effects for Embedded objects and Media commands for sounds/movies such as "PlayFrom(0.0)" and "togglePause".

Parent Elements
childTnLst (§4.6.25); subTnLst (§4.6.78); tnLst (§4.6.87)

Child Elements	Subclause
cBhvr (Common Behavior)	§4.6.22

Attributes	Description
cmd (Command)	This is a string used to define the actual command. When an event is specified, in IE, the runtime will register the specific event and make sure that time conditions (begin/end/etc.) elsewhere on the page which listen for the event get hooked up correctly. The possible values for this attribute are defined by the XML Schema string datatype.
type (Command Type)	This attribute is required and defines the type of command. When the value is set to "event", the "cmd" attribute string is used to send out an event to the target element. When the value is set to "call", the "cmd" string is used to call a method or function on the target element. When the value is set to "verb", the "cmd" string is treated as an verb on the element. When the value is set to "script", the cmd will be interpreted in the IE runtime as a string containing javascript statements. Note that this may cause a runtime script error if the script is invalid at the time it's called. If the targetElement is not found, the command will not execute. The possible values for this attribute are defined by the ST_TLCommandType simple

Attributes	Description
	type (§4.8.36).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLCommandBehavior">
   <sequence>
      <element name="cBhvr" type="CT_TLCommonBehaviorData" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute type="ST_TLCommandType" name="type" use="optional"/>
   <attribute name="cmd" type="xsd:string" use="optional"/>
</complexType>
```

4.6.29 cMediaNode (Common Media Node Properties)

This element is used to describe behavior of media elements, such as sound or movies, in an animation.

[Example: Consider a shape with a sound effect attached to its animation. The <cMediaNode> element should be used as follows:

```
<p:audio>
  <p:cMediaNode mute="1">
    <p:cTn display="0" masterRel="sameClick">
      <p:stCondLst>...
      <p:endCondLst>..
    </p:cTn>
    <p:tgtEl>..
  </p:cMediaNode>
</p:audio>
```

End Example]

Parent Elements
audio (§4.6.9); video (§4.6.93)

Child Elements	Subclause
cTn (Common Time Node Properties)	§4.6.33
tgtEl (Target Element)	§4.6.81

Attributes	Description
mute (Mute)	This attribute describes whether the media should be mute. The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description
numSld (Number of Slides)	This attribute describes the numbers of slides across which the media should play. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
showWhenStopped (Show When Stopped)	This attribute describes whether the media should be displayed when it is stopped. The possible values for this attribute are defined by the XML Schema boolean datatype.
vol (Volume)	This attribute describes the volume of the media element. The possible values for this attribute are defined by the ST_PositiveFixedPercentage simple type (§5.1.12.45).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLCommonMediaNodeData">
   <sequence>
      <element name="cTn" type="CT_TLCommonTimeNodeData" minOccurs="1" maxOccurs="1"/>
      <element name="tgtEl" type="CT_TLTimeTargetElement" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="vol" type="a:ST_PositiveFixedPercentage" default="50000" use="optional"/>
   <attribute name="mute" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="numSld" type="xsd:unsignedInt" use="optional" default="1"/>
   <attribute name="showWhenStopped" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

4.6.30 comb (Comb Slide Transition)

This element describes a comb slide transition effect.

[Example: Consider we have a slide with a comb slide transition animated vertically. The <comb> element should be used as follows:

```
<p:transition>
    <p:comb/>
</p:transition>
```

End example]

Parent Elements
transition (§4.4.1.46)

Attributes	Description
dir (Transition	This attribute specifies a horizontal or vertical transition.

Attributes	Description
Direction)	The possible values for this attribute are defined by the ST_Direction simple type (§4.8.5).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OrientationTransition">
   <attribute name="dir" type="ST_Direction" use="optional" default="horz"/>
</complexType>
```

4.6.31 cond (Condition)

This element specifies conditions on time nodes in a timeline. It is used within a list of start condition or list of end condition elements.

[Example: For example, suppose we have a shape with a two second delay after the animation is started.

```
<p:cTn>
  <p:stCondLst>
    <p:cond delay="2000"/>
  </p:stCondLst>
  <p:childTnLst>
    <p:set>...
    <p:animEffect transition="in" filter="blinds(horizontal)">
      <p:cBhvr>
        <p:cTn id="7" dur="1000"/>
        <p:tgtEl>
          <p:spTgt spid="4"/>
        </p:tgtEl>
      </p:cBhvr>
    </p:animEffect>
  </p:childTnLst>
</p:cTn>
```

End Example]

Parent Elements
endCondLst (§4.6.37); nextCondLst (§4.6.51); prevCondLst (§4.6.55); stCondLst (§4.6.73)

Child Elements	Subclause
rtn (Runtime Node Trigger Choice)	§4.6.64
tgtEl (Target Element)	§4.6.81
tn (Time Node)	§4.6.86

Attributes	Description
delay (Trigger Delay)	This attribute describes the delay after an animation is triggered. The possible values for this attribute are defined by the ST_TLTime simple type (§4.8.42).
evt (Trigger Event)	This attribute describes the event that triggers an animation. The possible values for this attribute are defined by the ST_TLTriggerEvent simple type (§4.8.52).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLTimeCondition">
   <choice minOccurs="0" maxOccurs="1">
      <element name="tgtEl" type="CT_TLTimeTargetElement"/>
      <element name="tn" type="CT_TLTriggerTimeNodeID"/>
      <element name="rtn" type="CT_TLTriggerRuntimeNode"/>
   </choice>
   <attribute name="evt" use="optional" type="ST_TLTriggerEvent"/>
   <attribute name="delay" type="ST_TLTime" use="optional"/>
</complexType>
```

4.6.32 cover (Cover Slide Transition)

This element describes a cover slide transition effect.

[Example: Consider we have a slide with a cover slide transition animated vertically. The <cover> element should be used as follows:

```
<p:transition>
    <p:cover/>
</p:transition>
```

End example]

Parent Elements
transition (§4.4.1.46)

Attributes	Description
dir (Direction)	This attribute specifies if the direction of the transition. The possible values for this attribute are defined by the ST_TransitionEightDirectionType simple type (§4.8.55).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EightDirectionTransition">
   <attribute name="dir" type="ST_TransitionEightDirectionType" use="optional" default="l"/>
</complexType>
```

4.6.33 cTn (Common Time Node Properties)

This element describes the properties that are common for time nodes.

Parent Elements
cBhvr (§4.6.22); cMediaNode (§4.6.29); excl (§4.6.40); par (§4.6.53); seq (§4.6.65)

Child Elements	Subclause
childTnLst (Children Time Node List)	§4.6.25
endCondLst (End Conditions List)	§4.6.37
endSync (EndSync)	§4.6.39
iterate (Iterate)	§4.6.49
stCondLst (Start Conditions List)	§4.6.73
subTnLst (Sub-TimeNodes List)	§4.6.78

Attributes	Description
accel (Acceleration)	This attribute describes the percentage of specified duration over which the element's time will take to accelerate from 0 up to the "run rate." The possible values for this attribute are defined by the ST_PositiveFixedPercentage simple type (§5.1.12.45).
afterEffect (After Effect)	This attribute specifies whether there is an after effect applied to the time node. The possible values for this attribute are defined by the XML Schema boolean datatype.
autoRev (Auto Reverse)	This attribute describes whether to automatically play the animation in reverse after playing it in the forward direction. The possible values for this attribute are defined by the XML Schema boolean datatype.
bldLvl (Build level)	This attribute describes the build level of the animation. The possible values for this attribute are defined by the XML Schema int datatype.
decel (Deceleration)	This attribute describes the percentage of specified duration over which the element's time will take to decelerate from the "run rate" down to 0. The possible values for this attribute are defined by the ST_PositiveFixedPercentage simple type (§5.1.12.45).

Attributes	Description
display (Display)	This attribute describes whether the state of the time node is visible or hidden. The possible values for this attribute are defined by the XML Schema boolean datatype.
dur (Duration)	This attribute describes the duration of the time node, expressed as unit time. The possible values for this attribute are defined by the ST_TLTime simple type (§4.8.42).
evtFilter (Event Filter)	This attribute describes the event filter for this time node. The possible values for this attribute are defined by the XML Schema string datatype.
fill (Fill)	This attribute describes the fill type for the time node. The possible values for this attribute are defined by the ST_TLTimeNodeFillType simple type (§4.8.45).
grpId (Group ID)	This attribute describes the Group ID of the time node. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
id (ID)	This attribute specifies the identifier for the timenode. The possible values for this attribute are defined by the ST_TLTimeNodeID simple type (§4.8.46).
masterRel (Master Relation)	This attribute specifies how the time node plays back relative to its master time node. The possible values for this attribute are defined by the ST_TLTimeNodeMasterRelation simple type (§4.8.47).
nodePh (Node Placeholder)	This attribute describes whether this node is a placeholder. The possible values for this attribute are defined by the XML Schema boolean datatype.
nodeType (Node Type)	This attribute specifies the type of time node. The possible values for this attribute are defined by the ST_TLTimeNodeType simple type (§4.8.51).
presetClass (Preset Types)	This attribute descries the class of effect in which it belongs. The possible values for this attribute are defined by the ST_TLTimeNodePresetClassType simple type (§4.8.48).
presetID (Preset ID)	This attribute describes the preset identifier for the time node. The possible values for this attribute are defined by the XML Schema int datatype.
presetSubtype	This attribute is a bitflag that specifies a direction or some other attribute of the effect.

Attributes	Description
(Preset SubType)	For example it can be set to specify a “From Bottom” for the Fly In effect, or “Bold” for the Change Font Style effect. The possible values for this attribute are defined by the XML Schema int datatype.
repeatCount (Repeat Count)	This attribute describes the number of times the element should repeat, in units of thousandths. The possible values for this attribute are defined by the ST_TLTime simple type (§4.8.42).
repeatDur (Repeat Duration)	This attribute describes the amount of time over which the element should repeat. If absent, the attribute is taken to be the same as the specified duration. The possible values for this attribute are defined by the ST_TLTime simple type (§4.8.42).
restart (Restart)	This attribute specifies if a node is to restart when it completes its action. The possible values for this attribute are defined by the ST_TLTimeNodeRestartType simple type (§4.8.49).
spd (Speed)	This attribute specifies the percentage by which to speed up (or slow down) the timing. If negative, the timing will be reversed. Example: if speed is 200% and the specified duration is 10 seconds, the actual duration will be 5 seconds. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
syncBehavior (Synchronization Behavior)	This attribute specifies how the time node synchronizes to its group. The possible values for this attribute are defined by the ST_TLTimeNodeSyncType simple type (§4.8.50).
tmFilter (Time Filter)	This attribute specifies the time filter for the time node. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLCommonTimeNodeData">
    <sequence>
        <element name="stCondLst" type="CT_TLTimeConditionList" minOccurs="0" maxOccurs="1"/>
        <element name="endCondLst" type="CT_TLTimeConditionList" minOccurs="0" maxOccurs="1"/>
        <element name="endSync" type="CT_TLTimeCondition" minOccurs="0" maxOccurs="1"/>
        <element name="iterate" type="CT_TLIterateData" minOccurs="0" maxOccurs="1"/>
        <element name="childTnLst" type="CT_TimeNodeList" minOccurs="0" maxOccurs="1"/>
        <element name="subTnLst" type="CT_TimeNodeList" minOccurs="0" maxOccurs="1"/>
    </sequence>
    <attribute name="id" type="ST_TLTimeNodeID" use="optional"/>
    <attribute name="presetID" type="xsd:int" use="optional"/>
    <attribute name="presetClass" type="ST_TLTimeNodePresetClassType" use="optional"/>
    <attribute name="presetSubtype" type="xsd:int" use="optional"/>
    <attribute name="dur" type="ST_TLTime" use="optional"/>
    <attribute name="repeatCount" type="ST_TLTime" use="optional" default="1000"/>
    <attribute name="repeatDur" type="ST_TLTime" use="optional"/>
    <attribute name="spd" type="a:ST_Percentage" use="optional" default="100000"/>
    <attribute name="accel" type="a:ST_PositiveFixedPercentage" use="optional" default="0"/>
    <attribute name="decel" type="a:ST_PositiveFixedPercentage" use="optional" default="0"/>
    <attribute name="autoRev" type="xsd:boolean" use="optional" default="false"/>
    <attribute name="restart" type="ST_TLTimeNodeRestartType" use="optional"/>
    <attribute name="fill" type="ST_TLTimeNodeFillType" use="optional"/>
    <attribute name="syncBehavior" type="ST_TLTimeNodeSyncType" use="optional"/>
    <attribute name="tmFilter" type="xsd:string" use="optional"/>
    <attribute name="evtFilter" type="xsd:string" use="optional"/>
    <attribute name="display" type="xsd:boolean" use="optional"/>
    <attribute name="masterRel" type="ST_TLTimeNodeMasterRelation" use="optional"/>
    <attribute name="bldLvl" type="xsd:int" use="optional"/>
    <attribute name="grpId" type="xsd:unsignedInt" use="optional"/>
    <attribute name="afterEffect" type="xsd:boolean" use="optional"/>
    <attribute name="nodeType" type="ST_TLTimeNodeType" use="optional"/>
    <attribute name="nodePh" type="xsd:boolean" use="optional"/>
</complexType>
```

4.6.34 cut (Cut Slide Transition)

This element describes a cut slide transition effect.

[Example: Consider we have a slide with a cut slide transition animated vertically. The <cut>element should be used as follows:

```
<p:transition>
    <p:cut/>
</p:transition>
```

End example]

Parent Elements
transition (§4.4.1.46)

Attributes	Description
thruBlk (Transition Through Black)	This attribute specifies if the transition will start from a black screen (and then transition the new slide over black). The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OptionalBlackTransition">
   <attribute name="thruBlk" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.6.35 diamond (Diamond Slide Transition)

This element describes a diamond slide transition effect.

[Example: Consider we have a slide with a diamond slide transition animated vertically. The <diamond> element should be used as follows:

```
<p:transition>
    <p:diamond/>
</p:transition>
```

End example]

Parent Elements
transition (§4.4.1.46)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

4.6.36 dissolve (Dissolve Slide Transition)

This element describes a dissolve slide transition effect.

[Example: Consider we have a slide with a dissolve slide transition animated vertically. The <dissolve> element should be used as follows:

```
<p:transition>
    <p:dissolve/>
</p:transition>
```

End example]

Parent Elements
transition (§4.4.1.46)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

4.6.37 endCondLst (End Conditions List)

This element describes a list of the end conditions that must be met in order to stop the time node.

[Example: Consider a shape a shape with an audio attached to the animation. The <endCondList> element should be used as follows to specifies when the sound is done:

```
<p:audio>
  <p:cMediaNode>
    <p:cTn display="0" masterRel="sameClick">
      <p:stCondLst>...
      <p:endCondLst>
        <p:cond evt="onStopAudio" delay="0">
          <p:tgtEl>
            <p:sldTgt/>
          </p:tgtEl>
        </p:cond>
      </p:endCondLst>
    </p:cTn>
    <p:tgtEl>...
  </p:cMediaNode>
</p:audio>
```

End Example]

Parent Elements
cTn (§4.6.33)

Child Elements	Subclause
cond (Condition)	§4.6.31

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLTimeConditionList">
   <sequence>
      <element name="cond" type="CT_TLTimeCondition" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

4.6.38 endSnd (Stop Sound Action)

This element will stops all previous sounds during a slide transition.

[Example: Consider a slide transition that stops all previous sounds. The<endSnd> element should be used as follows:

```
<p:transition>
    <p:sndAc>
        <p:endSnd/>
    </p:sndAc>
</p:transition>
```

End Example]

Parent Elements
sndAc (§4.6.69)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

4.6.39 endSync (EndSync)

This element is used to synchronizes the stopping of parallel elements in the timing tree. It is used on interactive timeline sequences to specify that the interactive sequence's duration ends when all of the child timenodes have ended. It is also used to make interactive sequences restart-able (so that the entire interactive sequence can be repeated if the trigger object is clicked on repeatedly).

[Example: Consider a shape with a fill change animation. The <endSync> element should be used as follows:

```
<p:seq concurrent="1" nextAc="seek">
  <p:cTn>
    <p:stCondLst/>
    <p:endSync evt="end" delay="0">
      <p:rtn val="all"/>
    </p:endSync>
    <p:childTnLst/>
  </p:cTn>
  <p:nextCondLst/>
</p:seq>
```

End Example]

Parent Elements
cTn (§4.6.33)

Child Elements	Subclause
rtn (Runtime Node Trigger Choice)	§4.6.64

Child Elements	Subclause
tgtEl (Target Element)	§4.6.81
tn (Time Node)	§4.6.86

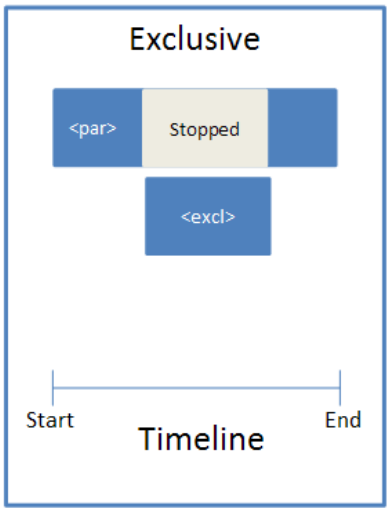
Attributes	Description
delay (Trigger Delay)	This attribute describes the delay after an animation is triggered. The possible values for this attribute are defined by the ST_TLTime simple type (§4.8.42).
evt (Trigger Event)	This attribute describes the event that triggers an animation. The possible values for this attribute are defined by the ST_TLTriggerEvent simple type (§4.8.52).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLTimeCondition">
   <choice minOccurs="0" maxOccurs="1">
      <element name="tgtEl" type="CT_TLTimeTargetElement"/>
      <element name="tn" type="CT_TLTriggerTimeNodeID"/>
      <element name="rtn" type="CT_TLTriggerRuntimeNode"/>
   </choice>
   <attribute name="evt" use="optional" type="ST_TLTriggerEvent"/>
   <attribute name="delay" type="ST_TLTime" use="optional"/>
</complexType>
```

4.6.40 excl (Exclusive)

This element describes the Exclusive time node. This time node is used to pause all other timelines when it is activated. Conceptually it can be though of as follows:



Parent Elements
childTnLst (§4.6.25); subTnLst (§4.6.78); tnLst (§4.6.87)

Child Elements	Subclause
cTn (Common Time Node Properties)	§4.6.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLTimeNodeExclusive">
   <sequence>
      <element name="cTn" type="CT_TLCommonTimeNodeData" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.6.41 fade (Fade Slide Transition)

This element describes a fade slide transition effect.

[Example: Consider we have a slide with a fade slide transition animated vertically. The <fade> element should be used as follows:

```
<p:transition>
    <p:fade/>
</p:transition>
```

End example]

Parent Elements
transition (§4.4.1.46)

Attributes	Description
thruBlk (Transition Through Black)	This attribute specifies if the transition will start from a black screen (and then transition the new slide over black). The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OptionalBlackTransition">
   <attribute name="thruBlk" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.6.42 fltVal (Float Value)

This element describes a float variant.

Parent Elements
progress (§4.6.57); to (§4.6.89); val (§4.6.92)

Attributes	Description
val (Value)	This attribute specifies the value of this element as a floating point. The possible values for this attribute are defined by the XML Schema float datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLAnimVariantFloatVal">
   <attribute name="val" type="xsd:float" use="required"/>
</complexType>
```

4.6.43 from (From)

This element specifies an x/y co-ordinate to start the animation from.

[Example: Consider a shape with an animation sequence that needs to start at a certain coordinate. The <from> element should be used as follows:

```
<p:animScale>
  <p:cBhvr>
    <p:cTn>…
    <p:tgtEl>
      <p:spTgt spid="4"/>
    </p:tgtEl>
  </p:cBhvr>
  <p:from x="100000" y="100000"/>
  <p:to x="80000" y="100000"/>
</p:animScale>
```

End Example]

Parent Elements
animMotion (§4.6.4); animScale (§4.6.6)

Attributes	Description
x (X coordinate)	This attribute describes the X coordinate. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
y (Y coordinate)	This attribute describes the Y coordinate.

Attributes	Description
	The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLPoint">
   <attribute name="x" type="a:ST_Percentage" use="required"/>
   <attribute name="y" type="a:ST_Percentage" use="required"/>
</complexType>
```

4.6.44 from (From)

This element is used to specify the starting color of the target element.

[Example: Consider a shape with an animation fill change from one accent color to another. The <from> element should be used as follows:

```
<p:animClr clrSpc="rgb" dir="cw">
  <p:cBhvr>
    <p:cTn id="6" dur="2000" fill="hold"/>
    <p:tgtEl>...
    <p:attrNameLst>
      <p:attrName>fillcolor</p:attrName>
    </p:attrNameLst>
  </p:cBhvr>
  <p:from>
    <a:schemeClr val="accent3"/>
  </p:from>
  <p:to>
    <a:schemeClr val="accent2"/>
  </p:to>
</p:animClr>
```

End Example]

Parent Elements
animClr (§4.6.2)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30

Child Elements	Subclause
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

4.6.45 graphicEl (Graphic Element)

This element specifies a graphical element which to animate.

[Example: Consider a diagram with an animation effect applied to it. The <graphicEl> element should be used as follows:

```
<p:set>
  <p:cBhvr>
    <p:cTn id="6" dur="1" fill="hold">...
    <p:tgtEl>
      <p:spTgt spid="4">
        <p:graphicEl>
          <a:dgm id="{87C2C707-C3F4-4E81-A967-A8B8AE13E575}"/>
        </p:graphicEl>
      </p:spTgt>
    </p:tgtEl>
    <p:attrNameLst>...
  </p:cBhvr>
  <p:to>
</p:set>
```

End Example]

Parent Elements
spTgt (§4.6.72)

Child Elements	Subclause
chart (Chart to Animate)	§5.1.2.1.3
dgm (Diagram to Animate)	§5.1.2.1.12

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AnimationElementChoice">
   <choice minOccurs="1" maxOccurs="1">
      <element name="dgm" type="CT_AnimationDgmElement"/>
      <element name="chart" type="CT_AnimationChartElement"/>
   </choice>
</complexType>
```

4.6.46 hsl (HSL)

This element specifies an incremental HSL (Hue, Saturation, Lightness) value to add to a color animation.

[Example: Consider a shape with a lightening emphasis animation. The <hsl> element should be used as follows:

```
<p:animClr clrSpc="hsl">
  <p:cBhvr>
    <p:cTn id="8" dur="500" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="4"/>
    </p:tgtEl>
    <p:attrNameLst>
      <p:attrName>stroke.color</p:attrName>
    </p:attrNameLst>
  </p:cBhvr>
  <p:by>
    <p:hsl h="0" s="0" l="0"/>
  </p:by>
</p:animClr>
```

End Example]

Parent Elements
by (§4.6.21)

Attributes	Description
h (Hue)	Specifies hue as an angle. The valid values range from [0, 360] degrees. The possible values for this attribute are defined by the ST_Angle simple type (§5.1.12.3).
l (Lightness)	Specifies a lightness as fixed percentage in 1000ths of a percent. The valid values range from [-100%, 100%]. The possible values for this attribute are defined by the ST_FixedPercentage simple type (§5.1.12.22).
s (Saturation)	Specifies a saturation as a fixed percentage in 1000ths of a percent. The valid values range from [-100%, 100%].

Attributes	Description
	The possible values for this attribute are defined by the ST_FixedPercentage simple type (§5.1.12.22).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLByHslColorTransform">
   <attribute name="h" type="a:ST_Angle" use="required"/>
   <attribute name="s" type="a:ST_FixedPercentage" use="required"/>
   <attribute name="l" type="a:ST_FixedPercentage" use="required"/>
</complexType>
```

4.6.47 inkTgt (Ink Target)

This element specifies an animation target element that is represented by a sub-shape in a legacy graphical object.

[Example: Consider an ink diagram with an animation blinds transition effect applied to it. The <inkTgt> element should be used as follows:

```
<p:animEffect transition="in" filter="blinds(horizontal)">
  <p:cBhvr>
    <p:cTn id="7" dur="500"/>
    <p:tgtEl>
      <p:inkTgt spid="_x0000_s2057"/>
    </p:tgtEl>
  </p:cBhvr>
</p:animEffect>
```

End Example]

Parent Elements
tgtEl (§4.6.81)

Attributes	Description
spid (Shape ID)	This attribute specifies the shape identifier. The possible values for this attribute are defined by the ST_ShapeID simple type (§5.1.12.55).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLSubShapeId">
   <attribute name="spid" type="a:ST_ShapeID" use="required"/>
</complexType>
```

4.6.48 intVal (Integer)

This element describes an integer variant.

Parent Elements
progress (§4.6.57); to (§4.6.89); val (§4.6.92)

Attributes	Description
val (Value)	This attribute specifies the value of this element as a floating point. The possible values for this attribute are defined by the XML Schema int datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLAnimVariantIntegerVal">
   <attribute name="val" type="xsd:int" use="required"/>
</complexType>
```

4.6.49 iterate (Iterate)

This element specifies how the animation should be successively applied to sub elements of the target element for a repeated effect. It can be applied to contained timing and animation structures over the letters, words, or shapes within a target element.

[Example: Consider a text animation where the words appear letter by letter. The <iterate> element should be used as follows:

```
<p:par>
  <p:cTn id="1" >
    <p:stCondLst>...
    <p:iterate type="lt">
      <p:tmPct val="10000"/>
    </p:iterate>
    <p:childTnLst>...
  </p:cTn>
</p:par>
```

End Example]

Parent Elements
cTn (§4.6.33)

Child Elements	Subclause
tmAbs (Time Absolute)	§4.6.82

Child Elements	Subclause
tmPct (Time Percentage)	§4.6.83

Attributes	Description
backwards (Backwards)	This attribute specifies whether to go backwards in the timeline to the previous node. The possible values for this attribute are defined by the XML Schema boolean datatype.
type (Iterate Type)	This attribute specifies the iteration behavior and applies it to each letter, word or shape within a container element. Valid values are by word, by letter, or by element. If there is no text or block elements such as shapes within the container or a single word, letter, or shape (depending on iterate type) then no iteration will happen and the behavior is applied to the element itself instead. The possible values for this attribute are defined by the ST_IterateType simple type (§4.8.8).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLIterateData">
   <choice minOccurs="1" maxOccurs="1">
      <element name="tmAbs" type="CT_TLIterateIntervalTime"/>
      <element name="tmPct" type="CT_TLIterateIntervalPercentage"/>
   </choice>
   <attribute name="type" type="ST_IterateType" use="optional" default="el"/>
   <attribute name="backwards" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.6.50 newsflash (Newsflash Slide Transition)

This element describes a newsflash slide transition effect.

[Example: Consider we have a slide with a newsflash slide transition animated vertically. The <newsflash> element should be used as follows:

```
<p:transition>
    <p:newsflash/>
</p:transition>
```

End example]

Parent Elements
transition (§4.4.1.46)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

4.6.51 nextCondLst (Next Conditions List)

This element describes a list of conditions that must be met to advance to the next animation sequence.

[Example: Consider a shape with a text emphasis changing the size of its font.

```
<p:seq concurrent="1" nextAc="seek">
  <p:cTn id="2" dur="indefinite" nodeType="mainSeq">...
  <p:prevCondLst>...
  <p:nextCondLst>
    <p:cond evt="onNext" delay="0">
      <p:tgtEl>
        <p:sldTgt/>
      </p:tgtEl>
    </p:cond>
  </p:nextCondLst>
</p:seq>
```

End example]

Parent Elements
seq (§4.6.65)

Child Elements	Subclause
cond (Condition)	§4.6.31

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLTimeConditionList">
   <sequence>
      <element name="cond" type="CT_TLTimeCondition" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

4.6.52 oleChartEl (Embedded Chart Element)

This element specifies the subelement of an embedded chart to animate.

[Example: Consider an embedded Chart with a entrance animation effect applied to each of the graph's categories. The <oldChartEl> element should be used as follows:

```
<p:animEffect transition="in" filter="blinds(horizontal)">
  <p:cBhvr>
    <p:cTn id="12" dur="500"/>
```

```
      <p:tgtEl>
        <p:spTgt spid="19460">
          <p:oleChartEl type="category" lvl="1"/>
        </p:spTgt>
      </p:tgtEl>
    </p:cBhvr>
  </p:animEffect>
```

End Example]

Parent Elements
spTgt (§4.6.72)

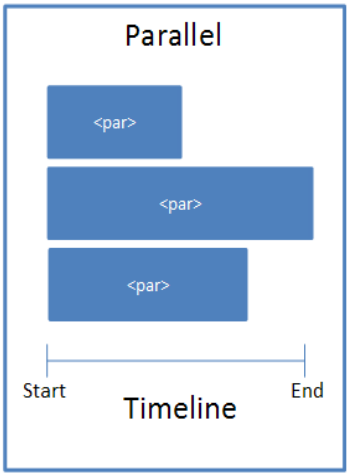
Attributes	Description
lvl (Level)	This attribute describes the element levels to animate. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
type (Type)	This attribute specifies how to chart should be built during its animation. The possible values for this attribute are defined by the ST_TLChartSubelementType simple type (§4.8.35).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLOleChartTargetElement">
   <attribute name="type" type="ST_TLChartSubelementType" use="required"/>
   <attribute name="lvl" type="xsd:unsignedInt" use="optional" default="0"/>
</complexType>
```

4.6.53 par (Parallel Time Node)

This element describes the Parallel time node which can be activated along with other parallel time node containers. Conceptually it can be thought of as follows:



[Example: Consider a simple animation with a blind entrance. The <par> element should be used as follows:

```
<p:timing>
  <p:tnLst>
    <p:par>
      <p:cTn id="1" dur="indefinite" restart="never" nodeType="tmRoot">
        <p:childTnLst>
          <p:seq concurrent="1" nextAc="seek">
            ...
          </p:seq>
        </p:childTnLst>
      </p:cTn>
    </p:par>
  </p:tnLst>
</p:timing>
```

End Example]

Parent Elements
childTnLst (§4.6.25); subTnLst (§4.6.78); tnLst (§4.6.87)

Child Elements	Subclause
cTn (Common Time Node Properties)	§4.6.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLTimeNodeParallel">
   <sequence>
      <element name="cTn" type="CT_TLCommonTimeNodeData" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.6.54 plus (Plus Slide Transition)

This element describes a plus slide transition effect.

[Example: Consider we have a slide with a plus slide transition animated vertically. The <plus> element should be used as follows:

```
<p:transition>
    <p:plus/>
</p:transition>
```

End example]

Parent Elements
transition (§4.4.1.46)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

4.6.55 prevCondLst (Previous Conditions List)

This element describes a list of conditions that must be met in order to go backwards in an animation sequence.

[Example: Consider trying to emphasize text within a shape by changing the size of its font.

```
<p:seq concurrent="1" nextAc="seek">
  <p:cTn id="2" dur="indefinite" nodeType="mainSeq">
  </p:cTn>
  <p:prevCondLst>
    <p:cond evt="onPrev" delay="0">
      <p:tgtEl>
        <p:sldTgt/>
      </p:tgtEl>
    </p:cond>
  </p:prevCondLst>
  <p:nextCondLst>
  </p:nextCondLst>
</p:seq>
```

End example]

Parent Elements
seq (§4.6.65)

Child Elements	Subclause

Child Elements	Subclause
cond (Condition)	§4.6.31

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLTimeConditionList">
   <sequence>
      <element name="cond" type="CT_TLTimeCondition" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

4.6.56 pRg (Paragraph Text Range)

This element specifies a text range to animate based on starting and ending paragraph number.

[Example: Consider an animation entrance of the first 3 text paragraphs. The <pRg> element should be used as follows:

```
<p:animEffect transition="in" filter="checkerboard(across)">
  <p:cBhvr>
    <p:cTn id="12" dur="500"/>
    <p:tgtEl>
      <p:spTgt spid="3">
        <p:txEl>
          <p:pRg st="0" end="2"/>
        </p:txEl>
      </p:spTgt>
    </p:tgtEl>
  </p:cBhvr>
</p:animEffect>
```

End Example]

Parent Elements
txEl (§4.6.91)

Attributes	Description
end (End)	This attribute defines the end of the index range. The possible values for this attribute are defined by the ST_Index simple type (§4.8.7).
st (Start)	This attribute defines the start of the index range. The possible values for this attribute are defined by the ST_Index simple type (§4.8.7).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_IndexRange">
   <attribute name="st" type="ST_Index" use="required"/>
   <attribute name="end" type="ST_Index" use="required"/>
</complexType>
```

4.6.57 progress (Progress)

This element defines the animation's overall progress that remains throughout the entire animation.

Parent Elements
animEffect (§4.6.3)

Child Elements	Subclause
boolVal (Boolean Variant)	§4.6.19
clrVal (Color Value)	§4.6.27
fltVal (Float Value)	§4.6.42
intVal (Integer)	§4.6.48
strVal (String Value)	§4.6.75

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLAnimVariant">
   <choice minOccurs="1" maxOccurs="1">
      <element name="boolVal" type="CT_TLAnimVariantBooleanVal"/>
      <element name="intVal" type="CT_TLAnimVariantIntegerVal"/>
      <element name="fltVal" type="CT_TLAnimVariantFloatVal"/>
      <element name="strVal" type="CT_TLAnimVariantStringVal"/>
      <element name="clrVal" type="a:CT_Color"/>
   </choice>
</complexType>
```

4.6.58 pull (Pull Slide Transition)

This element describes a pull slide transition effect.

[Example: Consider we have a slide with a pull slide transition. The <pull> element should be used as follows:

```
<p:transition>
    <p:pull/>
</p:transition>
```

End example]

Parent Elements
transition (§4.4.1.46)

Attributes	Description
dir (Direction)	This attribute specifies if the direction of the transition. The possible values for this attribute are defined by the ST_TransitionEightDirectionType simple type (§4.8.55).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EightDirectionTransition">
   <attribute name="dir" type="ST_TransitionEightDirectionType" use="optional" default="l"/>
</complexType>
```

4.6.59 push (Push Slide Transition)

This element describes a push slide transition effect.

[Example: Consider we have a slide with a push slide transition. The <push> element should be used as follows:

```
<p:transition>
  <p:push dir="r"/>
</p:transition>
```

End example]

Parent Elements
transition (§4.4.1.46)

Attributes	Description
dir (Direction)	This attribute specifies the direction of the slide transition. The possible values for this attribute are defined by the ST_TransitionSideDirectionType simple type (§4.8.57).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SideDirectionTransition">
   <attribute name="dir" type="ST_TransitionSideDirectionType" use="optional" default="l"/>
</complexType>
```

4.6.60 random (Random Slide Transition)

This element describes a random slide transition effect.

[Example: Consider we have a slide with a random slide transition. The <random> element should be used as follows:

```
<p:transition>
    <p:random/>
</p:transition>
```

End example]

Parent Elements
transition (§4.4.1.46)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

4.6.61 randomBar (Random Bar Slide Transition)

This element describes a random bar slide transition effect.

[Example: Consider we have a slide with a random bar slide transition. The <randomBar> element should be used as follows:

```
<p:transition>
    <p:randomBar/>
</p:transition>
```

End example]

Parent Elements
transition (§4.4.1.46)

Attributes	Description
dir (Transition Direction)	This attribute specifies a horizontal or vertical transition. The possible values for this attribute are defined by the ST_Direction simple type (§4.8.5).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OrientationTransition">
   <attribute name="dir" type="ST_Direction" use="optional" default="horz"/>
</complexType>
```

4.6.62 rCtr (Rotation Center)

This element describes the center of the rotation used to rotate a motion path by X angle.

[*Example*: For example, suppose we have a simple animation with a checkerbox text entrance.

```
<p:animMotion origin="layout" path="M 0 0  L 0.25 0.33333  E"
pathEditMode="relative" rAng="0" ptsTypes="">
  <p:cBhvr>
    <p:cTn id="6" dur="2000" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="3"/>
    </p:tgtEl>
    <p:attrNameLst>
      <p:attrName>ppt_x</p:attrName>
      <p:attrName>ppt_y</p:attrName>
    </p:attrNameLst>
  </p:cBhvr>
  <p:rCtr x="457200" y="274638"/>
</p:animMotion>
```

end example]

Parent Elements
animMotion (§4.6.4)

Attributes	Description
x (X coordinate)	This attribute describes the X coordinate. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
y (Y coordinate)	This attribute describes the Y coordinate. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLPoint">
   <attribute name="x" type="a:ST_Percentage" use="required"/>
   <attribute name="y" type="a:ST_Percentage" use="required"/>
</complexType>
```

4.6.63 rgb (RGB)

The element specifies an incremental RGB value to add to the color property.

[Example: Consider a shape with a color emphasis animation. The <rgb> element should be used as follows:

```
<p:animClr clrSpc="rgb">
  <p:cBhvr>
    <p:cTn id="8" dur="500" fill="hold"/>
```

```
      <p:tgtEl>
        <p:spTgt spid="4"/>
      </p:tgtEl>
      <p:attrNameLst>
        <p:attrName>stroke.color</p:attrName>
      </p:attrNameLst>
    </p:cBhvr>
    <p:by>
      <p:rgb r="10" g="20" b="30"/>
    </p:by>
  </p:animClr>
```

End Example]

Parent Elements
by (§4.6.21)

Attributes	Description
b (Blue)	This attribute specifies a blue as a fixed percentage in 1000ths of a percent. The valid values range from [-100%, 100%]. The possible values for this attribute are defined by the ST_FixedPercentage simple type (§5.1.12.22).
g (Green)	This attribute specifies a green as a fixed percentage in 1000ths of a percent. The valid values range from [-100%, 100%]. The possible values for this attribute are defined by the ST_FixedPercentage simple type (§5.1.12.22).
r (Red)	This attribute specifies a red as a fixed percentage in 1000ths of a percent. The valid values range from [-100%, 100%]. The possible values for this attribute are defined by the ST_FixedPercentage simple type (§5.1.12.22).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLByRgbColorTransform">
   <attribute name="r" type="a:ST_FixedPercentage" use="required"/>
   <attribute name="g" type="a:ST_FixedPercentage" use="required"/>
   <attribute name="b" type="a:ST_FixedPercentage" use="required"/>
</complexType>
```

4.6.64 rtn (Runtime Node Trigger Choice)

This element specifies the child time node that triggers a time condition. References a child time node or all child nodes. Order is based on the child's end time.

[Example: Consider an animation which will end the synchronization of all parallel time nodes when all the child nodes have ended their animation. The <rtn> element should be used as follows:

```
<p:cTn>
  <p:stCondLst>...
  <p:endSync evt="end" delay="0">
    <p:rtn val="all"/>
  </p:endSync>
  <p:childTnLst>...
</p:cTn>
```

End Example]

Parent Elements
cond (§4.6.31); endSync (§4.6.39)

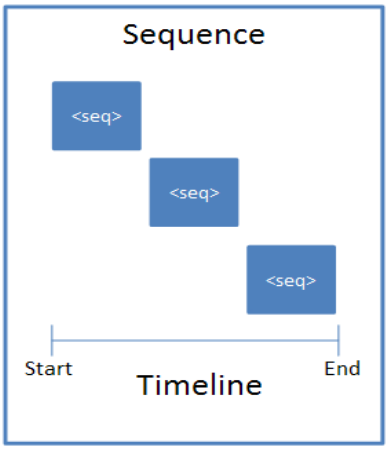
Attributes	Description
val (Value)	This attribute describes the value that will trigger the runtime node. The possible values for this attribute are defined by the ST_TLTriggerRuntimeNode simple type (§4.8.53).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLTriggerRuntimeNode">
   <attribute name="val" type="ST_TLTriggerRuntimeNode" use="required"/>
</complexType>
```

4.6.65 seq (Sequence Time Node)

This element describes the Sequence time node and it can only be activated when the one before it finishes. Conceptually it can be though of as follows:



[Example: For example, suppose we have a simple animation with a blind entrance.

```
<p:timing>
  <p:tnLst>
    <p:par>
      <p:cTn id="1" dur="indefinite" restart="never" nodeType="tmRoot">
        <p:childTnLst>
          <p:seq concurrent="1" nextAc="seek">
            ...
          </p:seq>
        </p:childTnLst>
      </p:cTn>
    </p:par>
  </p:tnLst>
</p:timing>
```

End Example]

Parent Elements
childTnLst (§4.6.25); subTnLst (§4.6.78); tnLst (§4.6.87)

Child Elements	Subclause
cTn (Common Time Node Properties)	§4.6.33
nextCondLst (Next Conditions List)	§4.6.51
prevCondLst (Previous Conditions List)	§4.6.55

Attributes	Description
concurrent (Concurrent)	This attribute specifies if concurrency is enabled or disabled. By default this attribute has a value of "disabled". When the value is set to "enabled", the previous element is left

Attributes	Description
	enabled when advancing to the next element in a sequence instead of being ended. This is only relevant for advancing via the next condition element being triggered. The only other way to advance to the next element would be to have the current element end, which implies it is no longer concurrent. The possible values for this attribute are defined by the XML Schema boolean datatype.
nextAc (Next Action)	This attribute specifies what to do when going forward in sequence. By default this attribute has a value of "none". When this is set to seek it will seek the element to a natural end time (not necessarily the actual end time). The natural end position is defined as the latest non-infinite end time of the children. If a child loops forever, the end of its first loop is used as its "end time" for the purposes of this calculation. Some container elements may have infinite durations due to an infinite-duration child element. The engine needs to recurse down through all infinite duration containers to calculate their natural duration in case a child might have non-infinite duration within it that needs to be taken into account. The possible values for this attribute are defined by the ST_TLNextActionType simple type (§4.8.38).
prevAc (Previous Action)	This attribute specifies what to do when going backwards in a sequence. By default it is set to "none" and nothing special is done. When the value is "skipTimed", the sequence will continue to go backwards until it reaches a sequence element that was defined to begin only on the next condition element. The possible values for this attribute are defined by the ST_TLPreviousActionType simple type (§4.8.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLTimeNodeSequence">
   <sequence>
      <element name="cTn" type="CT_TLCommonTimeNodeData" minOccurs="1" maxOccurs="1"/>
      <element name="prevCondLst" type="CT_TLTimeConditionList" minOccurs="0" maxOccurs="1"/>
      <element name="nextCondLst" type="CT_TLTimeConditionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="concurrent" type="xsd:boolean" use="optional"/>
   <attribute name="prevAc" type="ST_TLPreviousActionType" use="optional"/>
   <attribute name="nextAc" type="ST_TLNextActionType" use="optional"/>
</complexType>
```

4.6.66 set (Set Time Node Behavior)

This element allows the setting of a particular property value to a fixed value while the behavior is active and restores the value when the behavior is reset or turned off.

[Example: For example, suppose we want to set certain properties during an animation effect. The <set> element should be used as follows:

```
<p:childTnLst>
  <p:set>
    <p:cBhvr>
      <p:cTn id="6" dur="1" fill="hold">...
      <p:tgtEl>
        <p:spTgt spid="4"/>
      </p:tgtEl>
      <p:attrNameLst>
        <p:attrName>style.visibility</p:attrName>
      </p:attrNameLst>
    </p:cBhvr>
    <p:to>
      <p:strVal val="visible"/>
    </p:to>
  </p:set>
  <p:animEffect transition="in" filter="blinds(horizontal)">...
</p:childTnLst>
```

End Example]

Parent Elements
childTnLst (§4.6.25); subTnLst (§4.6.78); tnLst (§4.6.87)

Child Elements	Subclause
cBhvr (Common Behavior)	§4.6.22
to (To)	§4.6.89

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLSetBehavior">
   <sequence>
      <element name="cBhvr" type="CT_TLCommonBehaviorData" minOccurs="1" maxOccurs="1"/>
      <element name="to" type="CT_TLAnimVariant" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

4.6.67 sldTgt (Slide Target)

This element specifies the slide as the target element.

[Example: For example, suppose we have a simple animation with a blind entrance.

```
<p:seq concurrent="1" nextAc="seek">
```

```
<p:cTn id="2" dur="indefinite" nodeType="mainSeq">...
<p:prevCondLst>...
<p:nextCondLst>
  <p:cond evt="onNext" delay="0">
    <p:tgtEl>
      <p:sldTgt/>
    </p:tgtEl>
  </p:cond>
</p:nextCondLst>
</p:seq>
```

End Example]

Parent Elements
tgtEl (§4.6.81)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

4.6.68 snd (Sound)

This element specifies the audio information to play during a slide transition.

[Example: Consider a slide transition with an audio effect. The <snd> element should be used as follows:

```
<p:transition>
    <p:sndAc>
        <p:stSnd>
            <p:snd r:embed="rId2" r:link="rId3"/>
        </p:stSnd>
    </p:sndAc>
</p:transition>
```

End Example]

Parent Elements
stSnd (§4.6.76)

Attributes	Description
builtIn (Recognized Built-In Sound) Namespace: .../drawingml/200	Specifies whether or not this sound is a built-in sound. If this attribute is set to true then the generating application is alerted to check the name attribute specified for this sound in it's list of built-in sounds and can then surface a custom name or UI as needed. The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description
6/main	
embed (Embedded Audio File Relationship ID) Namespace: .../officeDocument/2006/relationships	Specifies the identification information for an embedded audio file. This attribute is used to specify the location of an object that resides locally within the file. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
name (Sound Name) Namespace: .../drawingml/2006/main	Specifies the original name or given short name for the corresponding sound. This is used to distinguish this sound from others by providing a human readable name for the attached sound should the user need to identify the sound among others within the UI. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EmbeddedWAVAudioFile">
   <attribute ref="r:embed" use="required"/>
   <attribute name="name" type="xsd:string" use="optional" default=""/>
   <attribute name="builtIn" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.6.69 sndAc (Sound Action)

This element describes the Sound Action for slide transition.

[Example: Consider a slide transition with a sound effect. The <sndAc> element should be used as follows:

```
<p:transition>
    <p:sndAc>
        <p:stSnd>
            <p:snd r:embed="rId2" r:link="rId3"/>
        </p:stSnd>
    </p:sndAc>
</p:transition>
```

End Example]

Parent Elements
transition (§4.4.1.46)

Child Elements	Subclause
endSnd (Stop Sound Action)	§4.6.38

Child Elements	Subclause
stSnd (Start Sound Action)	§4.6.76

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TransitionSoundAction">
   <choice minOccurs="1" maxOccurs="1">
      <element name="stSnd" type="CT_TransitionStartSoundAction"/>
      <element name="endSnd" type="CT_Empty"/>
   </choice>
</complexType>
```

4.6.70 sndTgt (Sound Target)

This element describes the sound information for a target object.

[Example: Consider a shape with a sound effect animation. The <sndTgt> element should be used as follows:

```
<p:subTnLst>
  <p:audio>
    <p:cMediaNode>
      <p:cTn display="0" masterRel="sameClick">...
      <p:tgtEl>
        <p:sndTgt r:embed="rId2" r:link="rId3"/>
      </p:tgtEl>
    </p:cMediaNode>
  </p:audio>
</p:subTnLst>
```

End Example]

Parent Elements
tgtEl (§4.6.81)

Attributes	Description
builtIn (Recognized Built-In Sound) Namespace: .../drawingml/2006/main	Specifies whether or not this sound is a built-in sound. If this attribute is set to true then the generating application is alerted to check the name attribute specified for this sound in it's list of built-in sounds and can then surface a custom name or UI as needed. The possible values for this attribute are defined by the XML Schema boolean datatype.
embed (Embedded Audio File Relationship ID) Namespace:	Specifies the identification information for an embedded audio file. This attribute is used to specify the location of an object that resides locally within the file. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

Attributes	Description
.../officeDocument/2006/relationships	
name (Sound Name) Namespace: .../drawingml/2006/main	Specifies the original name or given short name for the corresponding sound. This is used to distinguish this sound from others by providing a human readable name for the attached sound should the user need to identify the sound among others within the UI. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EmbeddedWAVAudioFile">
   <attribute ref="r:embed" use="required"/>
   <attribute name="name" type="xsd:string" use="optional" default=""/>
   <attribute name="builtIn" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.6.71 split (Split Slide Transition)

This element describes a split slide transition effect.

[Example: Consider we have a slide with a split slide transition. The <split> element should be used as follows:

```
<p:transition>
    <p:split dir="in"/>
</p:transition>
```

End example]

Parent Elements
transition (§4.4.1.46)

Attributes	Description
dir (Direction)	This attribute specifies the direction of a "split" slide transition. The possible values for this attribute are defined by the ST_TransitionInOutDirectionType simple type (§4.8.56).
orient (Orientation)	This attribute specifies the orientation of a "split" slide transition. The possible values for this attribute are defined by the ST_Direction simple type (§4.8.5).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SplitTransition">
   <attribute name="orient" type="ST_Direction" use="optional" default="horz"/>
   <attribute name="dir" type="ST_TransitionInOutDirectionType" use="optional" default="out"/>
</complexType>
```

4.6.72 spTgt (Shape Target)

The element specifies the shape in which to apply a certain animation to.

[Example: Consider a shape whose id is 3 in which we want to apply a fade animation effect. The <spTgt> should be used as follows:

```
<p:animEffect transition="in" filter="fade">
  <p:cBhvr>
    <p:cTn id="7" dur="2000"/>
    <p:tgtEl>
      <p:spTgt spid="3"/>
    </p:tgtEl>
  </p:cBhvr>
</p:animEffect>
```

End Example]

Parent Elements
tgtEl (§4.6.81)

Child Elements	Subclause
bg (Background)	§4.6.10
graphicEl (Graphic Element)	§4.6.45
oleChartEl (Embedded Chart Element)	§4.6.52
subSp (Subshape)	§4.6.77
txEl (Text Element)	§4.6.91

Attributes	Description
spid (Shape ID)	This attribute specifies the shape identifier. The possible values for this attribute are defined by the ST_ShapeID simple type (§5.1.12.55).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLShapeTargetElement">
   <choice minOccurs="0" maxOccurs="1">
      <element name="bg" type="CT_Empty"/>
      <element name="subSp" type="CT_TLSubShapeId"/>
      <element name="oleChartEl" type="CT_TLOleChartTargetElement"/>
      <element name="txEl" type="CT_TLTextTargetElement"/>
      <element name="graphicEl" type="a:CT_AnimationElementChoice"/>
   </choice>
   <attribute name="spid" type="a:ST_ShapeID" use="required"/>
</complexType>
```

4.6.73 stCondLst (Start Conditions List)

This element contains a list conditions that must be met for a time node to be activated.

[Example: For example, suppose we have a shape with an entrance appearance after 5 seconds. The <stCondLst>element should be used as follows:

```
<p:par>
  <p:cTn id="5" nodeType="clickEffect">
    <p:stCondLst>
      <p:cond delay="5000"/>
    </p:stCondLst>
    <p:childTnLst>...
  </p:cTn>
</p:par>
```

End Example]

Parent Elements
cTn (§4.6.33)

Child Elements	Subclause
cond (Condition)	§4.6.31

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLTimeConditionList">
   <sequence>
      <element name="cond" type="CT_TLTimeCondition" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

4.6.74 strips (Strips Slide Transition)

This element describes a strips slide transition effect.

[Example: Consider we have a slide with a strips slide transition. The <strips> element should be used as follows:

```
<p:transition>
    <p:strips/>
</p:transition>
```

End example]

Parent Elements
transition (§4.4.1.46)

Attributes	Description
dir (Direction)	This attribute specifies if the direction of the transition. The possible values for this attribute are defined by the ST_TransitionCornerDirectionType simple type (§4.8.54).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CornerDirectionTransition">
   <attribute name="dir" type="ST_TransitionCornerDirectionType" use="optional" default="lu"/>
</complexType>
```

4.6.75 strVal (String Value)

This element describes a string variant.

Parent Elements
progress (§4.6.57); to (§4.6.89); val (§4.6.92)

Attributes	Description
val (Value)	This attribute specifies the value of this element as a string The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLAnimVariantStringVal">
   <attribute name="val" type="xsd:string" use="required"/>
</complexType>
```

4.6.76 stSnd (Start Sound Action)

This element describes the sound that will start playing during a slide transition.

[Example: Consider a slide transition that starts with a sound effect. The <stSnd> element should be used as follows:

```
<p:transition>
    <p:sndAc>
        <p:stSnd>
            <p:snd r:embed="rId2" r:link="rId3"/>
        </p:stSnd>
    </p:sndAc>
</p:transition>
```

End Example]

Parent Elements
sndAc (§4.6.69)

Child Elements	Subclause
snd (Sound)	§4.6.68

Attributes	Description
loop (Loop Sound)	This attribute specifies if the sound will loop until the next sound event occurs in slideshow. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TransitionStartSoundAction">
   <sequence>
      <element minOccurs="1" maxOccurs="1" name="snd" type="a:CT_EmbeddedWAVAudioFile"/>
   </sequence>
   <attribute name="loop" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.6.77 subSp (Subshape)

This element specifies the subshape of a legacy graphical object to animate.

[Example: Consider adding animation to a legacy diagram. The <subSp> element should be used as follows:

```
<p:animEffect transition="in" filter="blinds(horizontal)">
  <p:cBhvr>
    <p:cTn id="7" dur="500"/>
    <p:tgtEl>
      <p:spTgt spid="2053">
```

```
            <p:subSp spid="_x0000_s70664"/>
          </p:spTgt>
        </p:tgtEl>
      </p:cBhvr>
    </p:animEffect>
```

End Example]

Parent Elements
spTgt (§4.6.72)

Attributes	Description
spid (Shape ID)	This attribute specifies the shape identifier. The possible values for this attribute are defined by the ST_ShapeID simple type (§5.1.12.55).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLSubShapeId">
   <attribute name="spid" type="a:ST_ShapeID" use="required"/>
</complexType>
```

4.6.78 subTnLst (Sub-TimeNodes List)

This element describes time nodes that have a start time which is not based on the containing timenode. It is instead based on their master relationship (masterRel). At runtime, they are inserted dynamically into the timing tree as child timenodes for playback, based on the logic defined by the master relationship. These elements are used for animations such as "dim after" and "play sound effects"

[Example: Consider an animation with a "Fly In" effect on paragraphs so that each paragraph flies in on a separate click. Then the "Dim After" effect for paragraph 1 doe not happen until paragraph 2 flies in. The <subTnLst> element should be used as follows:

```
<p:par>
  <p:cTn id="5" grpId="0" nodeType="clickEffect">
    <p:stCondLst>...
    <p:childTnLst>...
    <p:subTnLst>
      <p:set>
        <p:cBhvr override="childStyle">
          <p:cTn fill="hold" masterRel="nextClick" afterEffect="1"/>
          <p:tgtEl>...
          <p:attrNameLst>...
        </p:cBhvr>
```

```
          <p:to>...
        </p:set>
      </p:subTnLst>
    </p:cTn>
  </p:par>
```

End Example]

Parent Elements
cTn (§4.6.33)

Child Elements	Subclause
anim (Animate)	§4.6.1
animClr (Animate Color Behavior)	§4.6.2
animEffect (Animate Effect)	§4.6.3
animMotion (Animate Motion)	§4.6.4
animRot (Animate Rotation)	§4.6.5
animScale (Animate Scale)	§4.6.6
audio (Audio)	§4.6.9
cmd (Command)	§4.6.28
excl (Exclusive)	§4.6.40
par (Parallel Time Node)	§4.6.53
seq (Sequence Time Node)	§4.6.65
set (Set Time Node Behavior)	§4.6.66
video (Video)	§4.6.93

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TimeNodeList">
   <choice minOccurs="1" maxOccurs="unbounded">
      <element name="par" type="CT_TLTimeNodeParallel"/>
      <element name="seq" type="CT_TLTimeNodeSequence"/>
      <element name="excl" type="CT_TLTimeNodeExclusive"/>
      <element name="anim" type="CT_TLAnimateBehavior"/>
      <element name="animClr" type="CT_TLAnimateColorBehavior"/>
      <element name="animEffect" type="CT_TLAnimateEffectBehavior"/>
      <element name="animMotion" type="CT_TLAnimateMotionBehavior"/>
      <element name="animRot" type="CT_TLAnimateRotationBehavior"/>
      <element name="animScale" type="CT_TLAnimateScaleBehavior"/>
      <element name="cmd" type="CT_TLCommandBehavior"/>
      <element name="set" type="CT_TLSetBehavior"/>
      <element name="audio" type="CT_TLMediaNodeAudio"/>
      <element name="video" type="CT_TLMediaNodeVideo"/>
   </choice>
</complexType>
```

4.6.79 tav (Time Animate Value)

This element defines a "keypoint" in animation interpolation.

[Example: Consider a shape with a "fly-in" animation. The <tav> element should be used as follows:

```
<p:anim calcmode="lin" valueType="num">
  <p:cBhvr additive="base">...
  <p:tavLst>
    <p:tav tm="0">
      <p:val>
        <p:strVal val="1+#ppt_h/2"/>
      </p:val>
    </p:tav>
    <p:tav tm="100000">
      <p:val>
        <p:strVal val="#ppt_y"/>
      </p:val>
    </p:tav>
  </p:tavLst>
</p:anim>
```

End Example]

Parent Elements
tavLst (§4.6.80)

Child Elements	Subclause
val (Value)	§4.6.92

Attributes	Description
fmla (Formula)	This attribute allows for a specific formula to be used during the animation. This is specified in a semicolon-separated list of formulas. The value generated by interpolating the values list will be fed into the formula as an input. The resulting value is used to set the property being animated. There should be n-1 number of formulas for n values in the values list. The formula is applied starting at the corresponding time in the keyTimes list and up to the next time in the list. Formulas can only support a calcMode (Calculation Mode) of linear or discrete. If another calcMode is specified or no calcMode is specified then a calcMode of linear will be assumed. Formulas must be written using the elements defined below. When a formula is used in an attribute that takes more than one parameter (such as for animateScale or animateMotion), a space will be taken as a delimiter between parameter values. To work around this, either don't put spaces in the formula or include parentheses around the formula. Formulas within values, from, to, by attributes can be made up of these: • Standard arithmetic operators: ‘+’, ‘-‘, ‘*’, ‘/’, ‘^’, ‘%’ (mod) • Constants: ‘pi’ ‘e’ • Conditional operators: ‘abs’, ‘min’, ‘max’, ‘?’ (if) • Comparison operators: '==', '>=', '<=', '!=', '!' • Trigonometric operators: ‘sin()’, ‘cos()’, ‘tan()’, ‘asin()’, ‘acos()’, ‘atan()’ • Natural logarithm ‘ln()’ • Property references (host supported properties) [Note: The above formula elements are different than those used in javascript expressions. For example "cos()" used in formulas would be "Math.cos()" in javascript. end note] The possible values for this attribute are defined by the XML Schema string datatype.
tm (Time)	This attribute specifies the time at which the attribute being animated will take on the value. The possible values for this attribute are defined by the ST_TLTimeAnimateValueTime simple type (§4.8.43).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLTimeAnimateValue">
   <sequence>
      <element name="val" type="CT_TLAnimVariant" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="tm" type="ST_TLTimeAnimateValueTime" use="optional" default="indefinite"/>
   <attribute name="fmla" type="xsd:string" use="optional" default=""/>
</complexType>
```

4.6.80 tavLst (Time Animated Value List)

This element specifies a list of time animated value elements.

[Example: Consider a shape with a "fly-in" animation. The <tav> element should be used as follows:

```
<p:anim calcmode="lin" valueType="num">
  <p:cBhvr additive="base">...
  <p:tavLst>
    <p:tav tm="0">
      <p:val>
        <p:strVal val="1+#ppt_h/2"/>
      </p:val>
    </p:tav>
    <p:tav tm="100000">
      <p:val>
        <p:strVal val="#ppt_y"/>
      </p:val>
    </p:tav>
  </p:tavLst>
</p:anim>
```

End Example]

Parent Elements
anim (§4.6.1)

Child Elements	Subclause
tav (Time Animate Value)	§4.6.79

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLTimeAnimateValueList">
   <sequence>
      <element name="tav" type="CT_TLTimeAnimateValue" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

4.6.81 tgtEl (Target Element)

This element specifies the target children elements which will have the animation effects applied to.

[Example: Consider a shape with ID 3 with a fade effect animation applied to it. The <tgtEl> element should be used as follows:

```
<p:animEffect transition="in" filter="fade">
  <p:cBhvr>
    <p:cTn id="7" dur="2000"/>
    <p:tgtEl>
      <p:spTgt spid="3"/>
    </p:tgtEl>
  </p:cBhvr>
</p:animEffect>
```

End Example]

Parent Elements
cBhvr (§4.6.22); cMediaNode (§4.6.29); cond (§4.6.31); endSync (§4.6.39)

Child Elements	Subclause
inkTgt (Ink Target)	§4.6.47
sldTgt (Slide Target)	§4.6.67
sndTgt (Sound Target)	§4.6.70
spTgt (Shape Target)	§4.6.72

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLTimeTargetElement">
   <choice minOccurs="1" maxOccurs="1">
      <element name="sldTgt" type="CT_Empty"/>
      <element name="sndTgt" type="a:CT_EmbeddedWAVAudioFile"/>
      <element name="spTgt" type="CT_TLShapeTargetElement"/>
      <element name="inkTgt" type="CT_TLSubShapeId"/>
   </choice>
</complexType>
```

4.6.82 tmAbs (Time Absolute)

This element describes the duration of the iteration interval in absolute time.

[Example: Consider a text animation where the words appear letter by letter every 10 seconds. The <tmAbs> element should be used as follows:

```
<p:par>
```

```
    <p:cTn id="5" >
      <p:stCondLst>...
      <p:iterate type="lt">
        <p:tmAbs val="10000"/>
      </p:iterate>
      <p:childTnLst>...
    </p:cTn>
  </p:par>
```

End Example]

Parent Elements
iterate (§4.6.49)

Attributes	Description
val (Time)	This attribute describes an amount of time, in milliseconds. The possible values for this attribute are defined by the ST_TLTime simple type (§4.8.42).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLIterateIntervalTime">
   <attribute name="val" type="ST_TLTime" use="required"/>
</complexType>
```

4.6.83 tmPct (Time Percentage)

This element describes the duration of the iteration interval in a percentage of time.

[Example: Consider a text animation where the words appear letter by letter every 10th of the animation duration. The <tmPct> element should be used as follows:

```
  <p:par>
    <p:cTn id="5" >
      <p:stCondLst>...
      <p:iterate type="lt">
        <p:tmPct val="10000"/>
      </p:iterate>
      <p:childTnLst>...
    </p:cTn>
  </p:par>
```

End Example]

Parent Elements
iterate (§4.6.49)

Attributes	Description
val (Value)	This attribute specifies the time expressed as a percentage. The possible values for this attribute are defined by the ST_PositivePercentage simple type (§5.1.12.46).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLIterateIntervalPercentage">
   <attribute name="val" type="a:ST_PositivePercentage" use="required"/>
</complexType>
```

4.6.84 tmpl (Template Effects)

This element specifies the "template" effects that are used by the build element. Template effects are used in text builds on the master slide. They define the rules of what effect should be applied to the 1st level paragraph, 2nd level paragraph, etc.

[Example: Consider a template with a fade in effect applied to it. The <tmpl> element should be used as follows:

```
<p:timing>
  <p:tnLst>...
  <p:bldLst>
    <p:bldP spid="3" grpId="0" build="p">
      <p:tmplLst>
        <p:tmpl lvl="1">
        </p:tmpl>
      </p:tmplLst>
    </p:bldP>
  </p:bldLst>
</p:timing>
```

End Example]

Parent Elements
tmplLst (§4.6.85)

Child Elements	Subclause
tnLst (Time Node List)	§4.6.87

Attributes	Description
lvl (Level)	This attribute describes the paragraph indent level to which this template effect applies. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLTemplate">
   <sequence>
      <element name="tnLst" type="CT_TimeNodeList" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="lvl" type="xsd:unsignedInt" use="optional" default="0"/>
</complexType>
```

4.6.85 tmplLst (Template effects)

This element describes a list of template effects that describe what kind of effects should be applied to a paragraph level properties.

[Example: Consider a template with a fade in effect applied to it. The <tmpl> element should be used as follows:

```
<p:timing>
  <p:tnLst>...
  <p:bldLst>
    <p:bldP spid="3" grpId="0" build="p">
      <p:tmplLst>
        <p:tmpl lvl="1">
        </p:tmpl>
      </p:tmplLst>
    </p:bldP>
  </p:bldLst>
</p:timing>
```

End Example]

Parent Elements
bldP (§4.6.16)

Child Elements	Subclause
tmpl (Template Effects)	§4.6.84

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLTemplateList">
   <sequence>
      <element name="tmpl" type="CT_TLTemplate" minOccurs="0" maxOccurs="9"/>
   </sequence>
</complexType>
```

4.6.86 tn (Time Node)

This element describes the time node trigger choice.

[Example: Consider a time node with an event condition. The <tn> element should be used as follows:

```
<p:par>
  <p:cTn id="5">
    <p:stCondLst>
      <p:cond delay="0"/>
    </p:stCondLst>
    <p:endCondLst>
      <p:cond evt="begin" delay="0">
        <p:tn val="5"/>
      </p:cond>
    </p:endCondLst>
    <p:childTnLst>...
  </p:cTn>
</p:par>
```

End Example]

Parent Elements
cond (§4.6.31); endSync (§4.6.39)

Attributes	Description
val (Value)	This attribute specifies a time node identifier. The possible values for this attribute are defined by the ST_TLTimeNodeID simple type (§4.8.46).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLTriggerTimeNodeID">
   <attribute name="val" type="ST_TLTimeNodeID" use="required"/>
</complexType>
```

4.6.87 tnLst (Time Node List)

This element specifies a list of time node elements used in an animation sequence.

[Example: Consider a simple animation sequence. The <tnLst> element should be used as follows:

```
<p:timing>
  <p:tnLst>
    <p:par>...
  </p:tnLst>
</p:timing>
```

End Example]

Parent Elements
timing (§4.4.1.44); tmpl (§4.6.84)

Child Elements	Subclause
anim (Animate)	§4.6.1
animClr (Animate Color Behavior)	§4.6.2
animEffect (Animate Effect)	§4.6.3
animMotion (Animate Motion)	§4.6.4
animRot (Animate Rotation)	§4.6.5
animScale (Animate Scale)	§4.6.6
audio (Audio)	§4.6.9
cmd (Command)	§4.6.28
excl (Exclusive)	§4.6.40
par (Parallel Time Node)	§4.6.53
seq (Sequence Time Node)	§4.6.65
set (Set Time Node Behavior)	§4.6.66
video (Video)	§4.6.93

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TimeNodeList">
   <choice minOccurs="1" maxOccurs="unbounded">
      <element name="par" type="CT_TLTimeNodeParallel"/>
      <element name="seq" type="CT_TLTimeNodeSequence"/>
      <element name="excl" type="CT_TLTimeNodeExclusive"/>
      <element name="anim" type="CT_TLAnimateBehavior"/>
      <element name="animClr" type="CT_TLAnimateColorBehavior"/>
      <element name="animEffect" type="CT_TLAnimateEffectBehavior"/>
      <element name="animMotion" type="CT_TLAnimateMotionBehavior"/>
      <element name="animRot" type="CT_TLAnimateRotationBehavior"/>
      <element name="animScale" type="CT_TLAnimateScaleBehavior"/>
      <element name="cmd" type="CT_TLCommandBehavior"/>
      <element name="set" type="CT_TLSetBehavior"/>
      <element name="audio" type="CT_TLMediaNodeAudio"/>
      <element name="video" type="CT_TLMediaNodeVideo"/>
   </choice>
</complexType>
```

4.6.88 to (To)

This element specifies the target location for an animation motion or animation scale effect

[Example: Consider an animation with a "light speed" entrance effect.

```
<p:animScale>
  <p:cBhvr>
    <p:cTn id="9" dur="200" decel="100000" autoRev="1" fill="hold">
      <p:stCondLst>
        <p:cond delay="600"/>
      </p:stCondLst>
    </p:cTn>
    <p:tgtEl>
      <p:spTgt spid="4"/>
    </p:tgtEl>
  </p:cBhvr>
  <p:from x="100000" y="100000"/>
  <p:to x="80000" y="100000"/>
</p:animScale>
```

End Example]

Parent Elements
animMotion (§4.6.4); animScale (§4.6.6)

Attributes	Description

Attributes	Description
x (X coordinate)	This attribute describes the X coordinate. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
y (Y coordinate)	This attribute describes the Y coordinate. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLPoint">
   <attribute name="x" type="a:ST_Percentage" use="required"/>
   <attribute name="y" type="a:ST_Percentage" use="required"/>
</complexType>
```

4.6.89 to (To)

The element specifies the certain attribute of a time node after an animation effect.

[Example: Consider an animation effect that leaves a string value visible afterwards. The <to> element should be used as follows:

```
<p:childTnLst>
  <p:set>
    <p:cBhvr>...
    <p:to>
      <p:strVal val="visible"/>
    </p:to>
  </p:set>
  <p:anim calcmode="lin" valueType="num">...
  <p:anim calcmode="lin" valueType="num">...
</p:childTnLst>
```

End Example]

Parent Elements
set (§4.6.66)

Child Elements	Subclause
boolVal (Boolean Variant)	§4.6.19
clrVal (Color Value)	§4.6.27

Child Elements	Subclause
fltVal (Float Value)	§4.6.42
intVal (Integer)	§4.6.48
strVal (String Value)	§4.6.75

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLAnimVariant">
   <choice minOccurs="1" maxOccurs="1">
      <element name="boolVal" type="CT_TLAnimVariantBooleanVal"/>
      <element name="intVal" type="CT_TLAnimVariantIntegerVal"/>
      <element name="fltVal" type="CT_TLAnimVariantFloatVal"/>
      <element name="strVal" type="CT_TLAnimVariantStringVal"/>
      <element name="clrVal" type="a:CT_Color"/>
   </choice>
</complexType>
```

4.6.90 to (To)

This element specifies the resulting color for the animation color change.

[Example: Consider emphasize a shape by changing its fill color from blue to red. The <to> element should be used as follows:

```
<p:childTnLst>
  <p:animClr clrSpc="rgsb">
    <p:cBhvr>...
    <p:to>
      <a:schemeClr val="accent2"/>
    </p:to>
  </p:animClr>
</p:childTnLst>
```

End example]

Parent Elements
animClr (§4.6.2)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32

Child Elements	Subclause
sysClr (System Color)	§5.1.2.2.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

4.6.91 txEl (Text Element)

This element specifies a text element to animate.

[Example: Consider a shape containing text to be animated. The <txEl> should be used as follows:

```
<p:tgtEl>
  <p:spTgt spid="5">
    <p:txEl>
      <p:pRg st="1" end="1"/>
    </p:txEl>
  </p:spTgt>
</p:tgtEl>
```

End Example]

Parent Elements
spTgt (§4.6.72)

Child Elements	Subclause
charRg (Character Range)	§4.6.23
pRg (Paragraph Text Range)	§4.6.56

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLTextTargetElement">
   <choice minOccurs="0" maxOccurs="1">
      <element name="charRg" type="CT_IndexRange"/>
      <element name="pRg" type="CT_IndexRange"/>
   </choice>
</complexType>
```

4.6.92 val (Value)

The element specifies a value for a time animate.

[Example: Consider a shape with a fade in animation effect. The <val> element should be used as follows:

```
<p:anim calcmode="lin" valueType="num">
  <p:cBhvr additive="base">...
  <p:tavLst>
    <p:tav tm="0">
      <p:val>
        <p:strVal val="0-#ppt_w/2"/>
      </p:val>
    </p:tav>
    <p:tav tm="100000">
      <p:val>
        <p:strVal val="#ppt_x"/>
      </p:val>
    </p:tav>
  </p:tavLst>
</p:anim>
```

End Example]

Parent Elements
tav (§4.6.79)

Child Elements	Subclause
boolVal (Boolean Variant)	§4.6.19
clrVal (Color Value)	§4.6.27
fltVal (Float Value)	§4.6.42
intVal (Integer)	§4.6.48
strVal (String Value)	§4.6.75

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLAnimVariant">
   <choice minOccurs="1" maxOccurs="1">
      <element name="boolVal" type="CT_TLAnimVariantBooleanVal"/>
      <element name="intVal" type="CT_TLAnimVariantIntegerVal"/>
      <element name="fltVal" type="CT_TLAnimVariantFloatVal"/>
      <element name="strVal" type="CT_TLAnimVariantStringVal"/>
      <element name="clrVal" type="a:CT_Color"/>
   </choice>
</complexType>
```

4.6.93 video (Video)

This element specifies video information in an animation sequence.

[Example: Consider a slide with an animated video content. The <video> element should be used as follows:

```
<p:childTnLst>
  <p:seq concurrent="1" nextAc="seek">...
  <p:video>
    <p:cMediaNode>...
  </p:video>
</p:childTnLst>
End Example]
```

Parent Elements
childTnLst (§4.6.25); subTnLst (§4.6.78); tnLst (§4.6.87)

Child Elements	Subclause
cMediaNode (Common Media Node Properties)	§4.6.29

Attributes	Description
fullScrn (Full Screen)	This attribute specifies if the video will be displayed in full-screen. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TLMediaNodeVideo">
   <sequence>
      <element name="cMediaNode" type="CT_TLCommonMediaNodeData" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="fullScrn" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

4.6.94 wedge (Wedge Slide Transition)

This element describes a wedge slide transition effect.

[Example: Consider we have a slide with a wedge slide transition. The <wedge> element should be used as follows:

```
<p:transition>
    <p:wedge/>
</p:transition>
```

End example]

Parent Elements

Parent Elements
transition (§4.4.1.46)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

4.6.95 wheel (Wheel Slide Transition)

This element describes a wheel slide transition effect.

[Example: Consider we have a slide with a wheel slide transition. The <wheel> element should be used as follows:

```
<p:transition>
    <p:wheel/>
</p:transition>
```

End example]

Parent Elements
transition (§4.4.1.46)

Attributes	Description
spokes (Spokes)	This attributes specifies the number of spokes ("pie pieces") in the wheel The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_WheelTransition">
   <attribute name="spokes" type="xsd:unsignedInt" use="optional" default="4"/>
</complexType>
```

4.6.96 wipe (Wipe Slide Transition)

This element describes a wipe slide transition effect.

[Example: Consider we have a slide with a wipe slide transition. The <wipe> element should be used as follows:

```
<p:transition>
    <p:wipe/>
</p:transition>
```

End example]

Parent Elements
transition (§4.4.1.46)

Attributes	Description
dir (Direction)	This attribute specifies the direction of the slide transition. The possible values for this attribute are defined by the ST_TransitionSideDirectionType simple type (§4.8.57).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SideDirectionTransition">
   <attribute name="dir" type="ST_TransitionSideDirectionType" use="optional" default="l"/>
</complexType>
```

4.6.97 zoom (Zoom Slide Transition)

This element describes a zoom slide transition effect.

[Example: Consider we have a slide with a zoom slide transition. The <zoom> element should be used as follows:

```
<p:transition>
    <p:zoom/>
</p:transition>
```

End example]

Parent Elements
transition (§4.4.1.46)

Attributes	Description
dir (Direction)	This attribute specifies the direction of an "in/out" slide transition. The possible values for this attribute are defined by the ST_TransitionInOutDirectionType simple type (§4.8.56).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_InOutTransition">
   <attribute name="dir" type="ST_TransitionInOutDirectionType" use="optional" default="out"/>
</complexType>
```

4.7 Slide Synchronization Data

It is often desired to allow a user to pull slides from a library of pre-existing content. In order to facilitate synchronization of this content with a server, the sldSyncPr element exists to store modification/insertion time properties to enable this synchronization. The sldSyncPr element lives in the slideUpdateInfo directory of the PowerPoint file container.

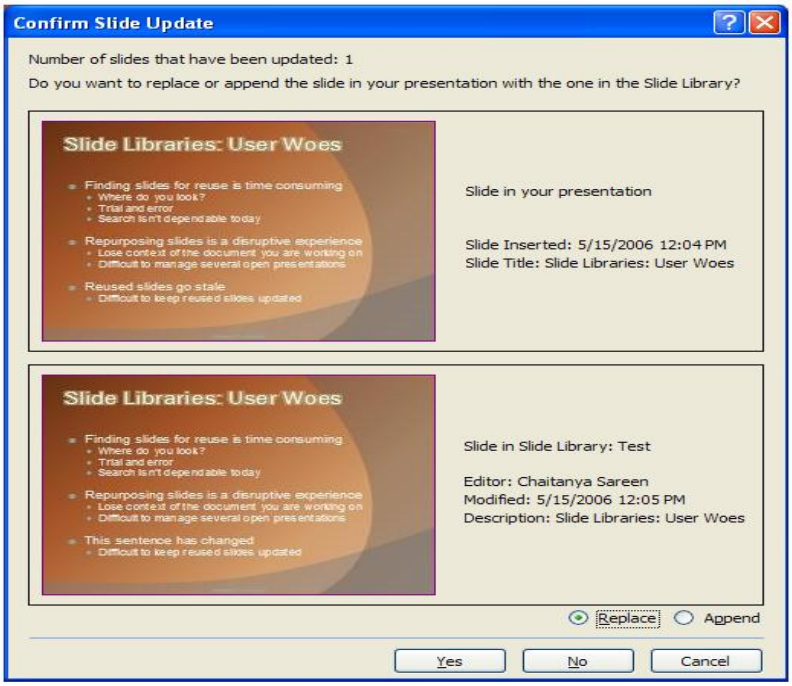
4.7.1 sldSyncPr (Slide Synchronization Properties)

This element specifies the properties associated with the slide synchronization specified for the current slide.

Parent Elements
Root element of PresentationML Slide Synchronization Data part

Child Elements	Subclause
extLst (Extension List)	§4.2.5

Attributes	Description
clientInsertedTime (Client Slide Insertion date/time)	The date and time that the server-based slide was inserted into the client presentation. This field is used purely for informative purposes so that the user may know when the slide was inserted. The date/time is stored in ISO 8061 format. [Example: Notice the use of clientInsertedTime in this UI to provide users information on when the slide was inserted from the slide library.

Attributes	Description
	 End Example] The possible values for this attribute are defined by the XML Schema dateTime datatype.
serverSldId (Server's Slide File ID)	Identifier of the slide file on the server. This ID is unrelated to the Slide's ID, and is a unique identifier for the slide in the slide library. The possible values for this attribute are defined by the XML Schema string datatype.
serverSldModifiedTime (Server's Slide File's modification date/time)	The last modification date and time for the slide file that is located on the server. This date, when compared with the last modification date and time on the server, determines whether an update of this slide is necessary. The date and time are stored in ISO 8061 format. The possible values for this attribute are defined by the XML Schema dateTime datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SlideSyncProperties">
   <sequence>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="serverSldId" type="xsd:string" use="required"/>
   <attribute name="serverSldModifiedTime" type="xsd:dateTime" use="required"/>
   <attribute name="clientInsertedTime" type="xsd:dateTime" use="required"/>
</complexType>
```

4.8 Simple Types

This is the complete list of simple types in the http://schemas.openxmlformats.org/presentationml/2006/main namespace.

4.8.1 ST_AlgClass (Cryptographic Algorithm Classes)

This simple type specifies the possible classes of cryptographic algorithm used by protection. [*Note*: The initial version of this Office Open XML Standard only supports a single version - hash - but future versions may expand this as necessary. *end note*]

[*Example*: Consider a PresentationML document with the following information stored in its protection element:

```
<p:… p:cryptAlgorithmClass="hash"
  p:cryptAlgorithmType="typeAny"
  p:cryptAlgorithmSid="1"
  p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />
```

The cryptAlgorithmClass attribute value of hash specifies that the algorithm used for the password is a hashing algorithm. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
hash (Hash Algorithm Class)	A hash class algorithm is used.
invalid (Invalid Algorithm Class)	An algorithm with an invalid class is used.

Referenced By
modifyVerifier@cryptAlgorithmClass (§4.3.1.17)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_AlgClass">
   <restriction base="xsd:string">
      <enumeration value="hash"/>
      <enumeration value="invalid"/>
   </restriction>
</simpleType>
```

4.8.2 ST_AlgType (Cryptographic Algorithm Type)

This simple type specifies the possible values for the type of cryptographic algorithm used by protection. [*Note*: The initial version of this Office Open XML Standard only supports a single type - typeAny - but future versions may expand this as necessary. *end note*]

[*Example*: Consider a PresentationML document with the following information stored in its protection element:

```
<p:… p:cryptAlgorithmClass="hash"
  p:cryptAlgorithmType="typeAny"
  p:cryptAlgorithmSid="1"
  p:hashData="9oN7nWkCAyEZib1RomSJTjmPpCY=" />
```

The cryptAlgorithmType attribute value of typeAny specifies that any type of algorithm may have been used for the password. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
invalid (Invalid Algorithm Type)	An invalid algorithm type is used.
typeAny (Any Algorithm Type)	Any algorithm type is used.

Referenced By
modifyVerifier@cryptAlgorithmType (§4.3.1.17)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_AlgType">
   <restriction base="xsd:string">
      <enumeration value="typeAny"/>
      <enumeration value="invalid"/>
   </restriction>
</simpleType>
```

4.8.3 ST_BookmarkIdSeed (Bookmark ID Seed)

This type specifies constraints for value of the Bookmark ID seed.

This simple type's contents are a restriction of the XML Schema unsignedInt datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 1.
- This simple type has a maximum value of less than 2147483648.

Referenced By
presentation@bookmarkIdSeed (§4.3.1.24)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_BookmarkIdSeed">
   <restriction base="xsd:unsignedInt">
      <minInclusive value="1"/>
      <maxExclusive value="2147483648"/>
   </restriction>
</simpleType>
```

4.8.4 ST_CryptProv (Cryptographic Provider Type)

This simple type specifies the possible types of cryptographic providers which may be used.

[*Example*: Consider a PresentationML document with the following information stored in one of its protection elements:

```
<p:modifyVerifier p:cryptProviderType="rsaAES" ... />
```

The cryptProviderType attribute value of `rsaAES` specifies that the cryptographic provider type shall be an Advanced Encryption Standard provider. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
invalid (Invalid Encryption Scheme)	Invalid Encryption scheme provided.
rsaAES (RSA AES Encryption Scheme)	Specifies that the provider shall support the Advanced Encryption Algorithm standard.
rsaFull (RSA Full Encryption Scheme)	Specifies that any suitable provider shall be used.

Referenced By
modifyVerifier@cryptProviderType (§4.3.1.17)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_CryptProv">
   <restriction base="xsd:string">
      <enumeration value="rsaAES"/>
      <enumeration value="rsaFull"/>
      <enumeration value="invalid"/>
   </restriction>
</simpleType>
```

4.8.5 ST_Direction (Direction)

This simple type defines a direction of either horizontal or vertical.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
horz (Horizontal)	Defines a horizontal direction.
vert (Vertical)	Defines a vertical direction.

Referenced By
blinds@dir (§4.6.18); checker@dir (§4.6.24); comb@dir (§4.6.30); guide@orient (§4.3.2.4); ph@orient (§4.4.1.33); randomBar@dir (§4.6.61); split@orient (§4.6.71)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Direction">
   <restriction base="xsd:token">
      <enumeration value="horz"/>
      <enumeration value="vert"/>
   </restriction>
</simpleType>
```

4.8.6 ST_HtmlPublishWebBrowserSupport (Web browsers supported for HTML output)

This type specifies the Web Browsers supported for output to HTML.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
v3 (Browser v3)	Generate HTML optimized for Netscape Navigator 3.0, Microsoft Internet Explorer 3, and equivalent browsers.
v3v4 (Browser v3v4)	Generate HTML optimized for Netscape Navigator 3.0 and 4.0, Microsoft Internet Explorer 3 and 4, and equivalent browsers.
v4 (Browser v4)	Generate HTML optimized for Netscape Navigator 4.0, Microsoft Internet Explorer 4, and equivalent browsers.

Referenced By
htmlPubPr@pubBrowser (§4.3.1.13)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_HtmlPublishWebBrowserSupport">
   <restriction base="xsd:token">
      <enumeration value="v4"/>
      <enumeration value="v3"/>
      <enumeration value="v3v4"/>
   </restriction>
</simpleType>
```

4.8.7 ST_Index (Index)

This type defines the position of an object in an ordered list.

This simple type's contents are a restriction of the XML Schema unsignedInt datatype.

Referenced By
charRg@end (§4.6.23); charRg@st (§4.6.23); cm@idx (§4.5.1); pRg@end (§4.6.56); pRg@st (§4.6.56); sldRg@end (§4.2.8); sldRg@st (§4.2.8)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Index">
   <restriction base="xsd:unsignedInt"/>
</simpleType>
```

4.8.8 ST_IterateType (Iterate Type)

This type specifies how the animation will be applied over subelements of the target element.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
el (Element)	Iterate by element.
lt (Letter)	Iterate by Letter.
wd (Word)	Iterate by Word.

Referenced By
iterate@type (§4.6.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_IterateType">
   <restriction base="xsd:token">
      <enumeration value="el"/>
      <enumeration value="wd"/>
      <enumeration value="lt"/>
   </restriction>
</simpleType>
```

4.8.9 ST_Name (Name string)

This type specifies a name, such as for a comment author or custom show.

This simple type's contents are a restriction of the XML Schema string datatype.

Referenced By
cmAuthor@initials (§4.5.2); cmAuthor@name (§4.5.2); custShow@name (§4.3.1.5)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Name">
   <restriction base="xsd:string"/>
</simpleType>
```

4.8.10 ST_OleObjectFollowColorScheme (Embedded object to Follow Color Scheme)

This simple type determines if the Embedded object will be re-colored to reflect changes to the color schemes.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
full (Full)	Setting this enumeration causes the Embedded object to respond to all changes in the color scheme in the presentation.
none (None)	Setting this enumeration causes the Embedded object to not respond to changes in the color scheme in the presentation.
textAndBackground (Text and Background)	Setting this enumeration causes the Embedded object to respond only to changes in the text and background colors of the color scheme in the presentation.

Referenced By
embed@followColorScheme (§4.4.2.2)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_OleObjectFollowColorScheme">
    <restriction base="xsd:token">
        <enumeration value="none"/>
        <enumeration value="full"/>
        <enumeration value="textAndBackground"/>
    </restriction>
</simpleType>
```

4.8.11 ST_PhotoAlbumFrameShape (Photo Album Shape for Photo Mask)

This type specifies the values for photo frame types within a photo album presentation.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
frameStyle1 (Rectangle Photo Frame)	
frameStyle2 (Rounded Rectangle Photo Frame)	
frameStyle3 (Simple White Photo Frame)	
frameStyle4 (Simple Black Photo Frame)	
frameStyle5 (Compound Black Photo Frame)	
frameStyle6 (Center Shadow Photo Frame)	
frameStyle7 (Soft Edge Photo Frame)	

Referenced By
photoAlbum@frame (§4.3.1.22)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PhotoAlbumFrameShape">
   <restriction base="xsd:token">
      <enumeration value="frameStyle1"/>
      <enumeration value="frameStyle2"/>
      <enumeration value="frameStyle3"/>
      <enumeration value="frameStyle4"/>
      <enumeration value="frameStyle5"/>
      <enumeration value="frameStyle6"/>
      <enumeration value="frameStyle7"/>
   </restriction>
</simpleType>
```

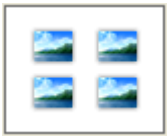
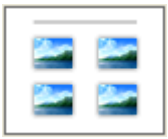
4.8.12 ST_PhotoAlbumLayout (Photo Album Layout Definition)

This type specifies the values for photo layouts within a photo album presentation.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
1pic (1 Photo per Slide)	 Specifies that photo album slides should have a single picture, centered horizontally and vertically, on the slide with no title.
1picTitle (1 Photo per Slide with Titles)	 Specifies that photo album slides should have a single picture and a single title text box, centered horizontally and vertically, on the slide.
2pic (2 Photos per Slide)	 Specifies that photo album slides should have two pictures of the same size, positioned side-by-side, centered horizontally and vertically, on the slide with no title.

Enumeration Value	Description
2picTitle (2 Photos per Slide with Titles)	 Specifies that photo album slides should have two pictures of the same size, positioned side-by-side, with a single title text box centered over them, collectively centered horizontally and vertically, on the slide.
4pic (4 Photos per Slide)	 Specifies that photo album slides should have four pictures of the same size, positioned in a two-by-two matrix, centered horizontally and vertically, on the slide with no title.
4picTitle (4 Photos per Slide with Titles)	 Specifies that photo album slides should have four pictures of the same size, positioned in a two-by-two matrix, with a single title text box centered over the matrix, centered horizontally and vertically, on the slide.
fitToSlide (Fit Photos to Slide)	 Specifies that photo album slides should have a single picture, stretched to fit the entire slide size, with no title.

Referenced By
photoAlbum@layout (§4.3.1.22)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PhotoAlbumLayout">
   <restriction base="xsd:token">
      <enumeration value="fitToSlide"/>
      <enumeration value="1pic"/>
      <enumeration value="2pic"/>
      <enumeration value="4pic"/>
      <enumeration value="1picTitle"/>
      <enumeration value="2picTitle"/>
      <enumeration value="4picTitle"/>
   </restriction>
</simpleType>
```

4.8.13 ST_PlaceholderSize (Placeholder Size)

This simple type facilitates the storing of the size of the placeholder. This size is described relative to the body placeholder on the master.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
full (Full)	Specifies that the placeholder should take the full size of the body placeholder on the master.
half (Half)	Specifies that the placeholder should take the half size of the body placeholder on the master. Half size vertically or horizontally? Needs a picture.
quarter (Quarter)	Specifies that the placeholder should take a quarter of the size of the body placeholder on the master. Picture would be helpful

Referenced By
ph@sz (§4.4.1.33)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PlaceholderSize">
   <restriction base="xsd:token">
      <enumeration value="full"/>
      <enumeration value="half"/>
      <enumeration value="quarter"/>
   </restriction>
</simpleType>
```

4.8.14 ST_PlaceholderType (Placeholder IDs)

This simple type facilitates the storing of the type of content a placeholder should contain.

[Note: Some types are not valid for all types of SlideBase. End note]

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
body (Body)	Contains body text. Valid for Slide, Slide Layout, Slide Master, Notes, Notes Master. Can be horizontal or vertical on Slide and Slide Layout.
chart (Chart)	Contains a chart or graph. Special type. Valid for Slide and Slide Layout.
clipArt (Clip Art)	Contains a single clip art image. Special type. Valid for Slide and Slide Layout.
ctrTitle (Centered Title)	Contains a title intended to be centered on the slide. Valid for Slide and Slide Layout.
dgm (Diagram)	Contains a diagram. Special type. Valid for Slide and Slide Layout.
dt (Date and Time)	Contains the date and time. Valid for Slide, Slide Layout, Slide Master, Notes, Notes Master, Handout Master
ftr (Footer)	Contains text to be used as a footer in the document. Valid for Slide, Slide Layout, Slide Master, Notes, Notes Master, Handout Master
hdr (Header)	Contains text to be used as a header for the document. Valid for Notes, Notes Master, Handout Master .
media (Media)	Contains multimedia content such as audio or a movie clip. Special type. Valid for Slide and Slide Layout.
obj (Object)	Contains any type of content. Special type. Valid for Slide and Slide Layout.
pic (Picture)	Contains a picture. Special type. Valid for Slide and Slide Layout.
sldImg (Slide Image)	Contains an image of the slide. Valid for Notes and Notes Master.
sldNum (Slide Number)	Contains the number of a slide. Valid for Slide, Slide Layout, Slide Master, Notes, Notes Master, Handout Master
subTitle (Subtitle)	Contains a subtitle. Valid for Slide and Slide Layout.
tbl (Table)	Contains a table. Special type. Valid for Slide and Slide Layout.
title (Title)	Contains a slide title. Valid for Slide, Slide Layout and

Enumeration Value	Description
	Slide Master. Can be horizontal or vertical on Slide and Slide Layout.

Referenced By
ph@type (§4.4.1.33)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PlaceholderType">
   <restriction base="xsd:token">
      <enumeration value="title"/>
      <enumeration value="body"/>
      <enumeration value="ctrTitle"/>
      <enumeration value="subTitle"/>
      <enumeration value="dt"/>
      <enumeration value="sldNum"/>
      <enumeration value="ftr"/>
      <enumeration value="hdr"/>
      <enumeration value="obj"/>
      <enumeration value="chart"/>
      <enumeration value="tbl"/>
      <enumeration value="clipArt"/>
      <enumeration value="dgm"/>
      <enumeration value="media"/>
      <enumeration value="sldImg"/>
      <enumeration value="pic"/>
   </restriction>
</simpleType>
```

4.8.15 ST_PrintColorMode (Print Color Mode)

This type specifies the color mode that should be used when printing a presentation document.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
bw (Black and White Mode)	Print should be in Black and White only
clr (Color Mode)	Print should be in Full Color
gray (Grayscale Mode)	Print should be in Grayscale only

Referenced By
prnPr@clrMode (§4.3.1.26)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PrintColorMode">
   <restriction base="xsd:token">
      <enumeration value="bw"/>
      <enumeration value="gray"/>
      <enumeration value="clr"/>
   </restriction>
</simpleType>
```

4.8.16 ST_PrintWhat (Default print output)

This type specifies the default print layout that should be used when printing

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
handouts1 (1 Slide / Handout Page)	1 Slide and Handout Page layout should be used.
handouts2 (2 Slides / Handout Page)	2 Slides and Handout Page layout should be used.
handouts3 (3 Slides / Handout Page)	3 Slides and Handout Page layout should be used.
handouts4 (4 Slides / Handout Page)	4 Slides and Handout Page layout should be used.
handouts6 (6 Slides / Handout Page)	6 Slides and Handout Page layout should be used.
handouts9 (9 Slides / Handout Page)	9 Slides and Handout Page layout should be used.
notes (Notes)	Notes layout should be used.
outline (Outline)	Outline layout should be used.
slides (Slides)	Slides layout should be used.

Referenced By
prnPr@prnWhat (§4.3.1.26)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PrintWhat">
   <restriction base="xsd:token">
      <enumeration value="slides"/>
      <enumeration value="handouts1"/>
      <enumeration value="handouts2"/>
      <enumeration value="handouts3"/>
      <enumeration value="handouts4"/>
      <enumeration value="handouts6"/>
      <enumeration value="handouts9"/>
      <enumeration value="notes"/>
      <enumeration value="outline"/>
   </restriction>
</simpleType>
```

4.8.17 ST_SlideId (Slide Identifier)

This type specifies the allowed numbering for the slide identifier.

This simple type's contents are a restriction of the XML Schema unsignedInt datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 256.
- This simple type has a maximum value of less than 2147483648.

Referenced By
sldId@id (§4.3.1.29)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SlideId">
   <restriction base="xsd:unsignedInt">
      <minInclusive value="256"/>
      <maxExclusive value="2147483648"/>
   </restriction>
</simpleType>
```

4.8.18 ST_SlideLayoutId (Slide Layout ID)

This simple type sets the bounds for the slide layout id value. This layout id is used to identify the different slide layout designs.

This simple type's contents are a restriction of the XML Schema unsignedInt datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 2147483648.

Referenced By
sldLayoutId@id (§4.4.1.37)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SlideLayoutId">
   <restriction base="xsd:unsignedInt">
      <minInclusive value="2147483648"/>
   </restriction>
</simpleType>
```

4.8.19 ST_SlideLayoutType (Slide Layout Type)

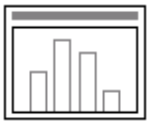
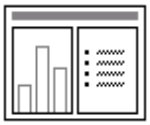
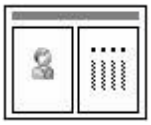
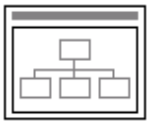
This simple type defines an arrangement of content on a slide. Each layout type is not tied to an exact positioning of placeholders, but rather provides a higher-level description of the type of content and positioning

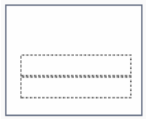
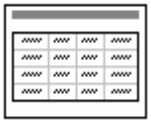
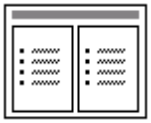
of placeholders. This information may be used by the application to aid in mapping between different layouts. The application may choose which, if any, of these layouts to make available through its user interface.

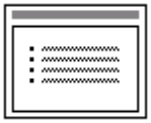
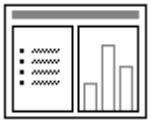
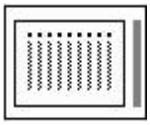
Each layout contains zero or more placeholders, each with a specific content type. An "object" placeholder may contain any type of data. Media placeholders are intended to hold video or audio clips. The enumeration value descriptions include illustrations of sample layouts for each value of the simple type.

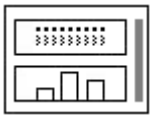
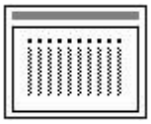
This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
blank (Slide Layout Type Enumeration (Blank))	 Blank
chart (Chart)	 Title and chart
chartAndTx (Slide Layout Type Enumeration (Chart and Text))	 Title, chart on left and text on right
clipArtAndTx (Clip Art and Text)	 Title, clipart on left, text on right
clipArtAndVertTx (Clip Art and Vertical Text)	 Title, clip art on left, vertical text on right
cust (Slide Layout Type Enumeration (Custom))	Custom layout defined by user
dgm (Slide Layout Type Enumeration (Diagram))	 Title and diagram
fourObj (Four Objects)	 Title and four objects
mediaAndTx (Slide Layout Type Enumeration (Media and Text))	 Title, media on left, text on right
obj (Title and Object)	 Title and object
objAndTwoObj (Object and Two Object)	Title, one object on left, two objects on right

Enumeration Value	Description
	
objAndTx (Slide Layout Type Enumeration (Object and Text))	 Title, object on left, text on right
objOnly (Object)	 Object only
objOverTx (Slide Layout Type Enumeration (Object over Text))	 Title, object on top, text on bottom
objTx (Title, Object, and Caption)	 Title, object and caption text
picTx (Picture and Caption)	 Title, picture, and caption text
secHead (Section Header)	 Section header title and subtitle text
tbl (Slide Layout Type Enumeration (Table))	 Title and table
title (Slide Layout Type Enumeration (Title))	 Title layout with centered title and subtitle placeholders
titleOnly (Slide Layout Type Enumeration (Title Only))	 Title only
twoColTx (Slide Layout Type Enumeration (Two Column Text))	 Title, text on left, text on right
twoObj (Two Objects)	 Title, object on left, object on right
twoObjAndObj (Two Objects and Object)	Title, two objects on left, one object on right

Enumeration Value	Description
	
twoObjAndTx (Two Objects and Text)	 Title, two objects on left, text on right
twoObjOverTx (Two Objects over Text)	 Title, two objects on top, text on bottom
twoTxTwoObj (Two Text and Two Objects)	 Title, two objects each with text
tx (Slide Layout Type Enumeration (Text))	 Title and text
txAndChart (Slide Layout Type Enumeration (Text and Chart))	 Title, text on left and chart on right
txAndClipArt (Text and Clip Art)	 Title, text on left, clip art on right
txAndMedia (Slide Layout Type Enumeration (Text and Media))	 Title, text on left, media on right
txAndObj (Slide Layout Type Enumeration (Text and Object))	 Title, text on left, object on right
txAndTwoObj (Text and Two Objects)	 Title, text on left, two objects on right
txOverObj (Slide Layout Type Enumeration (Text over Object))	 Title, text on top, object on bottom
vertTitleAndTx (Vertical Title and Text)	 Vertical title on right, vertical text on left
vertTitleAndTxOverChart (Vertical Title and Text Over Chart)	Vertical title on right, vertical text on top, chart on bottom

Enumeration Value	Description
	
vertTx (Vertical Text)	 Title and vertical text body

Referenced By
sldLayout@type (§4.4.1.36)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SlideLayoutType">
   <restriction base="xsd:token">
      <enumeration value="title"/>
      <enumeration value="tx"/>
      <enumeration value="twoColTx"/>
      <enumeration value="tbl"/>
      <enumeration value="txAndChart"/>
      <enumeration value="chartAndTx"/>
      <enumeration value="dgm"/>
      <enumeration value="chart"/>
      <enumeration value="txAndClipArt"/>
      <enumeration value="clipArtAndTx"/>
      <enumeration value="titleOnly"/>
      <enumeration value="blank"/>
      <enumeration value="txAndObj"/>
      <enumeration value="objAndTx"/>
      <enumeration value="objOnly"/>
      <enumeration value="obj"/>
      <enumeration value="txAndMedia"/>
      <enumeration value="mediaAndTx"/>
      <enumeration value="objOverTx"/>
      <enumeration value="txOverObj"/>
      <enumeration value="txAndTwoObj"/>
      <enumeration value="twoObjAndTx"/>
      <enumeration value="twoObjOverTx"/>
      <enumeration value="fourObj"/>
      <enumeration value="vertTx"/>
      <enumeration value="clipArtAndVertTx"/>
      <enumeration value="vertTitleAndTx"/>
      <enumeration value="vertTitleAndTxOverChart"/>
      <enumeration value="twoObj"/>
      <enumeration value="objAndTwoObj"/>
      <enumeration value="twoObjAndObj"/>
      <enumeration value="cust"/>
      <enumeration value="secHead"/>
      <enumeration value="twoTxTwoObj"/>
      <enumeration value="objTx"/>
      <enumeration value="picTx"/>
   </restriction>
</simpleType>
```

4.8.20 ST_SlideMasterId (Slide Master ID)

This type specifies the allowed numbering for the slide master identifier.

This simple type's contents are a restriction of the XML Schema unsignedInt datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 2147483648.

Referenced By
sldMasterId@id (§4.3.1.32)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SlideMasterId">
   <restriction base="xsd:unsignedInt">
      <minInclusive value="2147483648"/>
   </restriction>
</simpleType>
```

4.8.21 ST_SlideSizeCoordinate (Slide Size Coordinate)

This type specifies the slide size coordinate in EMUs (English Metric Units).

This simple type's contents are a restriction of the ST_PositiveCoordinate32 simple type (§5.1.12.43).

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 914400.
- This simple type has a maximum value of less than or equal to 51206400.

Referenced By
sldSz@cx (§4.3.1.34); sldSz@cy (§4.3.1.34)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SlideSizeCoordinate">
   <restriction base="a:ST_PositiveCoordinate32">
      <minInclusive value="914400"/>
      <maxInclusive value="51206400"/>
   </restriction>
</simpleType>
```

4.8.22 ST_SlideSizeType (Slide Size Type)

This type specifies the type of slide size that the slide should be optimized for.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
35mm (35mm Film)	Slide size should be optimized for 35mm film output
A3 (A3)	Slide size should be optimized for A3 output
A4 (A4)	Slide size should be optimized for A4 output
B4ISO (B4ISO)	Slide size should be optimized for B4ISO output
B4JIS (B4JIS)	Slide size should be optimized for B4JIS output

Enumeration Value	Description
B5ISO (B5ISO)	Slide size should be optimized for B5ISO output
B5JIS (B5JIS)	Slide size should be optimized for B5JIS output
banner (Banner)	Slide size should be optimized for banner output
custom (Custom)	Slide size should be optimized for custom output
hagakiCard (Hagaki Card)	Slide size should be optimized for hagaki card output
ledger (Ledger)	Slide size should be optimized for ledger output
letter (Letter)	Slide size should be optimized for letter output
overhead (Overhead)	Slide size should be optimized for overhead output
screen16x10 (Screen 16x10)	Slide size should be optimized for 16x10 screen output
screen16x9 (Screen 16x9)	Slide size should be optimized for 16x9 screen output
screen4x3 (Screen 4x3)	Slide size should be optimized for 4x3 screen output

Referenced By
sldSz@type (§4.3.1.34)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SlideSizeType">
   <restriction base="xsd:token">
      <enumeration value="screen4x3"/>
      <enumeration value="letter"/>
      <enumeration value="A4"/>
      <enumeration value="35mm"/>
      <enumeration value="overhead"/>
      <enumeration value="banner"/>
      <enumeration value="custom"/>
      <enumeration value="ledger"/>
      <enumeration value="A3"/>
      <enumeration value="B4ISO"/>
      <enumeration value="B5ISO"/>
      <enumeration value="B4JIS"/>
      <enumeration value="B5JIS"/>
      <enumeration value="hagakiCard"/>
      <enumeration value="screen16x9"/>
      <enumeration value="screen16x10"/>
   </restriction>
</simpleType>
```

4.8.23 ST_SplitterBarState (Splitter Bar State)

This type specifies the state that the splitter bar should be shown in. The splitter bar separates a primary and secondary region within a viewing area.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
maximized (Max)	The primary region occupies the greatest amount of the viewing area allowed by the application.
minimized (Min)	The primary region occupies the least amount of the viewing area allowed by the application.
restored (Restored)	The primary region has a specific intermediate size.

Referenced By
normalViewPr@horzBarState (§4.3.2.6); normalViewPr@vertBarState (§4.3.2.6)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SplitterBarState">
   <restriction base="xsd:token">
      <enumeration value="minimized"/>
      <enumeration value="restored"/>
      <enumeration value="maximized"/>
   </restriction>
</simpleType>
```

4.8.24 ST_TLAnimateBehaviorCalcMode (Time List Animate Behavior Calculate Mode)

This type specifies how the animation flows from point to point.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
discrete (Calc Mode Enum (Discrete))	Descrete
fmla (Calc Mode Enum (Formula))	Formula
lin (Calc Mode Enum (Linear))	Linear

Referenced By
anim@calcmode (§4.6.1)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLAnimateBehaviorCalcMode">
   <restriction base="xsd:token">
      <enumeration value="discrete"/>
      <enumeration value="lin"/>
      <enumeration value="fmla"/>
   </restriction>
</simpleType>
```

4.8.25 ST_TLAnimateBehaviorValueType (Time List Animate Behavior Value Types)

This type specifies the type of property value.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
clr (Value Type Enum (Color))	Color
num (Value Type Enum (Number))	Number
str (Value Type Enum (String))	String

Referenced By
anim@valueType (§4.6.1)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLAnimateBehaviorValueType">
   <restriction base="xsd:token">
      <enumeration value="str"/>
      <enumeration value="num"/>
      <enumeration value="clr"/>
   </restriction>
</simpleType>
```

4.8.26 ST_TLAnimateColorDirection (Time List Animate Color Direction)

This type specifies the direction in which to interpolate the animation (clockwise or counterclockwise).

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
ccw (Counter-Clockwise)	Counter-Clockwise
cw (Direction Enum (Clockwise))	Clockwise

Referenced By
animClr@dir (§4.6.2)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLAnimateColorDirection">
   <restriction base="xsd:token">
      <enumeration value="cw"/>
      <enumeration value="ccw"/>
   </restriction>
</simpleType>
```

4.8.27 ST_TLAnimateColorSpace (Time List Animate Color Space)

This type specifies the color space of the animation.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
hsl (Color Space Enum (HSL))	Hue, Saturation, Luminance
rgb (Color Space Enum (RGB))	Red, Green, Blue

Referenced By
animClr@clrSpc (§4.6.2)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLAnimateColorSpace">
   <restriction base="xsd:token">
      <enumeration value="rgb"/>
      <enumeration value="hsl"/>
   </restriction>
</simpleType>
```

4.8.28 ST_TLAnimateEffectTransition (Time List Animate Effect Transition)

This type specifies whether the effect is a transition in, transition out, or neither.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
in (Transition Enum (In))	In

Enumeration Value	Description
none (Transition Enum (None))	None
out (Transition Enum (Out))	Out

Referenced By
animEffect@transition (§4.6.3)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLAnimateEffectTransition">
   <restriction base="xsd:token">
      <enumeration value="in"/>
      <enumeration value="out"/>
      <enumeration value="none"/>
   </restriction>
</simpleType>
```

4.8.29 ST_TLAnimateMotionBehaviorOrigin (Time List Animate Motion Behavior Origin)

This type specifies what the origin of the motion path is relative to.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
layout (Origin Enum (Layout))	Layout
parent (Origin Enum (Parent))	Parent

Referenced By
animMotion@origin (§4.6.4)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLAnimateMotionBehaviorOrigin">
   <restriction base="xsd:token">
      <enumeration value="parent"/>
      <enumeration value="layout"/>
   </restriction>
</simpleType>
```

4.8.30 ST_TLAnimateMotionPathEditMode (Time List Animate Motion Path Edit Mode)

This type specifies how the motion path moves when the target element is moved.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
fixed (Path Edit Mode Enum (Fixed))	Fixed
relative (Path Edit Mode Enum (Relative))	Relative

Referenced By
animMotion@pathEditMode (§4.6.4)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLAnimateMotionPathEditMode">
   <restriction base="xsd:token">
      <enumeration value="relative"/>
      <enumeration value="fixed"/>
   </restriction>
</simpleType>
```

4.8.31 ST_TLBehaviorAccumulateType (Behavior Accumulate Type)

This type makes a repeating animation build with each iteration when set to "always."

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
always (Accumulate Enum (Always))	Always
none (Accumulate Enum (None))	None

Referenced By
cBhvr@accumulate (§4.6.22)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLBehaviorAccumulateType">
   <restriction base="xsd:token">
      <enumeration value="none"/>
      <enumeration value="always"/>
   </restriction>
</simpleType>
```

4.8.32 ST_TLBehaviorAdditiveType (Behavior Additive Type)

This type specifies how to apply the animation values to the original value for the property.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
base (Additive Enum (Base))	Base
mult (Additive Enum (Multiply))	Multiply
none (None)	None
repl (Additive Enum (Replace))	Replace
sum (Additive Enum (Sum))	Sum

Referenced By
cBhvr@additive (§4.6.22)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLBehaviorAdditiveType">
   <restriction base="xsd:token">
      <enumeration value="base"/>
      <enumeration value="sum"/>
      <enumeration value="repl"/>
      <enumeration value="mult"/>
      <enumeration value="none"/>
   </restriction>
</simpleType>
```

4.8.33 ST_TLBehaviorOverrideType (Behavior Override Type)

This type specifies how a behavior should override values of the attribute being animated on the target element. The "childStyle" will clear the attributes on the children contained inside the target element.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
childStyle (Override Enum (Child Style))	Child Style
normal (Override Enum (Normal))	Normal

Referenced By
cBhvr@override (§4.6.22)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLBehaviorOverrideType">
   <restriction base="xsd:token">
      <enumeration value="normal"/>
      <enumeration value="childStyle"/>
   </restriction>
</simpleType>
```

4.8.34 ST_TLBehaviorTransformType (Behavior Transform Type)

This type specifies how the behavior animates the target element.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
img (Image)	Image transform
pt (Point)	Point transform

Referenced By
cBhvr@xfrmType (§4.6.22)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLBehaviorTransformType">
   <restriction base="xsd:token">
      <enumeration value="pt"/>
      <enumeration value="img"/>
   </restriction>
</simpleType>
```

4.8.35 ST_TLChartSubelementType (Chart Subelement Type)

This type defines an animation target element that is represented by a subelement of a chart.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
category (Chart Build Element Type Enum (Category))	Category
gridLegend (Chart Build Element Type Enum (Grid Legend))	Background Element (Grid and Legend)
ptInCategory (Chart Build Element Type Enum (Point in Cat))	Category Element
ptInSeries (Chart Build Element Type Enum (Point in	Series Element

Enumeration Value	Description
Series))	
series (Chart Build Element Type Enum (Series))	Series

Referenced By
oleChartEl@type (§4.6.52)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLChartSubelementType">
   <restriction base="xsd:token">
      <enumeration value="gridLegend"/>
      <enumeration value="series"/>
      <enumeration value="category"/>
      <enumeration value="ptInSeries"/>
      <enumeration value="ptInCategory"/>
   </restriction>
</simpleType>
```

4.8.36 ST_TLCommandType (Command Type)

This type specifies a command type.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
call (Command Type Enum (Call))	Call
evt (Command Type Enum (Event))	Event
verb (Command Type Enum (Verb))	Verb

Referenced By
cmd@type (§4.6.28)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLCommandType">
   <restriction base="xsd:token">
      <enumeration value="evt"/>
      <enumeration value="call"/>
      <enumeration value="verb"/>
   </restriction>
</simpleType>
```

4.8.37 ST_TLDiagramBuildType (Diagram Build Types)

This type specifies the different diagram build types.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
allAtOnce (Diagram Build Type Enum (All At Once))	All At Once
breadthByLvl (Diagram Build Type Enum (Breadth By Level))	Breadth By Level
breadthByNode (Diagram Build Type Enum (Breadth By Node))	Breadth By Node
ccw (Diagram Build Type Enum (Counter-Clockwise))	Counter-Clockwise
ccwIn (Diagram Build Type Enum (Counter-Clockwise-In))	Counter-Clockwise-In
ccwOut (Diagram Build Type Enum (Counter-Clockwise-Out))	Counter-Clockwise-Out
cust (Diagram Build Type Enum (Custom))	Custom
cw (Diagram Build Type Enum (Clockwise))	Clockwise
cwIn (Diagram Build Type Enum (Clockwise-In))	Clockwise-In
cwOut (Diagram Build Type Enum (Clockwise-Out))	Clockwise-Out
depthByBranch (Diagram Build Type Enum (Depth By Branch))	Depth By Branch
depthByNode (Diagram Build Type Enum (Depth By Node))	Depth By Node
down (Diagram Build Type Enum (Down))	Down
inByRing (Diagram Build Type Enum (In-By-Ring))	In-By-Ring
outByRing (Diagram Build Type Enum (Out-By-Ring))	Out-By-Ring
up (Diagram Build Type Enum (Up))	Up
whole (Diagram Build Type Enum (Whole))	Whole

Referenced By
bldDgm@bld (§4.6.12)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLDiagramBuildType">
   <restriction base="xsd:token">
      <enumeration value="whole"/>
      <enumeration value="depthByNode"/>
      <enumeration value="depthByBranch"/>
      <enumeration value="breadthByNode"/>
      <enumeration value="breadthByLvl"/>
      <enumeration value="cw"/>
      <enumeration value="cwIn"/>
      <enumeration value="cwOut"/>
      <enumeration value="ccw"/>
      <enumeration value="ccwIn"/>
      <enumeration value="ccwOut"/>
      <enumeration value="inByRing"/>
      <enumeration value="outByRing"/>
      <enumeration value="up"/>
      <enumeration value="down"/>
      <enumeration value="allAtOnce"/>
      <enumeration value="cust"/>
   </restriction>
</simpleType>
```

4.8.38 ST_TLNextActionType (Next Action Type)

This type specifies what to do when going forward in a sequence. When the value is "seek," it will seek the current child element to its natural end time before advancing to the next element.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
none (Next Action Type Enum (None))	None
seek (Next Action Type Enum (Seek))	Seek

Referenced By
seq@nextAc (§4.6.65)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLNextActionType">
   <restriction base="xsd:token">
      <enumeration value="none"/>
      <enumeration value="seek"/>
   </restriction>
</simpleType>
```

4.8.39 ST_TLOleChartBuildType (Embedded Chart Build Type)

This type describes how to build an embedded Chart.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
allAtOnce (Chart Build Type Enum (All At Once))	All At Once
category (Chart Build Type Enum (Category))	By Category
categoryEl (Chart Build Type Enum (Category Element))	By Category Element
series (Chart Build Type Enum (Series))	By Series
seriesEl (Chart Build Type Enum (Series Element))	By Series Element

Referenced By
bldOleChart@bld (§4.6.15)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLOleChartBuildType">
   <restriction base="xsd:token">
      <enumeration value="allAtOnce"/>
      <enumeration value="series"/>
      <enumeration value="category"/>
      <enumeration value="seriesEl"/>
      <enumeration value="categoryEl"/>
   </restriction>
</simpleType>
```

4.8.40 ST_TLParaBuildType (Paragraph Build Type)

This type describes how to build a paragraph.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
allAtOnce (All At Once)	Specifies to animate all paragraphs at once.
cust (Custom)	Specifies the build has custom user settings.
p (Paragraph)	Specifies to animate paragraphs grouped by bullet level.
whole (Whole)	Specifies to animate the entire body of text as one

Enumeration Value	Description
	block.

Referenced By
bldP@build (§4.6.16)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLParaBuildType">
   <restriction base="xsd:token">
      <enumeration value="allAtOnce"/>
      <enumeration value="p"/>
      <enumeration value="cust"/>
      <enumeration value="whole"/>
   </restriction>
</simpleType>
```

4.8.41 ST_TLPreviousActionType (Previous Action Type)

This type specifies what to do when going backwards in a sequence. When the value is "skipTimed," the sequence will continue to go backwards until it reaches a sequence element that was defined to being only on a "next" event.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
none (Previous Action Type Enum (None))	None
skipTimed (Previous Action Type Enum (Skip Timed))	Skip Timed

Referenced By
seq@prevAc (§4.6.65)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLPreviousActionType">
   <restriction base="xsd:token">
      <enumeration value="none"/>
      <enumeration value="skipTimed"/>
   </restriction>
</simpleType>
```

4.8.42 ST_TLTime (Time)

This type specifies time after which to automatically advance the build to the next step. An amount of time, in milliseconds.

This simple type is defined as a union of the following types:

- TheXML Schema unsignedInt datatype.
- TheST_TLTimeIndefinite simple type (§4.8.44).

Referenced By
bldP@advAuto (§4.6.16); cond@delay (§4.6.31); cTn@dur (§4.6.33); cTn@repeatCount (§4.6.33); cTn@repeatDur (§4.6.33); endSync@delay (§4.6.39); tmAbs@val (§4.6.82)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLTime">
   <union memberTypes="xsd:unsignedInt ST_TLTimeIndefinite"/>
</simpleType>
```

4.8.43 ST_TLTimeAnimateValueTime (Animation Time)

This type specifies a percentage within the time span of the element. A value of indefinite means the attribute should be ignored.

This simple type is defined as a union of the following types:

- TheST_PositiveFixedPercentage simple type (§5.1.12.45).
- TheST_TLTimeIndefinite simple type (§4.8.44).

Referenced By
tav@tm (§4.6.79)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLTimeAnimateValueTime">
   <union memberTypes="a:ST_PositiveFixedPercentage ST_TLTimeIndefinite"/>
</simpleType>
```

4.8.44 ST_TLTimeIndefinite (Indefinite Time Declaration)

This type specifies a value that designates an "indefinite" amount time -- typically means this property is subordinate to other, defined properties.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
indefinite (Indefinite Type Enum)	Specifies Indefinite Time

Referenced By

Referenced By
ST_TLTime (§4.8.42); ST_TLTimeAnimateValueTime (§4.8.43)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLTimeIndefinite">
   <restriction base="xsd:token">
      <enumeration value="indefinite"/>
   </restriction>
</simpleType>
```

4.8.45 ST_TLTimeNodeFillType (Time Node Fill Type)

This type specifies what modifications the effect leaves on the target element's properties when the effect ends.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
freeze (Freeze)	Freeze
hold (TimeNode Fill Type Enum (Hold))	Hold
remove (Remove)	Remove
transition (Transition)	Transition

Referenced By
cTn@fill (§4.6.33)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLTimeNodeFillType">
   <restriction base="xsd:token">
      <enumeration value="remove"/>
      <enumeration value="freeze"/>
      <enumeration value="hold"/>
      <enumeration value="transition"/>
   </restriction>
</simpleType>
```

4.8.46 ST_TLTimeNodeID (Time Node ID)

This type represents a node or event on the timeline by its identifier.

This simple type's contents are a restriction of the XML Schema unsignedInt datatype.

Referenced By
cTn@id (§4.6.33); tn@val (§4.6.86)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLTimeNodeID">
   <restriction base="xsd:unsignedInt"/>
</simpleType>
```

4.8.47 ST_TLTimeNodeMasterRelation (Time Node Master Relation)

This type specifies how the time node plays back relative to its master time node.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
lastClick (TimeNode Master Relation Enum (Last Click))	Last Click
nextClick (TimeNode Master Relation Enum (Next Click))	Next Click
sameClick (TimeNode Master Relation Enum (Same Click))	Same Click

Referenced By
cTn@masterRel (§4.6.33)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLTimeNodeMasterRelation">
   <restriction base="xsd:token">
      <enumeration value="sameClick"/>
      <enumeration value="lastClick"/>
      <enumeration value="nextClick"/>
   </restriction>
</simpleType>
```

4.8.48 ST_TLTimeNodePresetClassType (Time Node Preset Class Type)

This type specifies the class of effect in which this effect belongs.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
emph (Preset Type Enum (Emphasis))	Emphasis Preset
entr (Preset Type Enum (Entrance))	Entrance Preset
exit (Exit)	Exit Preset

Enumeration Value	Description
mediacall (Preset Type Enum (Media Call))	Media Call Preset
path (Preset Type Enum (Path))	Path Preset
verb (Preset Type Enum (Verb))	Verb Preset

Referenced By
cTn@presetClass (§4.6.33)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLTimeNodePresetClassType">
   <restriction base="xsd:token">
      <enumeration value="entr"/>
      <enumeration value="exit"/>
      <enumeration value="emph"/>
      <enumeration value="path"/>
      <enumeration value="verb"/>
      <enumeration value="mediacall"/>
   </restriction>
</simpleType>
```

4.8.49 ST_TLTimeNodeRestartType (Time Node Restart Type)

This type determines whether an effect can play more than once.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
always (Restart Enum (Always))	Always restart node
never (Restart Enum (Never))	Never restart node
whenNotActive (Restart Enum (When Not Active))	Restart when node is not active

Referenced By
cTn@restart (§4.6.33)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLTimeNodeRestartType">
   <restriction base="xsd:token">
      <enumeration value="always"/>
      <enumeration value="whenNotActive"/>
      <enumeration value="never"/>
   </restriction>
</simpleType>
```

4.8.50 ST_TLTimeNodeSyncType (Time Node Sync Type)

This type specifies how the time node synchronizes to its group.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
canSlip (TimeNode Sync Enum (Can Slip))	Can Slip
locked (TimeNode Sync Enum (Locked))	Locked

Referenced By
cTn@syncBehavior (§4.6.33)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLTimeNodeSyncType">
   <restriction base="xsd:token">
      <enumeration value="canSlip"/>
      <enumeration value="locked"/>
   </restriction>
</simpleType>
```

4.8.51 ST_TLTimeNodeType (Time Node Type)

This type specifies time node types.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
afterEffect (Node Type Enum (After Effect))	After Effect
afterGroup (Node Type Enum (After Group))	After Group
clickEffect (Node Type Enum (Click Effect))	Click Effect
clickPar (Node Type Enum (Click Paragraph))	Click Paragraph
interactiveSeq (Node Type Enum (Interactive Sequence))	Interactive Sequence
mainSeq (Node Type Enum (Main Sequence))	Main Sequence
tmRoot (Node Type Enum (Timing Root))	Timing Root
withEffect (Node Type Enum (With Effect))	With Effect
withGroup (Node Type Enum (With Group))	With Group

Referenced By
cTn@nodeType (§4.6.33)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLTimeNodeType">
   <restriction base="xsd:token">
      <enumeration value="clickEffect"/>
      <enumeration value="withEffect"/>
      <enumeration value="afterEffect"/>
      <enumeration value="mainSeq"/>
      <enumeration value="interactiveSeq"/>
      <enumeration value="clickPar"/>
      <enumeration value="withGroup"/>
      <enumeration value="afterGroup"/>
      <enumeration value="tmRoot"/>
   </restriction>
</simpleType>
```

4.8.52 ST_TLTriggerEvent (Trigger Event)

This type specifies a particular event that causes the time condition to be true.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
begin (Trigger Event Enum (Begin))	Fire trigger at the beginning
end (Trigger Event Enum (End))	Fire trigger at the end
onBegin (Trigger Event Enum (On Begin))	Fire trigger at the beginning
onClick (Trigger Event Enum (On Click))	Fire trigger on a mouse click
onDblClick (Trigger Event Enum (On Double Click))	Fire trigger on double-mouse click
onEnd (Trigger Event Enum (On End))	Fire trigger at the end
onMouseOut (Trigger Event Enum (On Mouse Out))	Fire trigger on mouse out
onMouseOver (Trigger Event Enum (On Mouse Over))	Fire trigger on mouse over
onNext (Trigger Event Enum (On Next))	Fire trigger on next node
onPrev (Trigger Event Enum (On Previous))	Fire trigger on previous node
onStopAudio (Trigger Event Enum (On Stop Audio))	Fire trigger on stop audio

Referenced By
cond@evt (§4.6.31); endSync@evt (§4.6.39)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLTriggerEvent">
   <restriction base="xsd:token">
      <enumeration value="onBegin"/>
      <enumeration value="onEnd"/>
      <enumeration value="begin"/>
      <enumeration value="end"/>
      <enumeration value="onClick"/>
      <enumeration value="onDblClick"/>
      <enumeration value="onMouseOver"/>
      <enumeration value="onMouseOut"/>
      <enumeration value="onNext"/>
      <enumeration value="onPrev"/>
      <enumeration value="onStopAudio"/>
   </restriction>
</simpleType>
```

4.8.53 ST_TLTriggerRuntimeNode (Trigger RunTime Node)

This type specifies the child time node that triggers a time condition. References a child TimeNode or all child nodes. Order is based on the child's end time.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
all (Trigger RunTime Node Enum (All))	All
first (Trigger RunTime Node (First))	First
last (Trigger RunTime Node (Last))	Last

Referenced By
rtn@val (§4.6.64)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TLTriggerRuntimeNode">
   <restriction base="xsd:token">
      <enumeration value="first"/>
      <enumeration value="last"/>
      <enumeration value="all"/>
   </restriction>
</simpleType>
```

4.8.54 ST_TransitionCornerDirectionType (Transition Corner Direction Type)

This type specifies diagonal directions for slide transitions.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
ld (Transition Corner Direction Enum (Left-Down))	Specifies the slide transition direction of left-down
lu (Transition Corner Direction Enum (Left-Up))	Specifies the slide transition direction of left-up
rd (Transition Corner Direction Enum (Right-Down))	Specifies the slide transition direction of right-down
ru (Transition Corner Direction Enum (Right-Up))	Specifies the slide transition direction of right-up

Referenced By
ST_TransitionEightDirectionType (§4.8.55); strips@dir (§4.6.74)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TransitionCornerDirectionType">
   <restriction base="xsd:token">
      <enumeration value="lu"/>
      <enumeration value="ru"/>
      <enumeration value="ld"/>
      <enumeration value="rd"/>
   </restriction>
</simpleType>
```

4.8.55 ST_TransitionEightDirectionType (Transition Eight Direction)

This type specifies the direction of an animation.

This simple type is defined as a union of the following types:

- TheST_TransitionSideDirectionType simple type (§4.8.57).
- TheST_TransitionCornerDirectionType simple type (§4.8.54).

Referenced By
cover@dir (§4.6.32); pull@dir (§4.6.58)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TransitionEightDirectionType">
   <union memberTypes="ST_TransitionSideDirectionType ST_TransitionCornerDirectionType"/>
</simpleType>
```

4.8.56 ST_TransitionInOutDirectionType (Transition In/Out Direction Type)

This type specifies if a slide transition should go in or out.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
in (Transition In/Out Direction Enum (In))	Specifies the slide transition should go in
out (Transition In/Out Direction Enum (Out))	Specifies the slide transition should go out

Referenced By
split@dir (§4.6.71); zoom@dir (§4.6.97)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TransitionInOutDirectionType">
   <restriction base="xsd:token">
      <enumeration value="out"/>
      <enumeration value="in"/>
   </restriction>
</simpleType>
```

4.8.57 ST_TransitionSideDirectionType (Transition Slide Direction Type)

This type defines a set of slide transition directions.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
d (Transition Slide Direction Enum (Down))	Specifies that the transition direction is down
l (Transition Slide Direction Enum (Left))	Specifies that the transition direction is left
r (Transition Slide Direction (Right))	Specifies that the transition direction is right
u (Transition Slide Direction Enum (Up))	Specifies that the transition direction is up

Referenced By
push@dir (§4.6.59); ST_TransitionEightDirectionType (§4.8.55); wipe@dir (§4.6.96)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TransitionSideDirectionType">
   <restriction base="xsd:token">
      <enumeration value="l"/>
      <enumeration value="u"/>
      <enumeration value="r"/>
      <enumeration value="d"/>
   </restriction>
</simpleType>
```

4.8.58 ST_TransitionSpeed (Transition Speed)

This simple type defines the allowed transition speeds for transitioning from the current slide to the next.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
fast (Fast)	Fast slide transition.
med (Medium)	Medium slide transition.
slow (low)	Slow slide transition.

Referenced By
transition@spd (§4.4.1.46)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TransitionSpeed">
   <restriction base="xsd:token">
      <enumeration value="slow"/>
      <enumeration value="med"/>
      <enumeration value="fast"/>
   </restriction>
</simpleType>
```

4.8.59 ST_ViewType (List of View Types)

This type specifies the type of view that should be used when displaying the presentation document to the user.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
handoutView (Handout View)	Handout View mode should be used.
notesMasterView (Notes Master View)	Notes Master View mode should be used.
notesView (Notes View)	Notes View mode should be used.
outlineView (Outline View)	Outline View mode should be used.
sldMasterView (Slide Master View)	Slide Master View mode should be used.
sldSorterView (Slide Sorter View)	Slide Sorter View mode should be used.
sldThumbnailView (Slide Thumbnail View)	Slide Thumbnail View mode should be used.
sldView (Normal Slide View)	Normal Slide View mode should be used.

Referenced By
viewPr@lastView (§4.3.2.18)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ViewType">
   <restriction base="xsd:token">
      <enumeration value="sldView"/>
      <enumeration value="sldMasterView"/>
      <enumeration value="notesView"/>
      <enumeration value="handoutView"/>
      <enumeration value="notesMasterView"/>
      <enumeration value="outlineView"/>
      <enumeration value="sldSorterView"/>
      <enumeration value="sldThumbnailView"/>
   </restriction>
</simpleType>
```

4.8.60 ST_WebColorType (HTML Slide Navigation Control Colors)

This type specifies the coloring that should be used when outputting to web formats.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
blackTextOnWhite (Black Text on White Colors)	Black Text on White coloring should be used.
browser (Browser Colors)	Browser coloring should be used.
none (Non-specific Colors)	No specific coloring has been specified.
presentationAccent (Presentation Accent Colors)	Presentation accent coloring should be used.
presentationText (Presentation Text Colors)	Presentation text coloring should be used.
whiteTextOnBlack (White Text on Black Colors)	White text on black coloring should be used.

Referenced By
webPr@clr (§4.3.1.36)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_WebColorType">
   <restriction base="xsd:token">
      <enumeration value="none"/>
      <enumeration value="browser"/>
      <enumeration value="presentationText"/>
      <enumeration value="presentationAccent"/>
      <enumeration value="whiteTextOnBlack"/>
      <enumeration value="blackTextOnWhite"/>
   </restriction>
</simpleType>
```

4.8.61 ST_WebEncoding (Web Encoding)

This type specifies a string representing the HTML character set used when outputting to web formats.

This simple type's contents are a restriction of the XML Schema string datatype.

Referenced By
webPr@encoding (§4.3.1.36)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_WebEncoding">
   <restriction base="xsd:string"/>
</simpleType>
```

4.8.62 ST_WebScreenSize (HTML/Web Screen Size Target)

This type specifies the intended screen resolution for output to web formats.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
1024x768 (HTML/Web Size Enumeration 1024x768)	Screen size is 1024x768 pixels
1152x882 (HTML/Web Size Enumeration 1152x882)	Screen size is 1152x882 pixels
1152x900 (HTML/Web Size Enumeration 1152x900)	Screen size is 1152x900 pixels
1280x1024 (HTML/Web Size Enumeration 1280x1024)	Screen size is 1280x1024 pixels
1600x1200 (HTML/Web Size Enumeration 1600x1200)	Screen size is 1600x1200 pixels
1800x1400 (HTML/Web Size Enumeration 1800x1400)	Screen size is 1800x1400 pixels
1920x1200 (HTML/Web Size Enumeration 1920x1200)	Screen size is 1920x1200 pixels

Enumeration Value	Description
544x376 (HTML/Web Size Enumeration 544x376)	Screen size is 544x376 pixels
640x480 (HTML/Web Size Enumeration 640x480)	Screen size is 640x480 pixels
720x512 (HTML/Web Size Enumeration 720x515)	Screen size is 720x512 pixels
800x600 (HTML/Web Size Enumeration 800x600)	Screen size is 800x600 pixels

Referenced By
webPr@imgSz (§4.3.1.36)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_WebScreenSize">
   <restriction base="xsd:token">
      <enumeration value="544x376"/>
      <enumeration value="640x480"/>
      <enumeration value="720x512"/>
      <enumeration value="800x600"/>
      <enumeration value="1024x768"/>
      <enumeration value="1152x882"/>
      <enumeration value="1152x900"/>
      <enumeration value="1280x1024"/>
      <enumeration value="1600x1200"/>
      <enumeration value="1800x1400"/>
      <enumeration value="1920x1200"/>
   </restriction>
</simpleType>
```

5. DrawingML Reference Material

The subordinate subclauses specify the semantics for the XML markup comprising DrawingML content, which may be used within the contents of WordprocessingML, SpreadsheetML, or PresentationML documents.

5.1 DrawingML - Main

The DrawingML Main namespace defines all of the base constructs for all types of DrawingML objects (charts, diagrams, shapes, pictures, and so on). These constructs and primitives are defined below.

5.1.1 Table of Contents

This subclause is informative.

End of informative text.

5.1.2 Basics

This section describes all the basic common elements associated with the DrawingML framework.

5.1.2.1 Core Drawing Object Information

Within DrawingML, there is the notion of core drawing elements. These are elements that both are vital to and common across the DrawingML framework. These elements denote the most integral pieces of the DrawingML document structure and thus are among the most widely used.

[*Note*: Measurement Units - Length units shall be expressed in device-independent physical units: English Metric units (EMUs), points, picas, and inches. Device-dependent units such as pixels shall not be used. *end note*]

5.1.2.1.1 bldChart (Build Chart)

This element specifies how to build the animation for a diagram.

[*Example*: Consider the following example where a chart is specified to be animated by category rather than as one entity. Thus, the bldChart element should be used as follows:

```
<p:bdldLst>
  <p:bldGraphic spid="4" grpId="0">
    <p:bldSub>
      <a:bldChart bld="category"/>
    </p:bldSub>
  </p:bldGraphic>
</p:bdldLst>
```

end example]

Parent Elements
bldSub (§4.6.17)

Attributes	Description
animBg (Animate Background)	Specifies whether or not the chart background elements should be animated as well. An example of background elements are grid lines and the chart legend. The possible values for this attribute are defined by the XML Schema boolean datatype.
bld (Build)	Specifies how the chart will be built. The animation will animate the sub-elements in the container in the particular order defined by this attribute. The possible values for this attribute are defined by the ST_AnimationChartBuildType simple type (§5.1.12.5).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AnimationChartBuildProperties">
   <attribute name="bld" type="ST_AnimationChartBuildType" use="optional" default="allAtOnce"/>
   <attribute name="animBg" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.1.2.1.2 bldDgm (Build Diagram)

This element specifies how to build the animation for a diagram.

[*Example*: Consider having a diagram appear as on entity as opposed to by section. The bldDgm element should be used as follows:

```
<p:bdldLst>
  <p:bldGraphic spid="4" grpId="0">
    <p:bldSub>
      <a:bldDgm bld="one"/>
    </p:bldSub>
  </p:bldGraphic>
</p:bdldLst>
```

end example]

Parent Elements
bldSub (§4.6.17)

Attributes	Description
bld (Build)	Specifies how the chart will be built. The animation will animate the sub-elements in the container in the particular order defined by this attribute. The possible values for this attribute are defined by the ST_AnimationDgmBuildType simple type (§5.1.12.7).
rev (Reverse Animation)	Specifies whether the animation of the objects in this diagram should be reversed or not. If this attribute is not specified, a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AnimationDgmBuildProperties">
   <attribute name="bld" type="ST_AnimationDgmBuildType" use="optional" default="allAtOnce"/>
   <attribute name="rev" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.1.2.1.3 chart (Chart to Animate)

This element specifies a reference to a chart that should be animated within a sequence of slide animations. In addition to simply acting as a reference to a chart there is also animation build steps defined.

Parent Elements
graphicEl (§4.6.45)

Attributes	Description
bldStep (Animation Build Step)	Specifies which step this part of the chart should be built using. For instance the chart can be built as one object meaning it will be animated as a single graphic. Alternatively the chart can be animated, or built as separate pieces. The possible values for this attribute are defined by the ST_ChartBuildStep simple type (§5.1.12.13).
categoryIdx (Category Index)	Specifies the index of the category within the corresponding chart that should be animated. The possible values for this attribute are defined by the XML Schema int datatype.
seriesIdx (Series Index)	Specifies the index of the series within the corresponding chart that should be animated.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema int datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AnimationChartElement">
   <attribute name="seriesIdx" type="xsd:int" use="optional" default="-1"/>
   <attribute name="categoryIdx" type="xsd:int" use="optional" default="-1"/>
   <attribute name="bldStep" type="ST_ChartBuildStep" use="required"/>
</complexType>
```

5.1.2.1.4 cNvCxnSpPr (Non-Visual Connector Shape Drawing Properties)

This element specifies the non-visual drawing properties for a connector shape. These non-visual properties are properties that the generating application would utilize when rendering the slide surface.

Parent Elements
nvCxnSpPr (§5.1.2.1.25)

Child Elements	Subclause
cxnSpLocks (Connection Shape Locks)	§5.1.2.1.11
endCxn (Connection End)	§5.1.2.1.13
extLst (Extension List)	§5.1.2.1.15
stCxn (Connection Start)	§5.1.2.1.36

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualConnectorProperties">
   <sequence>
      <element name="cxnSpLocks" type="CT_ConnectorLocking" minOccurs="0" maxOccurs="1"/>
      <element name="stCxn" type="CT_Connection" minOccurs="0" maxOccurs="1"/>
      <element name="endCxn" type="CT_Connection" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.2.1.5 cNvGraphicFramePr (Non-Visual Graphic Frame Drawing Properties)

This element specifies the non-visual drawing properties for a graphic frame. These non-visual properties are properties that the generating application would utilize when rendering the slide surface.

Parent Elements
nvGraphicFramePr (§5.1.2.1.26)

Child Elements	Subclause

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
graphicFrameLocks (Graphic Frame Locks)	§5.1.2.1.19

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualGraphicFrameProperties">
   <sequence>
      <element name="graphicFrameLocks" type="CT_GraphicalObjectFrameLocking" minOccurs="0"
        maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.2.1.6 cNvGrpSpPr (Non-Visual Group Shape Drawing Properties)

This element specifies the non-visual drawing properties for a group shape. These non-visual properties are properties that the generating application would utilize when rendering the slide surface.

Parent Elements
nvGrpSpPr (§5.1.2.1.27)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
grpSpLocks (Group Shape Locks)	§5.1.2.1.21

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualGroupDrawingShapeProps">
   <sequence>
      <element name="grpSpLocks" type="CT_GroupLocking" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.2.1.7 cNvPicPr (Non-Visual Picture Drawing Properties)

This element specifies the non-visual properties for the picture canvas. These properties are to be used by the generating application to determine how certain properties are to be changed for the picture object in question.

[*Example*: Consider the following DrawingML.

```
<p:pic>
  …
  <p:nvPicPr>
    <p:cNvPr id="4" name="Lilly_by_Lisher.jpg"/>
    <p:cNvPicPr>
      <a:picLocks noChangeAspect="1"/>
    </p:cNvPicPr>
    <p:nvPr/>
  </p:nvPicPr>
  …
</p:pic>
```

end example]

Parent Elements
nvPicPr (§5.1.2.1.28)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
picLocks (Picture Locks)	§5.1.2.1.31

Attributes	Description
preferRelativeResize (Relative Resize Preferred)	Specifies if the user interface should show the resizing of the picture based on the picture's current size or its original size. If this attribute is set to true, then scaling will be relative to the original picture size as opposed to the current picture size. [*Example*: Consider the case where a picture has been resized within a document and is now 50% of the originally inserted picture size. Now if the user chooses to make a later adjustment to the size of this picture within the generating application, then the value of this attribute should be checked. If this attribute is set to true then a value of 50% will be shown. Similarly, if this attribute is set to false, then a value of 100% should be shown because the picture has not yet been resized from its current (smaller) size. *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualPictureProperties">
   <sequence>
      <element name="picLocks" type="CT_PictureLocking" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="preferRelativeResize" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.1.2.1.8 cNvPr (Non-Visual Drawing Properties)

This element specifies non-visual canvas properties. This allows for additional information that does not affect the appearance of the picture to be stored.

[*Example*: Consider the following DrawingML.

```
<p:pic>
  …
  <p:nvPicPr>
    <p:cNvPr id="4" name="Lilly_by_Lisher.jpg"/>
  </p:nvPicPr>
  …
</p:pic>
```

end example]

Parent Elements
nvCxnSpPr (§5.1.2.1.25); nvGraphicFramePr (§5.1.2.1.26); nvGrpSpPr (§5.1.2.1.27); nvPicPr (§5.1.2.1.28); nvSpPr (§5.1.2.1.29)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
hlinkClick (Click Hyperlink)	§5.1.5.3.5
hlinkHover (Hyperlink for Hover)	§5.1.2.1.23

Attributes	Description
descr (Alternative Text for Object)	Specifies alternative text for the current DrawingML object, for use by assistive technologies or applications which will not display the current object. If this element is omitted, then no alternative text is present for the parent object. [*Example*: Consider a DrawingML object defined as follows: `<… descr="A picture of a bowl of fruit">`

Attributes	Description
	The descr attribute contains alternative text which may be used in place of the actual DrawingML object. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
hidden (Hidden)	Specifies whether this DrawingML object shall be displayed. When a DrawingML object is displayed within a document, that object may be hidden (i.e., present, but not visible). This attribute shall determine whether the object shall be rendered or made hidden. [*Note*: An application may have settings which allow this object to be viewed. *end note*] If this attribute is omitted, then the parent DrawingML object shall be displayed (i.e., not hidden). [*Example*: Consider an inline DrawingML object which shall be hidden within the document's content. This setting would be specified as follows: `<… hidden="true" />` The hidden attribute has a value of true, which specifies that the DrawingML object is hidden and not displayed when the document is displayed. *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
id (Unique Identifier)	Specifies a unique identifier for the current DrawingML object within the current document. This ID may be used to assist in uniquely identifying this object so that it can be referred to by other parts of the document. If multiple objects within the same document share the same id attribute value, then the document shall be considered non-conformant. [*Example*: Consider a DrawingML object defined as follows: `<… id="10" … >` The id attribute has a value of 10, which is the unique identifier for this DrawingML object. *end example*] The possible values for this attribute are defined by the ST_DrawingElementId simple type (§5.1.12.19).
name (Name)	Specifies the name of the object. [*Note*: Typically, this will be used to store the original file name of a picture object. *end note*] [*Example*: Consider a DrawingML object defined as follows: `< … name="foo.jpg" >`

Attributes	Description
	The name attribute has a value of foo.jpg, which is the name of this DrawingML object. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualDrawingProps">
   <sequence>
      <element name="hlinkClick" type="CT_Hyperlink" minOccurs="0" maxOccurs="1"/>
      <element name="hlinkHover" type="CT_Hyperlink" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="id" type="ST_DrawingElementId" use="required"/>
   <attribute name="name" type="xsd:string" use="required"/>
   <attribute name="descr" type="xsd:string" use="optional" default=""/>
   <attribute name="hidden" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.1.2.1.9 cNvSpPr (Non-Visual Shape Drawing Properties)

This element specifies the non-visual drawing properties for a shape. These properties are to be used by the generating application to determine how the shape should be dealt with

[*Example*: Consider the shape that has a shape lock applied to it.

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="2" name="Rectangle 1"/>
    <p:cNvSpPr>
      <a:spLocks noGrp="1"/>
    </p:cNvSpPr>
  </p:nvSpPr>
…
</p:sp>
```

This shape lock is stored within the non-visual drawing properties for this shape. *end example*]

Parent Elements
nvSpPr (§5.1.2.1.29)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15

Child Elements	Subclause
spLocks (Shape Locks)	§5.1.2.1.34

Attributes	Description
txBox (Text Box)	Specifies that the corresponding shape is a text box and thus should be treated as such by the generating application. If this attribute is omitted then it is assumed that the corresponding shape is not specifically a text box. [*Note*: Because a shape is not specified to be a text box does not mean that it cannot have text attached to it. A text box is merely a specialized shape with specific properties. *end note*] The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualDrawingShapeProps">
   <sequence>
      <element name="spLocks" type="CT_ShapeLocking" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="txBox" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.1.2.1.10 cxnSp (Connection Shape)

This element specifies a connection shape that is used to connect two sp elements. Once a connection is specified using a cxnSp, it is left to the generating application to determine the exact path the connector will take. That is the connector routing algorithm is left up to the generating application as the desired path might be different depending on the specific needs of the application.

[*Example*: Consider the following connector shape that connects two regular shapes.

```
<p:spTree>
```

```
…
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="1" name="Rectangle 1"/>
    <p:cNvSpPr/>
    <p:nvPr/>
  </p:nvSpPr>
  …
</p:sp>
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="2" name="Rectangle 2"/>
    <p:cNvSpPr/>
    <p:nvPr/>
  </p:nvSpPr>
  …
</p:sp>
<p:cxnSp>
  <p:nvCxnSpPr>
    <p:cNvPr id="3" name="Elbow Connector 3"/>
    <p:cNvCxnSpPr>
      <a:stCxn id="1" idx="3"/>
      <a:endCxn id="2" idx="1"/>
    </p:cNvCxnSpPr>
    <p:nvPr/>
  </p:nvCxnSpPr>
  …
</p:cxnSp>
</p:spTree>
```

end example]

Parent Elements
grpSp (§5.1.2.1.20); lockedCanvas (§5.4.2.1)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
nvCxnSpPr (Non-Visual Properties for a Connection Shape)	§5.1.2.1.25
spPr (Shape Properties)	§5.1.2.1.35
style (Shape Style)	§5.1.2.1.37

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GvmlConnector">
   <sequence>
      <element name="nvCxnSpPr" type="CT_GvmlConnectorNonVisual" minOccurs="1" maxOccurs="1"/>
      <element name="spPr" type="CT_ShapeProperties" minOccurs="1" maxOccurs="1"/>
      <element name="style" type="CT_ShapeStyle" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.2.1.11 cxnSpLocks (Connection Shape Locks)

This element specifies all locking properties for a connection shape. These properties inform the generating application about specific properties that have been previously locked and thus should not be changed.

Parent Elements
cNvCxnSpPr (§5.6.2.4); cNvCxnSpPr (§4.4.1.8); cNvCxnSpPr (§5.8.2.3); cNvCxnSpPr (§5.1.2.1.4)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15

Attributes	Description
noAdjustHandles (Disallow Showing Adjust Handles)	Specifies that the generating application should not show adjust handles for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noChangeArrowheads (Disallow Arrowhead Changes)	Specifies that the generating application should not allow arrowhead changes for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noChangeAspect (Disallow Aspect Ratio Change)	Specifies that the generating application should not allow aspect ratio changes for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noChangeShapeType (Disallow Shape Type Change)	Specifies that the generating application should not allow shape type changes for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noEditPoints	Specifies that the generating application should not allow shape point changes for the

Attributes	Description
(Disallow Shape Point Editing)	corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noGrp (Disallow Shape Grouping)	Specifies that the generating application should not allow shape grouping for the corresponding connection shape. That is it cannot be combined within other shapes to form a group of shapes. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noMove (Disallow Shape Movement)	Specifies that the generating application should not allow position changes for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noResize (Disallow Shape Resize)	Specifies that the generating application should not allow size changes for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noRot (Disallow Shape Rotation)	Specifies that the generating application should not allow shape rotation changes for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noSelect (Disallow Shape Selection)	Specifies that the generating application should not allow selecting of the corresponding connection shape. That means also that no picture, shapes or text attached to this connection shape can be selected if this attribute has been specified. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ConnectorLocking">
   <sequence>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attributeGroup ref="AG_Locking"/>
</complexType>
```

5.1.2.1.12 dgm (Diagram to Animate)

This element specifies a reference to a diagram that should be animated within a sequence of slide animations. In addition to simply acting as a reference to a diagram there is also animation build steps defined.

Parent Elements

Parent Elements
graphicEl (§4.6.45)

Attributes	Description
bldStep (Animation Build Step)	Specifies which step this part of the diagram should be built using. For instance the diagram can be built as one object meaning it will be animated as a single graphic. Alternatively the diagram can be animated, or built as separate pieces. The possible values for this attribute are defined by the ST_DgmBuildStep simple type (§5.1.12.18).
id (Identifier)	Specifies the GUID of the shape for this build step in the animation. The possible values for this attribute are defined by the ST_Guid simple type (§5.1.12.27).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AnimationDgmElement">
   <attribute name="id" type="ST_Guid" use="optional" default="{00000000-0000-0000-0000-
     000000000000}"/>
   <attribute name="bldStep" type="ST_DgmBuildStep" use="optional" default="sp"/>
</complexType>
```

5.1.2.1.13 endCxn (Connection End)

This element specifies the ending connection that should be made by the corresponding connector shape. This connects the end tail of the connector to the final destination shape.

Parent Elements
cNvCxnSpPr (§5.6.2.4); cNvCxnSpPr (§4.4.1.8); cNvCxnSpPr (§5.8.2.3); cNvCxnSpPr (§5.1.2.1.4)

Attributes	Description
id (Identifier)	Specifies the id of the shape to make the final connection to. The possible values for this attribute are defined by the ST_DrawingElementId simple type (§5.1.12.19).
idx (Index)	Specifies the index into the connection site table of the final connection shape. That is there are many connection sites on a shape and it must be specified which connection site the corresponding connector shape should connect to. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Connection">
   <attribute name="id" type="ST_DrawingElementId" use="required"/>
   <attribute name="idx" type="xsd:unsignedInt" use="required"/>
</complexType>
```

5.1.2.1.14 ext (Extension)

This element specifies an extension that is used for future extensions to the current version of DrawingML. This allows for the specifying of currently unknown elements in the future that will be used for later versions of generating applications.

Parent Elements
extLst (§5.1.2.1.15); extLst (§5.9.2.13)

Child Elements	Subclause
Any element from any namespace	n/a

Attributes	Description
uri (Uniform Resource Identifier)	Specifies the URI, or uniform resource identifier that represents the data stored under this tag. The URI is used to identify the correct 'server' that can process the contents of this tag. The possible values for this attribute are defined by the XML Schema token datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OfficeArtExtension">
   <sequence>
      <any processContents="lax"/>
   </sequence>
   <attribute name="uri" type="xsd:token"/>
</complexType>
```

5.1.2.1.15 extLst (Extension List)

This element specifies the extension list within which all future extensions of type ext will be defined. The extension list along with corresponding future extensions is used to extend the storage capabilities of the DrawingML framework. This allows for various new types of data to be stored natively within the framework.

Parent Elements
audioCd (§5.1.3.1); audioFile (§5.1.3.2); backdrop (§5.1.7.2); blip (§5.1.10.13); bodyPr (§5.1.5.1.1); bodyStyle (§4.4.1.5); cell3D (§5.1.6.1); clrMap (§4.4.1.6); clrMap (§5.1.8.1); clrMapOvr (§5.7.2.30); clrScheme (§5.1.8.2); cNvCxnSpPr (§5.6.2.4); cNvCxnSpPr (§4.4.1.8); cNvCxnSpPr (§5.8.2.3); cNvCxnSpPr (§5.1.2.1.4); cNvGraphicFramePr (§4.4.1.9); cNvGraphicFramePr (§5.5.2.4); cNvGraphicFramePr (§5.8.2.4);

Parent Elements
cNvGraphicFramePr (§5.6.2.5); cNvGraphicFramePr (§5.1.2.1.5); cNvGrpSpPr (§5.1.2.1.6); cNvGrpSpPr (§5.8.2.5); cNvGrpSpPr (§5.6.2.6); cNvGrpSpPr (§4.4.1.10); cNvPicPr (§5.6.2.7); cNvPicPr (§4.4.1.11); cNvPicPr (§5.2.2.2); cNvPicPr (§5.1.2.1.7); cNvPicPr (§5.8.2.6); cNvPr (§5.2.2.3); cNvPr (§5.8.2.7); cNvPr (§4.4.1.12); cNvPr (§5.6.2.8); cNvPr (§5.1.2.1.8); cNvSpPr (§5.1.2.1.9); cNvSpPr (§5.8.2.8); cNvSpPr (§5.6.2.9); cNvSpPr (§4.4.1.13); cxnSp (§5.1.2.1.10); cxnSpLocks (§5.1.2.1.11); defaultTextStyle (§4.3.1.7); defPPr (§5.1.5.2.2); defRPr (§5.1.5.3.2); docPr (§5.5.2.5); endParaRPr (§5.1.5.2.3); font (§5.1.4.2.13); fontScheme (§5.1.4.1.18); graphicFrame (§5.1.2.1.18); graphicFrameLocks (§5.1.2.1.19); gridCol (§5.1.6.2); grpSp (§5.1.2.1.20); grpSpLocks (§5.1.2.1.21); grpSpPr (§5.8.2.14); grpSpPr (§4.4.1.20); grpSpPr (§5.1.2.1.22); grpSpPr (§5.6.2.17); hlinkClick (§5.1.5.3.5); hlinkHover (§5.1.2.1.23); hlinkMouseOver (§5.1.5.3.6); ln (§5.1.2.1.24); lnB (§5.1.6.3); lnBlToTr (§5.1.6.4); lnDef (§5.1.4.1.20); lnL (§5.1.6.5); lnR (§5.1.6.6); lnT (§5.1.6.7); lnTlToBr (§5.1.6.8); lockedCanvas (§5.4.2.1); lstStyle (§5.1.5.4.12); lvl1pPr (§5.1.5.4.13); lvl2pPr (§5.1.5.4.14); lvl3pPr (§5.1.5.4.15); lvl4pPr (§5.1.5.4.16); lvl5pPr (§5.1.5.4.17); lvl6pPr (§5.1.5.4.18); lvl7pPr (§5.1.5.4.19); lvl8pPr (§5.1.5.4.20); lvl9pPr (§5.1.5.4.21); majorFont (§5.1.4.1.24); minorFont (§5.1.4.1.25); notesStyle (§4.4.1.25); objectDefaults (§5.1.8.7); otherStyle (§4.4.1.32); overrideClrMapping (§5.1.8.8); pic (§5.1.2.1.30); picLocks (§5.1.2.1.31); pPr (§5.1.5.2.7); quickTimeFile (§5.1.3.4); rPr (§5.1.5.3.9); scene3d (§5.1.4.1.26); scene3d (§5.9.5.5); sp (§5.1.2.1.33); sp3d (§5.1.7.12); sp3d (§5.9.5.6); spDef (§5.1.4.1.27); spLocks (§5.1.2.1.34); spPr (§5.6.2.29); spPr (§5.7.2.198); spPr (§5.8.2.23); spPr (§5.1.2.1.35); spPr (§4.4.1.41); spPr (§5.9.3.7); spPr (§5.2.2.6); tableStyle (§5.1.6.9); tblPr (§5.1.6.13); tblStyle (§5.1.4.2.26); tc (§5.1.6.14); tcBdr (§5.1.4.2.28); tcPr (§5.1.6.15); tcTxStyle (§5.1.4.2.30); theme (§5.1.8.9); themeElements (§5.1.8.10); titleStyle (§4.4.1.45); tr (§5.1.6.16); txDef (§5.1.4.1.28); txSp (§5.1.2.1.41); uLn (§5.1.5.3.14); videoFile (§5.1.3.6)

Child Elements	Subclause
ext (Extension)	§5.1.2.1.14

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OfficeArtExtensionList">
   <sequence>
      <group ref="EG_OfficeArtExtensionList" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.2.1.16 graphic (Graphic Object)

This element specifies the existence of a single graphic object. Document authors should refer to this element when they wish to persist a graphical object of some kind. The specification for this graphical object will be provided entirely by the document author and referenced within the graphicData child element.

Parent Elements
anchor (§5.5.2.3); graphicFrame (§5.1.2.1.18); graphicFrame (§5.8.2.12); graphicFrame (§5.6.2.15); graphicFrame (§4.4.1.18); inline (§5.5.2.8)

Child Elements	Subclause

Child Elements	Subclause
graphicData (Graphic Object Data)	§5.1.2.1.17

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GraphicalObject">
   <sequence>
      <element name="graphicData" type="CT_GraphicalObjectData"/>
   </sequence>
</complexType>
```

5.1.2.1.17 graphicData (Graphic Object Data)

This element specifies the reference to a graphic object within the document. This graphic object is provided entirely by the document authors who choose to persist this data within the document.

[*Note*: Depending on the type of graphical object used not every generating application that supports the OOXML framework will have the ability to render the graphical object. *end note*]

Parent Elements
graphic (§5.1.2.1.16)

Child Elements	Subclause
Any element from any namespace	n/a

Attributes	Description
uri (Uniform Resource Identifier)	Specifies the URI, or uniform resource identifier that represents the data stored under this tag. The URI is used to identify the correct 'server' that can process the contents of this tag. The possible values for this attribute are defined by the XML Schema token datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GraphicalObjectData">
   <sequence>
      <any minOccurs="0" maxOccurs="unbounded" processContents="strict"/>
   </sequence>
   <attribute name="uri" type="xsd:token"/>
</complexType>
```

5.1.2.1.18 graphicFrame (Graphic Frame)

This element specifies the existence of a graphics frame. This frame contains a graphic that was generated by an external source and needs a container in which to be displayed on the slide surface.

Parent Elements
grpSp (§5.1.2.1.20); lockedCanvas (§5.4.2.1)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
graphic (Graphic Object)	§5.1.2.1.16
nvGraphicFramePr (Non-Visual Properties for a Graphic Frame)	§5.1.2.1.26
xfrm (2D Transform for Individual Objects)	§5.1.9.6

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GvmlGraphicalObjectFrame">
   <sequence>
      <element name="nvGraphicFramePr" type="CT_GvmlGraphicFrameNonVisual" minOccurs="1"
        maxOccurs="1"/>
      <element ref="graphic" minOccurs="1" maxOccurs="1"/>
      <element name="xfrm" type="CT_Transform2D" minOccurs="1" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.2.1.19 graphicFrameLocks (Graphic Frame Locks)

This element specifies all locking properties for a graphic frame. These properties inform the generating application about specific properties that have been previously locked and thus should not be changed.

Parent Elements
cNvGraphicFramePr (§4.4.1.9); cNvGraphicFramePr (§5.5.2.4); cNvGraphicFramePr (§5.8.2.4); cNvGraphicFramePr (§5.6.2.5); cNvGraphicFramePr (§5.1.2.1.5)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15

Attributes	Description
noChangeAspect (Disallow Aspect Ratio Change)	Specifies that the generating application should not allow aspect ratio changes for the corresponding graphic frame. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noDrilldown (Disallow Selection of Child Shapes)	Specifies that the generating application should not allow selecting of objects within the corresponding graphic frame but allow selecting of the graphic frame itself. If this attribute is not specified, then a value of false is assumed.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema boolean datatype.
noGrp (Disallow Shape Grouping)	Specifies that the generating application should not allow shape grouping for the corresponding graphic frame. That is it cannot be combined within other shapes to form a group of shapes. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noMove (Disallow Shape Movement)	Specifies that the corresponding graphic frame cannot be moved. Objects that reside within the graphic frame can still be moved unless they also have been locked. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noResize (Disallow Shape Resize)	Specifies that the generating application should not allow size changes for the corresponding graphic frame. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noSelect (Disallow Shape Selection)	Specifies that the generating application should not allow selecting of the corresponding picture. That means also that no picture, shapes or text attached to this picture can be selected if this attribute has been specified. If this attribute is not specified, then a value of false is assumed. [*Note*: If this attribute is specified to be true then the graphic frame cannot be selected and the objects within the graphic frame cannot be selected as well. That is the entire graphic frame including all sub-parts are considered un-selectable. *end note*] The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GraphicalObjectFrameLocking">
   <sequence>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="noGrp" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="noDrilldown" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="noSelect" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="noChangeAspect" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="noMove" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="noResize" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.1.2.1.20 grpSp (Group shape)

This element specifies a group shape that represents many shapes grouped together. This shape is to be treated just as if it were a regular shape but instead of being described by a single geometry it is made up of all the shape geometries encompassed within it. Within a group shape each of the shapes that make up the group are

specified just as they normally would. The idea behind grouping elements however is that a single transform can apply to many shapes at the same time.

[*Example*: Consider the following group shape.

```
<p:grpSp>
  <p:nvGrpSpPr>
    <p:cNvPr id="10" name="Group 9"/>
    <p:cNvGrpSpPr/>
    <p:nvPr/>
  </p:nvGrpSpPr>
  <p:grpSpPr>
    <a:xfrm>
      <a:off x="838200" y="990600"/>
      <a:ext cx="2426208" cy="978408"/>
      <a:chOff x="838200" y="990600"/>
      <a:chExt cx="2426208" cy="978408"/>
    </a:xfrm>
  </p:grpSpPr>
  <p:sp>
  …
  </p:sp>
  <p:sp>
  …
  </p:sp>
  <p:sp>
  …
  </p:sp>
</p:grpSp>
```

In the above example we see three shapes specified within a single group. These three shapes have their position and sizes specified just as they normally would within the shape tree. The generating application should apply the transformation after the bounding box for the group shape has been calculated. *end example*]

Parent Elements
grpSp (§5.1.2.1.20); lockedCanvas (§5.4.2.1)

Child Elements	Subclause
cxnSp (Connection Shape)	§5.1.2.1.10
extLst (Extension List)	§5.1.2.1.15
graphicFrame (Graphic Frame)	§5.1.2.1.18
grpSp (Group shape)	§5.1.2.1.20

Child Elements	Subclause
grpSpPr (Visual Group Shape Properties)	§5.1.2.1.22
nvGrpSpPr (Non-Visual Properties for a Group Shape)	§5.1.2.1.27
pic (Picture)	§5.1.2.1.30
sp (Shape)	§5.1.2.1.33
txSp (Text Shape)	§5.1.2.1.41

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GvmlGroupShape">
   <sequence>
      <element name="nvGrpSpPr" type="CT_GvmlGroupShapeNonVisual" minOccurs="1" maxOccurs="1"/>
      <element name="grpSpPr" type="CT_GroupShapeProperties" minOccurs="1" maxOccurs="1"/>
      <choice minOccurs="0" maxOccurs="unbounded">
         <element name="txSp" type="CT_GvmlTextShape"/>
         <element name="sp" type="CT_GvmlShape"/>
         <element name="cxnSp" type="CT_GvmlConnector"/>
         <element name="pic" type="CT_GvmlPicture"/>
         <element name="graphicFrame" type="CT_GvmlGraphicalObjectFrame"/>
         <element name="grpSp" type="CT_GvmlGroupShape"/>
      </choice>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.2.1.21 grpSpLocks (Group Shape Locks)

This element specifies all locking properties for a connection shape. These properties inform the generating application about specific properties that have been previously locked and thus should not be changed.

Parent Elements
cNvGrpSpPr (§5.1.2.1.6); cNvGrpSpPr (§5.8.2.5); cNvGrpSpPr (§5.6.2.6); cNvGrpSpPr (§4.4.1.10)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15

Attributes	Description
noChangeAspect (Disallow Aspect Ratio Change)	Specifies that the generating application should not allow aspect ratio changes for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noGrp (Disallow Shape Grouping)	Specifies that the corresponding group shape cannot be grouped. That is it cannot be combined within other shapes to form a group of shapes. If this attribute is not specified,

Attributes	Description
	then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noMove (Disallow Moving Shape)	Specifies that the corresponding graphic frame cannot be moved. Objects that reside within the graphic frame can still be moved unless they also have been locked. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noResize (Disallow Shape Resizing)	Specifies that the corresponding group shape cannot be resized. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noRot (Disallow Shape Rotation)	Specifies that the corresponding group shape cannot be rotated Objects that reside within the group can still be rotated unless they also have been locked. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noSelect (Disallow Shape Selection)	Specifies that the corresponding group shape cannot have any part of it be selected. That means that no picture, shapes or attached text can be selected either if this attribute has been specified. If this attribute is not specified, then a value of false is assumed. [*Note*: This property is inherited by sub-elements and thus all shapes within the group shape cannot be selected when this attribute is set to a value of true. *end note*] The possible values for this attribute are defined by the XML Schema boolean datatype.
noUngrp (Disallow Shape Ungrouping)	Specifies that the generating application should not show adjust handles for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GroupLocking">
   <sequence>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="noGrp" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="noUngrp" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="noSelect" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="noRot" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="noChangeAspect" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="noMove" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="noResize" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.1.2.1.22 grpSpPr (Visual Group Shape Properties)

This element specifies the properties that are to be common across all of the shapes within the corresponding group. If there are any conflicting properties within the group shape properties and the individual shape properties then the individual shape properties should take precedence.

Parent Elements
grpSp (§5.1.2.1.20); lockedCanvas (§5.4.2.1)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
effectDag (Effect Container)	§5.1.10.25
effectLst (Effect Container)	§5.1.10.26
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
scene3d (3D Scene Properties)	§5.1.4.1.26
solidFill (Solid Fill)	§5.1.10.54
xfrm (2D Transform for Grouped Objects)	§5.1.9.5

Attributes	Description
bwMode (Black and White Mode)	Specifies that the group shape should be rendered using only black and white coloring. That is the coloring information for the group shape should be converted to either black or white when rendering the corresponding shapes. No gray is to be used in rendering this image, only stark black and stark white. [*Note*: This does not mean that the group shapes themselves are stored with only black and white color information. This attribute instead sets the rendering mode that the shapes will use when rendering. *end note*] The possible values for this attribute are defined by the ST_BlackWhiteMode simple type (§5.1.12.10).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GroupShapeProperties">
   <sequence>
      <element name="xfrm" type="CT_GroupTransform2D" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_FillProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_EffectProperties" minOccurs="0" maxOccurs="1"/>
      <element name="scene3d" type="CT_Scene3D" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="bwMode" type="ST_BlackWhiteMode" use="optional"/>
</complexType>
```

5.1.2.1.23 hlinkHover (Hyperlink for Hover)

This element specifies the hyperlink information to be activated when the user's mouse is hovered over the corresponding object. The operation of the hyperlink is to have the specified action be activated when the mouse of the user hovers over the object. When this action is activated then additional attributes can be used to specify other tasks that should be performed along with the action.

Parent Elements
cNvPr (§5.2.2.3); cNvPr (§5.8.2.7); cNvPr (§4.4.1.12); cNvPr (§5.6.2.8); cNvPr (§5.1.2.1.8); docPr (§5.5.2.5)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
snd (Hyperlink Sound)	§5.1.2.1.32

Attributes	Description
action (Action Setting)	Specifies an action that is to be taken when this hyperlink is activated. This may be used to specify a slide to be navigated to or a script of code to be run. The possible values for this attribute are defined by the XML Schema string datatype.
endSnd (End Sounds)	Specifies if the URL in question should stop all sounds that are playing when it is clicked. The possible values for this attribute are defined by the XML Schema boolean datatype.
highlightClick (Highlight Click)	Specifies if this attribute has already been used within this document. That is when a hyperlink has already been visited that this attribute would be utilized so the generating application may determine the color of this text. If this attribute is omitted, then a value of 0 or false is implied. The possible values for this attribute are defined by the XML Schema boolean datatype.
history (Add Hyperlink to Page History)	Specifies whether to add this URI to the history when navigating to it. This allows for the viewing of this presentation without the storing of history information on the viewing machine. If this attribute is omitted, then a value of 1, or true is assumed.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema boolean datatype.
id (Drawing Object Hyperlink Target) Namespace: .../officeDocument/2006/relationships	Specifies the relationship id that when looked up in this slides relationship file will contain the target of this hyperlink. This attribute cannot be omitted. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
invalidUrl (Invalid URL)	Specifies the URL when it has been determined by the generating application that the URL is invalid. That is the generating application may still store the URL but it is known that this URL is not correct. The possible values for this attribute are defined by the XML Schema string datatype.
tgtFrame (Target Frame)	Specifies the target frame that is to be used when opening this hyperlink. When the hyperlink is activated this attribute will be used to determine if a new window must be launched for viewing or if an existing one may be used. If this attribute is omitted, than a new window will be opened. The possible values for this attribute are defined by the XML Schema string datatype.
tooltip (Hyperlink Tooltip)	Specifies the tooltip that should be displayed when the hyperlink text is hovered over with the mouse. If this attribute is omitted, than the hyperlink text itself may be displayed. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Hyperlink">
   <sequence>
      <element name="snd" type="CT_EmbeddedWAVAudioFile" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute ref="r:id" use="optional"/>
   <attribute name="invalidUrl" type="xsd:string" use="optional" default=""/>
   <attribute name="action" type="xsd:string" use="optional" default=""/>
   <attribute name="tgtFrame" type="xsd:string" use="optional" default=""/>
   <attribute name="tooltip" type="xsd:string" use="optional" default=""/>
   <attribute name="history" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="highlightClick" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="endSnd" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.1.2.1.24 ln (Outline)

This element specifies an outline style that can be applied to a number of different objects such as shapes and text. The line allows for the specifying of many different types of outlines including even line dashes and bevels.

Parent Elements
bottom (§5.1.4.2.6); defRPr (§5.1.5.3.2); endParaRPr (§5.1.5.2.3); insideH (§5.1.4.2.14); insideV (§5.1.4.2.15); left (§5.1.4.2.18); lnStyleLst (§5.1.4.1.21); right (§5.1.4.2.22); rPr (§5.1.5.3.9); spPr (§5.6.2.29); spPr (§5.7.2.198); spPr (§5.8.2.23); spPr (§5.1.2.1.35); spPr (§4.4.1.41); spPr (§5.9.3.7); spPr (§5.2.2.6); tl2br (§5.1.4.2.31); top (§5.1.4.2.32); tr2bl (§5.1.4.2.33); whole (§5.9.3.9)

Child Elements	Subclause
bevel (Line Join Bevel)	§5.1.10.9
custDash (Custom Dash)	§5.1.10.21
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
headEnd (Line Head/End Style)	§5.1.10.38
miter (Miter Line Join)	§5.1.10.43
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
prstDash (Preset Dash)	§5.1.10.48
round (Round Line Join)	§5.1.10.52
solidFill (Solid Fill)	§5.1.10.54
tailEnd (Tail line end style)	§5.1.10.57

Attributes	Description
algn (Stroke Alignment)	Specifies the alignment to be used for the underline stroke. The possible values for this attribute are defined by the ST_PenAlignment simple type (§5.1.12.40).
cap (Line Ending Cap Type)	Specifies the ending caps that should be used for this line. Examples of cap types are rounded, flat, etc. If this attribute is omitted, than a value of square is assumed. The possible values for this attribute are defined by the ST_LineCap simple type (§5.1.12.31).
cmpd (Compound Line Type)	Specifies the compound line type to be used for the underline stroke. If this attribute is omitted, then a value of sng is assumed. The possible values for this attribute are defined by the ST_CompoundLine simple type (§5.1.12.15).
w (Line Width)	Specifies the width to be used for the underline stroke. If this attribute is omitted, then a value of 0 is assumed. The possible values for this attribute are defined by the ST_LineWidth simple type

Attributes	Description
	(§5.1.12.35).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LineProperties">
   <sequence>
      <group ref="EG_LineFillProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_LineDashProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_LineJoinProperties" minOccurs="0" maxOccurs="1"/>
      <element name="headEnd" type="CT_LineEndProperties" minOccurs="0" maxOccurs="1"/>
      <element name="tailEnd" type="CT_LineEndProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="w" type="ST_LineWidth" use="optional"/>
   <attribute name="cap" type="ST_LineCap" use="optional"/>
   <attribute name="cmpd" type="ST_CompoundLine" use="optional"/>
   <attribute name="algn" type="ST_PenAlignment" use="optional"/>
</complexType>
```

5.1.2.1.25 nvCxnSpPr (Non-Visual Properties for a Connection Shape)

This element specifies all non-visual properties for a connection shape. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a connection shape. This allows for additional information that does not affect the appearance of the connection shape to be stored.

Parent Elements
cxnSp (§5.1.2.1.10)

Child Elements	Subclause
cNvCxnSpPr (Non-Visual Connector Shape Drawing Properties)	§5.1.2.1.4
cNvPr (Non-Visual Drawing Properties)	§5.1.2.1.8

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GvmlConnectorNonVisual">
   <sequence>
      <element name="cNvPr" type="CT_NonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
      <element name="cNvCxnSpPr" type="CT_NonVisualConnectorProperties" minOccurs="1"
        maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.2.1.26 nvGraphicFramePr (Non-Visual Properties for a Graphic Frame)

This element specifies all non-visual properties for a graphic frame. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a graphic

frame. This allows for additional information that does not affect the appearance of the graphic frame to be stored.

Parent Elements
graphicFrame (§5.1.2.1.18)

Child Elements	Subclause
cNvGraphicFramePr (Non-Visual Graphic Frame Drawing Properties)	§5.1.2.1.5
cNvPr (Non-Visual Drawing Properties)	§5.1.2.1.8

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GvmlGraphicFrameNonVisual">
   <sequence>
      <element name="cNvPr" type="CT_NonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
      <element name="cNvGraphicFramePr" type="CT_NonVisualGraphicFrameProperties" minOccurs="1"
        maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.2.1.27 nvGrpSpPr (Non-Visual Properties for a Group Shape)

This element specifies all non-visual properties for a group shape. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a group shape. This allows for additional information that does not affect the appearance of the group shape to be stored.

Parent Elements
grpSp (§5.1.2.1.20); lockedCanvas (§5.4.2.1)

Child Elements	Subclause
cNvGrpSpPr (Non-Visual Group Shape Drawing Properties)	§5.1.2.1.6
cNvPr (Non-Visual Drawing Properties)	§5.1.2.1.8

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GvmlGroupShapeNonVisual">
   <sequence>
      <element name="cNvPr" type="CT_NonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
      <element name="cNvGrpSpPr" type="CT_NonVisualGroupDrawingShapeProps" minOccurs="1"
        maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.2.1.28 nvPicPr (Non-Visual Properties for a Picture)

This element specifies all non-visual properties for a picture. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a picture. This allows for additional information that does not affect the appearance of the picture to be stored.

[*Example*: Consider the following PresentationML.

```
<p:pic>
  …
  <p:nvPicPr>
    …
  </p:nvPicPr>
  …
</p:pic>
```

end example]

Parent Elements
pic (§5.1.2.1.30)

Child Elements	Subclause
cNvPicPr (Non-Visual Picture Drawing Properties)	§5.1.2.1.7
cNvPr (Non-Visual Drawing Properties)	§5.1.2.1.8

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GvmlPictureNonVisual">
   <sequence>
      <element name="cNvPr" type="CT_NonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
      <element name="cNvPicPr" type="CT_NonVisualPictureProperties" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.2.1.29 nvSpPr (Non-Visual Properties for a Shape)

This element specifies all non-visual properties for a shape. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a shape. This allows for additional information that does not affect the appearance of the shape to be stored.

Parent Elements
sp (§5.1.2.1.33)

Child Elements	Subclause
cNvPr (Non-Visual Drawing Properties)	§5.1.2.1.8

Child Elements	Subclause
cNvSpPr (Non-Visual Shape Drawing Properties)	§5.1.2.1.9

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GvmlShapeNonVisual">
   <sequence>
      <element name="cNvPr" type="CT_NonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
      <element name="cNvSpPr" type="CT_NonVisualDrawingShapeProps" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.2.1.30 pic (Picture)

This element specifies the existence of a picture object within the document.

[*Example*: Consider the following PresentationML that specifies the existence of a picture within a document. This picture can have non-visual properties, a picture fill as well as shape properties attached to it.

```
<p:pic>
  <p:nvPicPr>
    <p:cNvPr id="4" name="lake.JPG" descr="Picture of a Lake" />
    <p:cNvPicPr>
      <a:picLocks noChangeAspect="1"/>
    </p:cNvPicPr>
    <p:nvPr/>
  </p:nvPicPr>
  <p:blipFill>
  …
  </p:blipFill>
  <p:spPr>
  …
  </p:spPr>
</p:pic>
```

end example]

Parent Elements
grpSp (§5.1.2.1.20); lockedCanvas (§5.4.2.1)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
extLst (Extension List)	§5.1.2.1.15
nvPicPr (Non-Visual Properties for a Picture)	§5.1.2.1.28

Child Elements	Subclause
spPr (Shape Properties)	§5.1.2.1.35
style (Shape Style)	§5.1.2.1.37

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GvmlPicture">
   <sequence>
      <element name="nvPicPr" type="CT_GvmlPictureNonVisual" minOccurs="1" maxOccurs="1"/>
      <element name="blipFill" type="CT_BlipFillProperties" minOccurs="1" maxOccurs="1"/>
      <element name="spPr" type="CT_ShapeProperties" minOccurs="1" maxOccurs="1"/>
      <element name="style" type="CT_ShapeStyle" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.2.1.31 picLocks (Picture Locks)

This element specifies all locking properties for a graphic frame. These properties inform the generating application about specific properties that have been previously locked and thus should not be changed.

Parent Elements
cNvPicPr (§5.6.2.7); cNvPicPr (§4.4.1.11); cNvPicPr (§5.2.2.2); cNvPicPr (§5.1.2.1.7); cNvPicPr (§5.8.2.6)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15

Attributes	Description
noAdjustHandles (Disallow Showing Adjust Handles)	Specifies that the generating application should not show adjust handles for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noChangeArrowheads (Disallow Arrowhead Changes)	Specifies that the generating application should not allow arrowhead changes for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noChangeAspect (Disallow Aspect Ratio Change)	Specifies that the generating application should not allow aspect ratio changes for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noChangeShapeTy	Specifies that the generating application should not allow shape type changes for the

Attributes	Description
pe (Disallow Shape Type Change)	corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noCrop (Disallow Crop Changes)	Specifies that the generating application should not allow cropping for the corresponding picture. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noEditPoints (Disallow Shape Point Editing)	Specifies that the generating application should not allow shape point changes for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noGrp (Disallow Shape Grouping)	Specifies that the generating application should not allow shape grouping for the corresponding connection shape. That is it cannot be combined within other shapes to form a group of shapes. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noMove (Disallow Shape Movement)	Specifies that the generating application should not allow position changes for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noResize (Disallow Shape Resize)	Specifies that the generating application should not allow size changes for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noRot (Disallow Shape Rotation)	Specifies that the generating application should not allow shape rotation changes for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noSelect (Disallow Shape Selection)	Specifies that the generating application should not allow selecting of the corresponding connection shape. That means also that no picture, shapes or text attached to this connection shape can be selected if this attribute has been specified. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PictureLocking">
   <sequence>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attributeGroup ref="AG_Locking"/>
   <attribute name="noCrop" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.1.2.1.32 snd (Hyperlink Sound)

This element specifies a sound to be played when a hyperlink within the document is activated. This sound is specified from within the parent hyperlink element.

Parent Elements
hlinkClick (§5.1.5.3.5); hlinkHover (§5.1.2.1.23); hlinkMouseOver (§5.1.5.3.6)

Attributes	Description
builtIn (Recognized Built-In Sound)	Specifies whether or not this sound is a built-in sound. If this attribute is set to true then the generating application is alerted to check the name attribute specified for this sound in it's list of built-in sounds and can then surface a custom name or UI as needed. The possible values for this attribute are defined by the XML Schema boolean datatype.
embed (Embedded Audio File Relationship ID) Namespace: .../officeDocument/2006/relationships	Specifies the identification information for an embedded audio file. This attribute is used to specify the location of an object that resides locally within the file. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
name (Sound Name)	Specifies the original name or given short name for the corresponding sound. This is used to distinguish this sound from others by providing a human readable name for the attached sound should the user need to identify the sound among others within the UI. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EmbeddedWAVAudioFile">
   <attribute ref="r:embed" use="required"/>
   <attribute name="name" type="xsd:string" use="optional" default=""/>
   <attribute name="builtIn" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.1.2.1.33 sp (Shape)

This element specifies the existence of a single shape. A shape can either be a preset or a custom geometry, defined using the DrawingML framework. In addition to a geometry each shape can have both visual and non-visual properties attached. Text and corresponding styling information can also be attached to a shape. This shape is specified along with all other shapes within either the shape tree or group shape elements.

[*Note*: Shapes are the preferred mechanism for specifying text on a slide. *end note*]

Parent Elements
grpSp (§5.1.2.1.20); lockedCanvas (§5.4.2.1)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
nvSpPr (Non-Visual Properties for a Shape)	§5.1.2.1.29
spPr (Shape Properties)	§5.1.2.1.35
style (Shape Style)	§5.1.2.1.37
txSp (Text Shape)	§5.1.2.1.41

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GvmlShape">
   <sequence>
      <element name="nvSpPr" type="CT_GvmlShapeNonVisual" minOccurs="1" maxOccurs="1"/>
      <element name="spPr" type="CT_ShapeProperties" minOccurs="1" maxOccurs="1"/>
      <element name="txSp" type="CT_GvmlTextShape" minOccurs="0" maxOccurs="1"/>
      <element name="style" type="CT_ShapeStyle" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.2.1.34 spLocks (Shape Locks)

This element specifies all locking properties for a shape. These properties inform the generating application about specific properties that have been previously locked and thus should not be changed.

Parent Elements
cNvSpPr (§5.1.2.1.9); cNvSpPr (§5.8.2.8); cNvSpPr (§5.6.2.9); cNvSpPr (§4.4.1.13)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15

Attributes	Description

Attributes	Description
noAdjustHandles (Disallow Showing Adjust Handles)	Specifies that the generating application should not show adjust handles for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noChangeArrowheads (Disallow Arrowhead Changes)	Specifies that the generating application should not allow arrowhead changes for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noChangeAspect (Disallow Aspect Ratio Change)	Specifies that the generating application should not allow aspect ratio changes for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noChangeShapeType (Disallow Shape Type Change)	Specifies that the generating application should not allow shape type changes for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noEditPoints (Disallow Shape Point Editing)	Specifies that the generating application should not allow shape point changes for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noGrp (Disallow Shape Grouping)	Specifies that the generating application should not allow shape grouping for the corresponding connection shape. That is it cannot be combined within other shapes to form a group of shapes. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noMove (Disallow Shape Movement)	Specifies that the generating application should not allow position changes for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noResize (Disallow Shape Resize)	Specifies that the generating application should not allow size changes for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noRot (Disallow Shape Rotation)	Specifies that the generating application should not allow shape rotation changes for the corresponding connection shape. If this attribute is not specified, then a value of false is assumed.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema boolean datatype.
noSelect (Disallow Shape Selection)	Specifies that the generating application should not allow selecting of the corresponding connection shape. That means also that no picture, shapes or text attached to this connection shape can be selected if this attribute has been specified. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
noTextEdit (Disallow Shape Text Editing)	Specifies that the generating application should not allow editing of the shape text for the corresponding shape. If this attribute is not specified, then a value of false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShapeLocking">
   <sequence>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attributeGroup ref="AG_Locking"/>
   <attribute name="noTextEdit" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.1.2.1.35 spPr (Shape Properties)

This element specifies the visual shape properties that can be applied to a shape.

Parent Elements
cxnSp (§5.1.2.1.10); lnDef (§5.1.4.1.20); pic (§5.1.2.1.30); sp (§5.1.2.1.33); spDef (§5.1.4.1.27); txDef (§5.1.4.1.28)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
custGeom (Custom Geometry)	§5.1.11.8
effectDag (Effect Container)	§5.1.10.25
effectLst (Effect Container)	§5.1.10.26
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35
ln (Outline)	§5.1.2.1.24
noFill (No Fill)	§5.1.10.44

Child Elements	Subclause
pattFill (Pattern Fill)	§5.1.10.47
prstGeom (Preset geometry)	§5.1.11.18
scene3d (3D Scene Properties)	§5.1.4.1.26
solidFill (Solid Fill)	§5.1.10.54
sp3d (Apply 3D shape properties)	§5.1.7.12
xfrm (2D Transform for Individual Objects)	§5.1.9.6

Attributes	Description
bwMode (Black and White Mode)	Specifies that the picture should be rendered using only black and white coloring. That is the coloring information for the picture should be converted to either black or white when rendering the picture. No gray is to be used in rendering this image, only stark black and stark white. [*Note*: This does not mean that the picture itself that is stored within the file is necessarily a black and white picture. This attribute instead sets the rendering mode that the picture will have applied to when rendering. *end note*] The possible values for this attribute are defined by the ST_BlackWhiteMode simple type (§5.1.12.10).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShapeProperties">
   <sequence>
      <element name="xfrm" type="CT_Transform2D" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_Geometry" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_FillProperties" minOccurs="0" maxOccurs="1"/>
      <element name="ln" type="CT_LineProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_EffectProperties" minOccurs="0" maxOccurs="1"/>
      <element name="scene3d" type="CT_Scene3D" minOccurs="0" maxOccurs="1"/>
      <element name="sp3d" type="CT_Shape3D" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="bwMode" type="ST_BlackWhiteMode" use="optional"/>
</complexType>
```

5.1.2.1.36 stCxn (Connection Start)

This element specifies the starting connection that should be made by the corresponding connector shape. This connects the head of the connector to the first shape.

Parent Elements
cNvCxnSpPr (§5.6.2.4); cNvCxnSpPr (§4.4.1.8); cNvCxnSpPr (§5.8.2.3); cNvCxnSpPr (§5.1.2.1.4)

Attributes	Description
id (Identifier)	Specifies the id of the shape to make the final connection to. The possible values for this attribute are defined by the ST_DrawingElementId simple type (§5.1.12.19).
idx (Index)	Specifies the index into the connection site table of the final connection shape. That is there are many connection sites on a shape and it must be specified which connection site the corresponding connector shape should connect to. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Connection">
   <attribute name="id" type="ST_DrawingElementId" use="required"/>
   <attribute name="idx" type="xsd:unsignedInt" use="required"/>
</complexType>
```

5.1.2.1.37 style (Shape Style)

This element specifies the style information for a shape.

Parent Elements
cxnSp (§5.1.2.1.10); lnDef (§5.1.4.1.20); pic (§5.1.2.1.30); sp (§5.1.2.1.33); spDef (§5.1.4.1.27); txDef (§5.1.4.1.28)

Child Elements	Subclause
effectRef (Effect Reference)	§5.1.4.2.8
fillRef (Fill Reference)	§5.1.4.2.10
fontRef (Font Reference)	§5.1.4.1.17
lnRef (Line Reference)	§5.1.4.2.19

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShapeStyle">
   <sequence>
      <element name="lnRef" type="CT_StyleMatrixReference" minOccurs="1" maxOccurs="1"/>
      <element name="fillRef" type="CT_StyleMatrixReference" minOccurs="1" maxOccurs="1"/>
      <element name="effectRef" type="CT_StyleMatrixReference" minOccurs="1" maxOccurs="1"/>
      <element name="fontRef" type="CT_FontReference" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.2.1.38 sx (Horizontal Ratio)

This element specifies the horizontal ratio for use within a scaling calculation.

Parent Elements
scale (§4.3.2.13)

Attributes	Description
d (Denominator)	Specifies the denominator to be used within the equation. The possible values for this attribute are defined by the XML Schema long datatype.
n (Numerator)	Specifies the numerator to be used within the equation. The possible values for this attribute are defined by the XML Schema long datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Ratio">
   <attribute name="n" type="xsd:long" use="required"/>
   <attribute name="d" type="xsd:long" use="required"/>
</complexType>
```

5.1.2.1.39 sy (Vertical Ratio)

This element specifies the vertical ratio for use within a scaling calculation.

Parent Elements
scale (§4.3.2.13)

Attributes	Description
d (Denominator)	Specifies the denominator to be used within the equation. The possible values for this attribute are defined by the XML Schema long datatype.
n (Numerator)	Specifies the numerator to be used within the equation. The possible values for this attribute are defined by the XML Schema long datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Ratio">
   <attribute name="n" type="xsd:long" use="required"/>
   <attribute name="d" type="xsd:long" use="required"/>
</complexType>
```

5.1.2.1.40 txBody (Shape Text Body)

This element specifies the existence of text to be contained within the corresponding shape. All visible text and visible text related properties are contained within this element. There can be multiple paragraphs and within paragraphs multiple runs of text.

Parent Elements
tc (§5.1.6.14); txSp (§5.1.2.1.41)

Child Elements	Subclause
bodyPr (Body Properties)	§5.1.5.1.1
lstStyle (Text List Styles)	§5.1.5.4.12
p (Text Paragraphs)	§5.1.5.2.6

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextBody">
   <sequence>
      <element name="bodyPr" type="CT_TextBodyProperties" minOccurs="1" maxOccurs="1"/>
      <element name="lstStyle" type="CT_TextListStyle" minOccurs="0" maxOccurs="1"/>
      <element name="p" type="CT_TextParagraph" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.1.2.1.41 txSp (Text Shape)

This element specifies the existence of a text shape within a parent shape. This text shape is specifically used for displaying text as it has only text related child elements.

Parent Elements
grpSp (§5.1.2.1.20); lockedCanvas (§5.4.2.1); sp (§5.1.2.1.33)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
txBody (Shape Text Body)	§5.1.2.1.40
useSpRect (Use Shape Text Rectangle)	§5.1.2.1.42
xfrm (2D Transform for Individual Objects)	§5.1.9.6

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GvmlTextShape">
   <sequence>
      <element name="txBody" type="CT_TextBody" minOccurs="1" maxOccurs="1"/>
      <choice>
         <element name="useSpRect" type="CT_GvmlUseShapeRectangle" minOccurs="1" maxOccurs="1"/>
         <element name="xfrm" type="CT_Transform2D" minOccurs="1" maxOccurs="1"/>
      </choice>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.2.1.42 useSpRect (Use Shape Text Rectangle)

This element specifies that the text rectangle from the parent shape should be used for this text shape. If this attribute is specified then the text rectangle, or text bounding box as it is also called should have the same dimensions as the text bounding box of the parent shape within which this text shape resides.

Parent Elements
txSp (§5.1.2.1.41)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GvmlUseShapeRectangle"/>
```

5.1.2.2 Colors

Given its own section within DrawingML Basics, colors are an integral part of the DrawingML framework. Colors are used in virtually every object to help describe it's appearance when it is rendered on the screen. Since not every generating application wishes to represent color in the same manner, it is possible to specify color in a number of different ways.

5.1.2.2.1 alpha (Alpha)

This element specifies its input color with the specific opacity, but with its color unchanged.

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

Attributes	Description

val (Value)	Specifies the opacity as expressed by a percentage value.
	
	[*Example*: The following represents a green solid fill which is 50% opaque
	
	`<a:solidFill>`
	`  <a:srgbClr val="00FF00">`
	`  <a:alpha val="50000"/>`
	`  </a:srgbClr>`
	`</a:solidFill>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_PositiveFixedPercentage simple type (§5.1.12.45).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PositiveFixedPercentage">
   <attribute name="val" type="ST_PositiveFixedPercentage" use="required"/>
</complexType>
```

5.1.2.2.2 alphaMod (Alpha Modulation)

This element specifies a more or less opaque version of its input color. An alpha modulate never increases the alpha beyond 100%. A 200% alpha modulate makes a input color twice as opaque as before. A 50% alpha modulate makes a input color half as opaque as before.

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

Attributes	Description

val (Value)	Specifies the opacity as expressed by a percentage relative to the input color.
	
	[*Example*: The following represents a green solid fill which is 50% opaque
	
	`<a:solidFill>`
	`  <a:srgbClr val="00FF00">`
	`  <a:alphaMod val="50000"/>`
	`  </a:srgbClr>`
	`</a:solidFill>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_PositivePercentage simple type (§5.1.12.46).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PositivePercentage">
   <attribute name="val" type="ST_PositivePercentage" use="required"/>
</complexType>
```

5.1.2.2.3 alphaOff (Alpha Offset)

This element specifies a more or less opaque version of its input color. Increases or decreases the input alpha percentage by the specified percentage offset. A 10% alpha offset increases a 50% opacity to 60%. A -10% alpha offset decreases a 50% opacity to 40%. The transformed alpha values are limited to a range of 0 to 100%. A 10% alpha offset increase to a 100% opaque object still results in 100% opacity.

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

Attributes	Description

val (Value)	Specifies the opacity as expressed by a percentage offset increase or decrease relative to the input color. Increases will never increase the opacity beyond 100%, decreases will never decrease the opacity below 0%.
	
	[*Example*: The following represents a green solid fill which is 90% opaque
	
	`<a:solidFill>`
	`  <a:srgbClr val="00FF00">`
	`  <a:alphaOff val="-10000"/>`
	`  </a:srgbClr>`
	`</a:solidFill>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_FixedPercentage simple type (§5.1.12.22).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FixedPercentage">
   <attribute name="val" type="ST_FixedPercentage" use="required"/>
</complexType>
```

5.1.2.2.4 blue (Blue)

This element specifies the input color with the specific blue component, but with the red and green color components unchanged.

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

Attributes	Description

val (Value)	Specifies the value of the blue component. The assigned value is specified as a percentage with 0% indicating minimal blue and 100% indicating maximum blue.
	
	[*Example*: The following manipulates the fill from having RGB value RRGGBB = (00, FF, 00) to value RRGGBB= (00, FF, FF)
	
	`<a:solidFill>`
	`  <a:srgbClr val="00FF00">`
	`  <a:blue val="100000"/>`
	`  </a:srgbClr>`
	`</a:solidFill>`

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Percentage">
   <attribute name="val" type="ST_Percentage" use="required"/>
</complexType>
```

5.1.2.2.5 blueMod (Blue Modification)

This element specifies the input color with its blue component modulated by the given percentage. A 50% blue modulate will reduce the blue component by half. A 200% blue modulate will double the blue component.

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

Attributes	Description

val (Value)	Specifies the value of the blue component. The assigned value is specified as a percentage with 0% indicating minimal blue and 100% indicating maximum blue.
	
	[*Example*: The following manipulates the fill from having RGB value RRGGBB = (00, FF, 00) to value RRGGBB= (00, FF, FF)
	
	`<a:solidFill>`
	`  <a:srgbClr val="00FF00">`
	`  <a:blue val="100000"/>`
	`  </a:srgbClr>`
	`</a:solidFill>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Percentage">
   <attribute name="val" type="ST_Percentage" use="required"/>
</complexType>
```

5.1.2.2.6 blueOff (Blue Offset)

This element specifies the input color with its blue component shifted, but with its red and green color components unchanged.

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

Attributes	Description

val (Value)	Specifies the value of the blue component. The assigned value is specified as a percentage with 0% indicating minimal blue and 100% indicating maximum blue.
	
	[*Example*: The following manipulates the fill from having RGB value RRGGBB = (00, FF, 00) to value RRGGBB= (00, FF, FF)
	
	`<a:solidFill>`
	`  <a:srgbClr val="00FF00">`
	`  <a:blue val="100000"/>`
	`  </a:srgbClr>`
	`</a:solidFill>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Percentage">
   <attribute name="val" type="ST_Percentage" use="required"/>
</complexType>
```

5.1.2.2.7 comp (Complement)

This element specifies that the color rendered should be the complement of its input color with the complement being defined as such. Two colors are called complementary if, when mixed they produce a shade of grey. For instance, the complement of red which is RGB (255, 0, 0) is cyan which is RGB (0, 255, 255).

Primary colors and secondary colors are typically paired in this way:

- red and cyan (where cyan is the mixture of green and blue)
- green and magenta (where magenta is the mixture of red and blue)
- blue and yellow (where yellow is the mixture of red and green)

[*Example*:

The following represents the complement of red:

```
<a:solidFill>
  <a:srgbClr val="FF0000">
    <a:comp/>
  </a:srgbClr>
</a:solidFill>
```

end example]

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ComplementTransform"/>
```

5.1.2.2.8 gamma (Gamma)

This element specifies that the output color rendered by the generating application should be the sRGB gamma shift of the input color.

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GammaTransform"/>
```

5.1.2.2.9 gray (Gray)

This element specifies a grayscale of its input color, taking into relative intensities of the red, green, and blue primaries.

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GrayscaleTransform"/>
```

5.1.2.2.10 green (Green)

This elements specifies the input color with the specified green component, but with its red and blue color components unchanged.

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32);

Parent Elements
sysClr (§5.1.2.2.33)

Attributes	Description

val (Value)	Specifies the value of the blue component. The assigned value is specified as a percentage with 0% indicating minimal blue and 100% indicating maximum blue.
	
	[*Example*: The following manipulates the fill from having RGB value RRGGBB = (00, FF, 00) to value RRGGBB= (00, FF, FF)
	
	`<a:solidFill>`
	`  <a:srgbClr val="00FF00">`
	`  <a:blue val="100000"/>`
	`  </a:srgbClr>`
	`</a:solidFill>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Percentage">
   <attribute name="val" type="ST_Percentage" use="required"/>
</complexType>
```

5.1.2.2.11 greenMod (Green Modification)

This element specifies the input color with its green component modulated by the given percentage. A 50% green modulate will reduce the green component by half. A 200% green modulate will double the green component.

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

Attributes	Description
val (Value)	Specifies the value of the blue component. The assigned value is specified as a percentage with 0% indicating minimal blue and 100% indicating maximum blue. [*Example*: The following manipulates the fill from having RGB value RRGGBB = (00, FF, 00) to value RRGGBB= (00, FF, FF) `<a:solidFill>`

Attributes	Description
	<a:srgbClr val="00FF00"> <a:blue val="100000"/> </a:srgbClr> </a:solidFill> *end example*] The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Percentage">
   <attribute name="val" type="ST_Percentage" use="required"/>
</complexType>
```

5.1.2.2.12 greenOff (Green Offset)

This element specifies the input color with its green component shifted, but with its red and blue color components unchanged.

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

Attributes	Description
val (Value)	Specifies the value of the blue component. The assigned value is specified as a percentage with 0% indicating minimal blue and 100% indicating maximum blue. [*Example*: The following manipulates the fill from having RGB value RRGGBB = (00, FF, 00) to value RRGGBB= (00, FF, FF) <a:solidFill> <a:srgbClr val="00FF00"> <a:blue val="100000"/> </a:srgbClr> </a:solidFill> *end example*] The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Percentage">
   <attribute name="val" type="ST_Percentage" use="required"/>
</complexType>
```

5.1.2.2.13 hslClr (Hue, Saturation, Luminance Color Model)

This element specifies a color using the HSL color model. A perceptual gamma of 2.2 is assumed.

Hue refers to the dominant wavelength of color, saturation refers to the purity of its hue, and luminance refers to its lightness or darkness.

As with all colors, colors defined with the HSL color model may have color transforms applied to it.

[*Example*:

The color blue having RGB value RRGGBB = (00, 00, 80) is equivalent to

```
<a:solidFill>
  <a:hslClr hue="14400000" sat="100000" lum="50000">
</a:solidFill>
```

end example]

Parent Elements
accent1 (§5.1.4.1.1); accent2 (§5.1.4.1.2); accent3 (§5.1.4.1.3); accent4 (§5.1.4.1.4); accent5 (§5.1.4.1.5); accent6 (§5.1.4.1.6); alphaInv (§5.1.10.4); bgClr (§5.1.10.10); bgRef (§4.4.1.3); buClr (§5.1.5.4.4); clrFrom (§5.1.10.17); clrMru (§4.3.1.4); clrRepl (§5.1.10.18); clrTo (§5.1.10.19); clrVal (§4.6.27); contourClr (§5.1.7.6); custClr (§5.1.4.1.8); dk1 (§5.1.4.1.9); dk2 (§5.1.4.1.10); duotone (§5.1.10.23); effectClrLst (§5.9.4.7); effectRef (§5.1.4.2.8); extrusionClr (§5.1.7.7); fgClr (§5.1.10.27); fillClrLst (§5.9.4.8); fillRef (§5.1.4.2.10); folHlink (§5.1.4.1.15); fontRef (§5.1.4.1.17); from (§4.6.44); glow (§5.1.10.32); gs (§5.1.10.36); highlight (§5.1.5.3.4); hlink (§5.1.4.1.19); innerShdw (§5.1.10.40); linClrLst (§5.9.4.9); lnRef (§5.1.4.2.19); lt1 (§5.1.4.1.22); lt2 (§5.1.4.1.23); outerShdw (§5.1.10.45); penClr (§4.3.1.21); prstShdw (§5.1.10.49); solidFill (§5.1.10.54); tcTxStyle (§5.1.4.2.30); to (§4.6.90); txEffectClrLst (§5.9.4.12); txFillClrLst (§5.9.4.13); txLinClrLst (§5.9.4.14)

Child Elements	Subclause
alpha (Alpha)	§5.1.2.2.1
alphaMod (Alpha Modulation)	§5.1.2.2.2
alphaOff (Alpha Offset)	§5.1.2.2.3
blue (Blue)	§5.1.2.2.4
blueMod (Blue Modification)	§5.1.2.2.5
blueOff (Blue Offset)	§5.1.2.2.6
comp (Complement)	§5.1.2.2.7
gamma (Gamma)	§5.1.2.2.8

Child Elements	Subclause
gray (Gray)	§5.1.2.2.9
green (Green)	§5.1.2.2.10
greenMod (Green Modification)	§5.1.2.2.11
greenOff (Green Offset)	§5.1.2.2.12
hue (Hue)	§5.1.2.2.14
hueMod (Hue Modulate)	§5.1.2.2.15
hueOff (Hue Offset)	§5.1.2.2.16
inv (Inverse)	§5.1.2.2.17
invGamma (Inverse Gamma)	§5.1.2.2.18
lum (Luminance)	§5.1.2.2.19
lumMod (Luminance Modulation)	§5.1.2.2.20
lumOff (Luminance Offset)	§5.1.2.2.21
red (Red)	§5.1.2.2.23
redMod (Red Modulation)	§5.1.2.2.24
redOff (Red Offset)	§5.1.2.2.25
sat (Saturation)	§5.1.2.2.26
satMod (Saturation Modulation)	§5.1.2.2.27
satOff (Saturation Offset)	§5.1.2.2.28
shade (Shade)	§5.1.2.2.31
tint (Tint)	§5.1.2.2.34

Attributes	Description
hue (Hue)	Specifies the angular value describing the wavelength. Expressed in 1/6000ths of a degree. The possible values for this attribute are defined by the ST_PositiveFixedAngle simple type (§5.1.12.44).
lum (Luminance)	Specifies the luminance referring to the lightness or darkness of the color. Expressed as a percentage with 0% referring to maximal dark (black) and 100% referring to maximal white. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
sat (Saturation)	Specifies the saturation referring to the purity of the hue. Expressed as a percentage with 0% referring to grey, 100% referring to the purest form of the hue. The possible values for this attribute are defined by the ST_Percentage simple type

Attributes	Description
	(§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HslColor">
   <sequence>
      <group ref="EG_ColorTransform" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="hue" type="ST_PositiveFixedAngle" use="required"/>
   <attribute name="sat" type="ST_Percentage" use="required"/>
   <attribute name="lum" type="ST_Percentage" use="required"/>
</complexType>
```

5.1.2.2.14 hue (Hue)

This element specifies the input color with the specified hue, but with its saturation and luminance unchanged.

[*Example*: The following two solid fills are equivalent.

```
<a:solidFill>
  <a:hslClr hue="14400000" sat="100000" lum="50000">
</a:solidFill>
<a:solidFill>
  <a:hslClr hue="0" sat="100000" lum="50000">
    <a:hue val="14400000"/>
  <a:hslClr/>
</a:solidFill>
```

end example]

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

Attributes	Description
val (Value)	Specifies the actual angle value to be used with the input color's hue component. The possible values for this attribute are defined by the ST_PositiveFixedAngle simple type (§5.1.12.44).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PositiveFixedAngle">
   <attribute name="val" type="ST_PositiveFixedAngle" use="required"/>
</complexType>
```

5.1.2.2.15 hueMod (Hue Modulate)

This element specifies the input color with its hue modulated by the given percentage. A 50% hue modulate decreases the angular hue value by half. A 200% hue modulate doubles the angular hue value.

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

Attributes	Description

val (Value)	Specifies the opacity as expressed by a percentage relative to the input color.
	
	[*Example*: The following represents a green solid fill which is 50% opaque
	
	`<a:solidFill>`
	`  <a:srgbClr val="00FF00">`
	`  <a:alphaMod val="50000"/>`
	`  </a:srgbClr>`
	`</a:solidFill>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_PositivePercentage simple type (§5.1.12.46).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PositivePercentage">
   <attribute name="val" type="ST_PositivePercentage" use="required"/>
</complexType>
```

5.1.2.2.16 hueOff (Hue Offset)

This element specifies the input color with its hue shifted, but with its saturation and luminance unchanged.

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

Attributes	Description
val (Value)	Specifies the actual angular value of the shift. The result of the shift must be between 0 and 360 degrees. Shifts resulting in angular values less than 0 will be treated as 0. Shifts resulting in angular values greater than 360 will be treated as 360. [*Example*: The following increases the hue angular value by 10 degrees.

Attributes	Description
	<a:solidFill> <a:hslClr hue="0" sat="100000" lum="50000"/> <a:hueOff val="600000"/> </a:solidFill> *end example*] The possible values for this attribute are defined by the ST_Angle simple type (§5.1.12.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Angle">
   <attribute name="val" type="ST_Angle" use="required"/>
</complexType>
```

5.1.2.2.17 inv (Inverse)

This element specifies the inverse of its input color. For example, the inverse of red (1, 0, 0) is cyan (0, 1, 1).

[*Example*:

The following represents cyan, the inverse of red:

```
<a:solidFill>
  <a:srgbClr val="FF0000">
    <a:inv/>
  </a:srgbClr>
</a:solidFill>
```

end example]

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_InverseTransform"/>
```

5.1.2.2.18 invGamma (Inverse Gamma)

This element specifies that the output color rendered by the generating application should be the inverse sRGB gamma shift of the input color.

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_InverseGammaTransform"/>
```

5.1.2.2.19 lum (Luminance)

This element specifies the input color with the specified luminance, but with its hue and saturation unchanged. Typically luminance values fall in the range [0%, 100%].

[*Example*:

The following two solid fills are equivalent:

```
<a:solidFill>
  <a:hslClr hue="14400000" sat="100000" lum="50000">
</a:solidFill>
<a:solidFill>
  <a:hslClr hue="14400000" sat="100000" lum="0">
    <a:lum val="50000"/>
  <a:hslClr/>
</a:solidFill>
```

end example]

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

Attributes	Description

val (Value)	Specifies the value of the blue component. The assigned value is specified as a percentage with 0% indicating minimal blue and 100% indicating maximum blue.
	
	[*Example*: The following manipulates the fill from having RGB value RRGGBB = (00, FF, 00) to value RRGGBB= (00, FF, FF)
	
	`<a:solidFill>`
	`  <a:srgbClr val="00FF00">`
	`  <a:blue val="100000"/>`
	`  </a:srgbClr>`
	`</a:solidFill>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Percentage">
   <attribute name="val" type="ST_Percentage" use="required"/>
</complexType>
```

5.1.2.2.20 lumMod (Luminance Modulation)

This element specifies the input color with its luminance modulated by the given percentage. A 50% luminance modulate will reduce the luminance by half. A 200% luminance modulate will double the luminance.

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

Attributes	Description

val (Value)	Specifies the value of the blue component. The assigned value is specified as a percentage with 0% indicating minimal blue and 100% indicating maximum blue.
	
	[*Example*: The following manipulates the fill from having RGB value RRGGBB = (00, FF, 00) to value RRGGBB= (00, FF, FF)
	
	`<a:solidFill>`
	`  <a:srgbClr val="00FF00">`
	`  <a:blue val="100000"/>`
	`  </a:srgbClr>`
	`</a:solidFill>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Percentage">
   <attribute name="val" type="ST_Percentage" use="required"/>
</complexType>
```

5.1.2.2.21 lumOff (Luminance Offset)

This element specifies the input color with its luminance shifted, but with its hue and saturation unchanged.

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

Attributes	Description

val (Value)	Specifies the value of the blue component. The assigned value is specified as a percentage with 0% indicating minimal blue and 100% indicating maximum blue.
	
	[*Example*: The following manipulates the fill from having RGB value RRGGBB = (00, FF, 00) to value RRGGBB= (00, FF, FF)
	
	`<a:solidFill>`
	`  <a:srgbClr val="00FF00">`
	`  <a:blue val="100000"/>`
	`  </a:srgbClr>`
	`</a:solidFill>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Percentage">
   <attribute name="val" type="ST_Percentage" use="required"/>
</complexType>
```

5.1.2.2.22 prstClr (Preset Color)

This element specifies a color which is bound to one of a predefined collection of colors.

[*Example*:

The following defines a solid fill bound to the "black" preset color.

```
<a:solidFill>
  <a:prstClr val="black">
</a:solidFill>
```

end example]

Parent Elements
accent1 (§5.1.4.1.1); accent2 (§5.1.4.1.2); accent3 (§5.1.4.1.3); accent4 (§5.1.4.1.4); accent5 (§5.1.4.1.5); accent6 (§5.1.4.1.6); alphaInv (§5.1.10.4); bgClr (§5.1.10.10); bgRef (§4.4.1.3); buClr (§5.1.5.4.4); clrFrom (§5.1.10.17); clrMru (§4.3.1.4); clrRepl (§5.1.10.18); clrTo (§5.1.10.19); clrVal (§4.6.27); contourClr (§5.1.7.6); custClr (§5.1.4.1.8); dk1 (§5.1.4.1.9); dk2 (§5.1.4.1.10); duotone (§5.1.10.23); effectClrLst (§5.9.4.7); effectRef (§5.1.4.2.8); extrusionClr (§5.1.7.7); fgClr (§5.1.10.27); fillClrLst (§5.9.4.8); fillRef (§5.1.4.2.10); folHlink (§5.1.4.1.15); fontRef (§5.1.4.1.17); from (§4.6.44); glow (§5.1.10.32); gs (§5.1.10.36); highlight (§5.1.5.3.4); hlink (§5.1.4.1.19); innerShdw (§5.1.10.40); linClrLst (§5.9.4.9); lnRef (§5.1.4.2.19); lt1 (§5.1.4.1.22); lt2 (§5.1.4.1.23); outerShdw (§5.1.10.45); penClr (§4.3.1.21); prstShdw (§5.1.10.49); solidFill (§5.1.10.54); tcTxStyle (§5.1.4.2.30); to (§4.6.90); txEffectClrLst (§5.9.4.12); txFillClrLst (§5.9.4.13); txLinClrLst (§5.9.4.14)

Child Elements	Subclause
alpha (Alpha)	§5.1.2.2.1
alphaMod (Alpha Modulation)	§5.1.2.2.2
alphaOff (Alpha Offset)	§5.1.2.2.3
blue (Blue)	§5.1.2.2.4
blueMod (Blue Modification)	§5.1.2.2.5
blueOff (Blue Offset)	§5.1.2.2.6
comp (Complement)	§5.1.2.2.7
gamma (Gamma)	§5.1.2.2.8
gray (Gray)	§5.1.2.2.9
green (Green)	§5.1.2.2.10
greenMod (Green Modification)	§5.1.2.2.11
greenOff (Green Offset)	§5.1.2.2.12
hue (Hue)	§5.1.2.2.14
hueMod (Hue Modulate)	§5.1.2.2.15
hueOff (Hue Offset)	§5.1.2.2.16
inv (Inverse)	§5.1.2.2.17
invGamma (Inverse Gamma)	§5.1.2.2.18
lum (Luminance)	§5.1.2.2.19
lumMod (Luminance Modulation)	§5.1.2.2.20
lumOff (Luminance Offset)	§5.1.2.2.21
red (Red)	§5.1.2.2.23
redMod (Red Modulation)	§5.1.2.2.24
redOff (Red Offset)	§5.1.2.2.25
sat (Saturation)	§5.1.2.2.26
satMod (Saturation Modulation)	§5.1.2.2.27
satOff (Saturation Offset)	§5.1.2.2.28
shade (Shade)	§5.1.2.2.31
tint (Tint)	§5.1.2.2.34

Attributes	Description
val (Value)	Specifies the actual preset color value. The possible values for this attribute are defined by the ST_PresetColorVal simple type (§5.1.12.48).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PresetColor">
   <sequence>
      <group ref="EG_ColorTransform" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="val" type="ST_PresetColorVal"/>
</complexType>
```

5.1.2.2.23 red (Red)

This element specifies the input color with the specified red component, but with its green and blue color components unchanged.

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

Attributes	Description

val (Value)	Specifies the value of the blue component. The assigned value is specified as a percentage with 0% indicating minimal blue and 100% indicating maximum blue.
	
	[*Example*: The following manipulates the fill from having RGB value RRGGBB = (00, FF, 00) to value RRGGBB= (00, FF, FF)
	
	`<a:solidFill>`
	`  <a:srgbClr val="00FF00">`
	`  <a:blue val="100000"/>`
	`  </a:srgbClr>`
	`</a:solidFill>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Percentage">
   <attribute name="val" type="ST_Percentage" use="required"/>
</complexType>
```

5.1.2.2.24 redMod (Red Modulation)

This element specifies the input color with its red component modulated by the given percentage. A 50% red modulate will reduce the red component by half. A 200% red modulate will double the red component.

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32);

Parent Elements
sysClr (§5.1.2.2.33)

Attributes	Description

val (Value)	Specifies the value of the blue component. The assigned value is specified as a percentage with 0% indicating minimal blue and 100% indicating maximum blue.
	
	[*Example*: The following manipulates the fill from having RGB value RRGGBB = (00, FF, 00) to value RRGGBB= (00, FF, FF)
	
	`<a:solidFill>`
	`  <a:srgbClr val="00FF00">`
	`  <a:blue val="100000"/>`
	`  </a:srgbClr>`
	`</a:solidFill>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Percentage">
   <attribute name="val" type="ST_Percentage" use="required"/>
</complexType>
```

5.1.2.2.25 redOff (Red Offset)

This element specifies the input color with its red component shifted, but with its green and blue color components unchanged.

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

Attributes	Description

val (Value)	Specifies the value of the blue component. The assigned value is specified as a percentage with 0% indicating minimal blue and 100% indicating maximum blue.
	
	[*Example*: The following manipulates the fill from having RGB value RRGGBB = (00, FF, 00) to value RRGGBB= (00, FF, FF)
	
	`<a:solidFill>`
	`  <a:srgbClr val="00FF00">`

Attributes	Description
	<a:blue val="100000"/> </a:srgbClr> </a:solidFill> *end example*] The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Percentage">
   <attribute name="val" type="ST_Percentage" use="required"/>
</complexType>
```

5.1.2.2.26 sat (Saturation)

This element specifies the input color with the specified saturation, but with its hue and luminance unchanged. Typically saturation values fall in the range [0%, 100%].

[*Example*:

The following two solid fills are equivalent:

```
<a:solidFill>
  <a:hslClr hue="14400000" sat="100000" lum="50000">
</a:solidFill>
<a:solidFill>
  <a:hslClr hue="14400000" sat="0" lum="50000">
    <a:sat val="100000"/>
  <a:hslClr/>
</a:solidFill>
```

end example]

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

Attributes	Description
val (Value)	Specifies the value of the blue component. The assigned value is specified as a percentage with 0% indicating minimal blue and 100% indicating maximum blue. [*Example*: The following manipulates the fill from having RGB value RRGGBB = (00, FF, 00) to value RRGGBB= (00, FF, FF)

Attributes	Description
	<a:solidFill> <a:srgbClr val="00FF00"> <a:blue val="100000"/> </a:srgbClr> </a:solidFill> *end example*] The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Percentage">
   <attribute name="val" type="ST_Percentage" use="required"/>
</complexType>
```

5.1.2.2.27 satMod (Saturation Modulation)

This element specifies the input color with its saturation modulated by the given percentage. A 50% saturation modulate will reduce the saturation by half. A 200% saturation modulate will double the saturation.

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

Attributes	Description
val (Value)	Specifies the value of the blue component. The assigned value is specified as a percentage with 0% indicating minimal blue and 100% indicating maximum blue. [*Example*: The following manipulates the fill from having RGB value RRGGBB = (00, FF, 00) to value RRGGBB= (00, FF, FF) <a:solidFill> <a:srgbClr val="00FF00"> <a:blue val="100000"/> </a:srgbClr> </a:solidFill> *end example*] The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Percentage">
   <attribute name="val" type="ST_Percentage" use="required"/>
</complexType>
```

5.1.2.2.28 satOff (Saturation Offset)

This element specifies the input color with its saturation shifted, but with its hue and luminance unchanged. A 10% offset to 20% saturation yields 30% saturation.

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

Attributes	Description

val (Value)	Specifies the value of the blue component. The assigned value is specified as a percentage with 0% indicating minimal blue and 100% indicating maximum blue.
	
	[*Example*: The following manipulates the fill from having RGB value RRGGBB = (00, FF, 00) to value RRGGBB= (00, FF, FF)
	
	`<a:solidFill>`
	`  <a:srgbClr val="00FF00">`
	`  <a:blue val="100000"/>`
	`  </a:srgbClr>`
	`</a:solidFill>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Percentage">
   <attribute name="val" type="ST_Percentage" use="required"/>
</complexType>
```

5.1.2.2.29 schemeClr (Scheme Color)

This element specifies a color bound to a user's theme. As with all elements which define a color, it is possible to apply a list of color transforms to the base color defined.

Parent Elements
accent1 (§5.1.4.1.1); accent2 (§5.1.4.1.2); accent3 (§5.1.4.1.3); accent4 (§5.1.4.1.4); accent5 (§5.1.4.1.5); accent6 (§5.1.4.1.6); alphaInv (§5.1.10.4); bgClr (§5.1.10.10); bgRef (§4.4.1.3); buClr (§5.1.5.4.4); clrFrom (§5.1.10.17); clrMru (§4.3.1.4); clrRepl (§5.1.10.18); clrTo (§5.1.10.19); clrVal (§4.6.27); contourClr (§5.1.7.6);

Parent Elements
custClr (§5.1.4.1.8); dk1 (§5.1.4.1.9); dk2 (§5.1.4.1.10); duotone (§5.1.10.23); effectClrLst (§5.9.4.7); effectRef (§5.1.4.2.8); extrusionClr (§5.1.7.7); fgClr (§5.1.10.27); fillClrLst (§5.9.4.8); fillRef (§5.1.4.2.10); folHlink (§5.1.4.1.15); fontRef (§5.1.4.1.17); from (§4.6.44); glow (§5.1.10.32); gs (§5.1.10.36); highlight (§5.1.5.3.4); hlink (§5.1.4.1.19); innerShdw (§5.1.10.40); linClrLst (§5.9.4.9); lnRef (§5.1.4.2.19); lt1 (§5.1.4.1.22); lt2 (§5.1.4.1.23); outerShdw (§5.1.10.45); penClr (§4.3.1.21); prstShdw (§5.1.10.49); solidFill (§5.1.10.54); tcTxStyle (§5.1.4.2.30); to (§4.6.90); txEffectClrLst (§5.9.4.12); txFillClrLst (§5.9.4.13); txLinClrLst (§5.9.4.14)

Child Elements	Subclause
alpha (Alpha)	§5.1.2.2.1
alphaMod (Alpha Modulation)	§5.1.2.2.2
alphaOff (Alpha Offset)	§5.1.2.2.3
blue (Blue)	§5.1.2.2.4
blueMod (Blue Modification)	§5.1.2.2.5
blueOff (Blue Offset)	§5.1.2.2.6
comp (Complement)	§5.1.2.2.7
gamma (Gamma)	§5.1.2.2.8
gray (Gray)	§5.1.2.2.9
green (Green)	§5.1.2.2.10
greenMod (Green Modification)	§5.1.2.2.11
greenOff (Green Offset)	§5.1.2.2.12
hue (Hue)	§5.1.2.2.14
hueMod (Hue Modulate)	§5.1.2.2.15
hueOff (Hue Offset)	§5.1.2.2.16
inv (Inverse)	§5.1.2.2.17
invGamma (Inverse Gamma)	§5.1.2.2.18
lum (Luminance)	§5.1.2.2.19
lumMod (Luminance Modulation)	§5.1.2.2.20
lumOff (Luminance Offset)	§5.1.2.2.21
red (Red)	§5.1.2.2.23
redMod (Red Modulation)	§5.1.2.2.24
redOff (Red Offset)	§5.1.2.2.25
sat (Saturation)	§5.1.2.2.26
satMod (Saturation Modulation)	§5.1.2.2.27
satOff (Saturation Offset)	§5.1.2.2.28
shade (Shade)	§5.1.2.2.31

Child Elements	Subclause
tint (Tint)	§5.1.2.2.34

Attributes	Description

val (Value)	Specifies the desired scheme.
	
	[*Example*: The following represents a color bound to the "lt1" theme color
	
	`<a:solidFill>`
	`  <a:schemeClr val="lt1"/>`
	`</a:solidFill>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_SchemeColorVal simple type (§5.1.12.54).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SchemeColor">
   <sequence>
      <group ref="EG_ColorTransform" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="val" type="ST_SchemeColorVal" use="required"/>
</complexType>
```

5.1.2.2.30 scrgbClr (RGB Color Model - Percentage Variant)

This element specifies a color using the red, green, blue RGB color model. Each component, red, green, and blue is expressed as a percentage from 0% to 100%. A linear gamma of 1.0 is assumed.

Specifies the level of red as expressed by a percentage offset increase or decrease relative to the input color.

[*Example*: The following represent the same color

```
<a:solidFill>
  <a:scrgbClr r="50000" g="50000" b="50000"/>
</a:solidFill>
<a:solidFill>
  <a:srgbClr val="BCBCBC"/>
</a:solidFill>
```

end example]

Parent Elements
accent1 (§5.1.4.1.1); accent2 (§5.1.4.1.2); accent3 (§5.1.4.1.3); accent4 (§5.1.4.1.4); accent5 (§5.1.4.1.5);

Parent Elements
accent6 (§5.1.4.1.6); alphaInv (§5.1.10.4); bgClr (§5.1.10.10); bgRef (§4.4.1.3); buClr (§5.1.5.4.4); clrFrom (§5.1.10.17); clrMru (§4.3.1.4); clrRepl (§5.1.10.18); clrTo (§5.1.10.19); clrVal (§4.6.27); contourClr (§5.1.7.6); custClr (§5.1.4.1.8); dk1 (§5.1.4.1.9); dk2 (§5.1.4.1.10); duotone (§5.1.10.23); effectClrLst (§5.9.4.7); effectRef (§5.1.4.2.8); extrusionClr (§5.1.7.7); fgClr (§5.1.10.27); fillClrLst (§5.9.4.8); fillRef (§5.1.4.2.10); folHlink (§5.1.4.1.15); fontRef (§5.1.4.1.17); from (§4.6.44); glow (§5.1.10.32); gs (§5.1.10.36); highlight (§5.1.5.3.4); hlink (§5.1.4.1.19); innerShdw (§5.1.10.40); linClrLst (§5.9.4.9); lnRef (§5.1.4.2.19); lt1 (§5.1.4.1.22); lt2 (§5.1.4.1.23); outerShdw (§5.1.10.45); penClr (§4.3.1.21); prstShdw (§5.1.10.49); solidFill (§5.1.10.54); tcTxStyle (§5.1.4.2.30); to (§4.6.90); txEffectClrLst (§5.9.4.12); txFillClrLst (§5.9.4.13); txLinClrLst (§5.9.4.14)

Child Elements	Subclause
alpha (Alpha)	§5.1.2.2.1
alphaMod (Alpha Modulation)	§5.1.2.2.2
alphaOff (Alpha Offset)	§5.1.2.2.3
blue (Blue)	§5.1.2.2.4
blueMod (Blue Modification)	§5.1.2.2.5
blueOff (Blue Offset)	§5.1.2.2.6
comp (Complement)	§5.1.2.2.7
gamma (Gamma)	§5.1.2.2.8
gray (Gray)	§5.1.2.2.9
green (Green)	§5.1.2.2.10
greenMod (Green Modification)	§5.1.2.2.11
greenOff (Green Offset)	§5.1.2.2.12
hue (Hue)	§5.1.2.2.14
hueMod (Hue Modulate)	§5.1.2.2.15
hueOff (Hue Offset)	§5.1.2.2.16
inv (Inverse)	§5.1.2.2.17
invGamma (Inverse Gamma)	§5.1.2.2.18
lum (Luminance)	§5.1.2.2.19
lumMod (Luminance Modulation)	§5.1.2.2.20
lumOff (Luminance Offset)	§5.1.2.2.21
red (Red)	§5.1.2.2.23
redMod (Red Modulation)	§5.1.2.2.24
redOff (Red Offset)	§5.1.2.2.25
sat (Saturation)	§5.1.2.2.26
satMod (Saturation Modulation)	§5.1.2.2.27

Child Elements	Subclause
satOff (Saturation Offset)	§5.1.2.2.28
shade (Shade)	§5.1.2.2.31
tint (Tint)	§5.1.2.2.34

Attributes	Description
b (Blue)	Specifies the percentage of blue. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
g (Green)	Specifies the percentage of green. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
r (Red)	Specifies the percentage of red. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ScRgbColor">
   <sequence>
      <group ref="EG_ColorTransform" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="r" type="ST_Percentage" use="required"/>
   <attribute name="g" type="ST_Percentage" use="required"/>
   <attribute name="b" type="ST_Percentage" use="required"/>
</complexType>
```

5.1.2.2.31 shade (Shade)

This element specifies a darker version of its input color. A 10% shade is 10% of the input color combined with 90% black.

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

Attributes	Description
val (Value)	Specifies the opacity as expressed by a percentage value. [*Example*: The following represents a green solid fill which is 50% opaque

Attributes	Description
	<a:solidFill> <a:srgbClr val="00FF00"> <a:alpha val="50000"/> </a:srgbClr> </a:solidFill> *end example*] The possible values for this attribute are defined by the ST_PositiveFixedPercentage simple type (§5.1.12.45).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PositiveFixedPercentage">
   <attribute name="val" type="ST_PositiveFixedPercentage" use="required"/>
</complexType>
```

5.1.2.2.32 srgbClr (RGB Color Model - Hex Variant)

This element specifies a color using the red, green, blue RGB color model. Red, green, and blue is expressed as sequence of hex digits, RRGGBB. A perceptual gamma of 2.2 is used.

Specifies the level of red as expressed by a percentage offset increase or decrease relative to the input color.

[*Example*: The following represent the same color

```
<a:solidFill>
  <a:scrgbClr r="50000" g="50000" b="50000"/>
</a:solidFill>
<a:solidFill>
  <a:srgbClr val="BCBCBC"/>
</a:solidFill>
```

end example]

Parent Elements
accent1 (§5.1.4.1.1); accent2 (§5.1.4.1.2); accent3 (§5.1.4.1.3); accent4 (§5.1.4.1.4); accent5 (§5.1.4.1.5); accent6 (§5.1.4.1.6); alphaInv (§5.1.10.4); bgClr (§5.1.10.10); bgRef (§4.4.1.3); buClr (§5.1.5.4.4); clrFrom (§5.1.10.17); clrMru (§4.3.1.4); clrRepl (§5.1.10.18); clrTo (§5.1.10.19); clrVal (§4.6.27); contourClr (§5.1.7.6); custClr (§5.1.4.1.8); dk1 (§5.1.4.1.9); dk2 (§5.1.4.1.10); duotone (§5.1.10.23); effectClrLst (§5.9.4.7); effectRef (§5.1.4.2.8); extrusionClr (§5.1.7.7); fgClr (§5.1.10.27); fillClrLst (§5.9.4.8); fillRef (§5.1.4.2.10); folHlink (§5.1.4.1.15); fontRef (§5.1.4.1.17); from (§4.6.44); glow (§5.1.10.32); gs (§5.1.10.36); highlight (§5.1.5.3.4); hlink (§5.1.4.1.19); innerShdw (§5.1.10.40); linClrLst (§5.9.4.9); lnRef (§5.1.4.2.19); lt1 (§5.1.4.1.22); lt2 (§5.1.4.1.23); outerShdw (§5.1.10.45); penClr (§4.3.1.21); prstShdw (§5.1.10.49); solidFill (§5.1.10.54); tcTxStyle (§5.1.4.2.30); to (§4.6.90); txEffectClrLst (§5.9.4.12); txFillClrLst (§5.9.4.13); txLinClrLst (§5.9.4.14)

Child Elements	Subclause
alpha (Alpha)	§5.1.2.2.1
alphaMod (Alpha Modulation)	§5.1.2.2.2
alphaOff (Alpha Offset)	§5.1.2.2.3
blue (Blue)	§5.1.2.2.4
blueMod (Blue Modification)	§5.1.2.2.5
blueOff (Blue Offset)	§5.1.2.2.6
comp (Complement)	§5.1.2.2.7
gamma (Gamma)	§5.1.2.2.8
gray (Gray)	§5.1.2.2.9
green (Green)	§5.1.2.2.10
greenMod (Green Modification)	§5.1.2.2.11
greenOff (Green Offset)	§5.1.2.2.12
hue (Hue)	§5.1.2.2.14
hueMod (Hue Modulate)	§5.1.2.2.15
hueOff (Hue Offset)	§5.1.2.2.16
inv (Inverse)	§5.1.2.2.17
invGamma (Inverse Gamma)	§5.1.2.2.18
lum (Luminance)	§5.1.2.2.19
lumMod (Luminance Modulation)	§5.1.2.2.20
lumOff (Luminance Offset)	§5.1.2.2.21
red (Red)	§5.1.2.2.23
redMod (Red Modulation)	§5.1.2.2.24
redOff (Red Offset)	§5.1.2.2.25
sat (Saturation)	§5.1.2.2.26
satMod (Saturation Modulation)	§5.1.2.2.27
satOff (Saturation Offset)	§5.1.2.2.28
shade (Shade)	§5.1.2.2.31
tint (Tint)	§5.1.2.2.34

Attributes	Description
val (Value)	The actual color value. Expressed as a sequence of hex digits RRGGBB. The possible values for this attribute are defined by the ST_HexBinary3 simple type (§5.1.12.28).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SRgbColor">
   <sequence>
      <group ref="EG_ColorTransform" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="val" type="ST_HexBinary3" use="required"/>
</complexType>
```

5.1.2.2.33 sysClr (System Color)

This element specifies a color bound to predefined operating system elements.

[*Example*: The following represents the default color used for displaying text in a window.

```
<a:solidFill>
  <a:sysClr val="windowText"/>
</a:solidFill>
```

end example]

Parent Elements
accent1 (§5.1.4.1.1); accent2 (§5.1.4.1.2); accent3 (§5.1.4.1.3); accent4 (§5.1.4.1.4); accent5 (§5.1.4.1.5); accent6 (§5.1.4.1.6); alphaInv (§5.1.10.4); bgClr (§5.1.10.10); bgRef (§4.4.1.3); buClr (§5.1.5.4.4); clrFrom (§5.1.10.17); clrMru (§4.3.1.4); clrRepl (§5.1.10.18); clrTo (§5.1.10.19); clrVal (§4.6.27); contourClr (§5.1.7.6); custClr (§5.1.4.1.8); dk1 (§5.1.4.1.9); dk2 (§5.1.4.1.10); duotone (§5.1.10.23); effectClrLst (§5.9.4.7); effectRef (§5.1.4.2.8); extrusionClr (§5.1.7.7); fgClr (§5.1.10.27); fillClrLst (§5.9.4.8); fillRef (§5.1.4.2.10); folHlink (§5.1.4.1.15); fontRef (§5.1.4.1.17); from (§4.6.44); glow (§5.1.10.32); gs (§5.1.10.36); highlight (§5.1.5.3.4); hlink (§5.1.4.1.19); innerShdw (§5.1.10.40); linClrLst (§5.9.4.9); lnRef (§5.1.4.2.19); lt1 (§5.1.4.1.22); lt2 (§5.1.4.1.23); outerShdw (§5.1.10.45); penClr (§4.3.1.21); prstShdw (§5.1.10.49); solidFill (§5.1.10.54); tcTxStyle (§5.1.4.2.30); to (§4.6.90); txEffectClrLst (§5.9.4.12); txFillClrLst (§5.9.4.13); txLinClrLst (§5.9.4.14)

Child Elements	Subclause
alpha (Alpha)	§5.1.2.2.1
alphaMod (Alpha Modulation)	§5.1.2.2.2
alphaOff (Alpha Offset)	§5.1.2.2.3
blue (Blue)	§5.1.2.2.4
blueMod (Blue Modification)	§5.1.2.2.5
blueOff (Blue Offset)	§5.1.2.2.6
comp (Complement)	§5.1.2.2.7
gamma (Gamma)	§5.1.2.2.8
gray (Gray)	§5.1.2.2.9
green (Green)	§5.1.2.2.10
greenMod (Green Modification)	§5.1.2.2.11

Child Elements	Subclause
greenOff (Green Offset)	§5.1.2.2.12
hue (Hue)	§5.1.2.2.14
hueMod (Hue Modulate)	§5.1.2.2.15
hueOff (Hue Offset)	§5.1.2.2.16
inv (Inverse)	§5.1.2.2.17
invGamma (Inverse Gamma)	§5.1.2.2.18
lum (Luminance)	§5.1.2.2.19
lumMod (Luminance Modulation)	§5.1.2.2.20
lumOff (Luminance Offset)	§5.1.2.2.21
red (Red)	§5.1.2.2.23
redMod (Red Modulation)	§5.1.2.2.24
redOff (Red Offset)	§5.1.2.2.25
sat (Saturation)	§5.1.2.2.26
satMod (Saturation Modulation)	§5.1.2.2.27
satOff (Saturation Offset)	§5.1.2.2.28
shade (Shade)	§5.1.2.2.31
tint (Tint)	§5.1.2.2.34

Attributes	Description
lastClr (Last Color)	Specifies the color value that was last computed by the generating application. The possible values for this attribute are defined by the ST_HexBinary3 simple type (§5.1.12.28).
val (Value)	Specifies the system color value. The possible values for this attribute are defined by the ST_SystemColorVal simple type (§5.1.12.58).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SystemColor">
   <sequence>
      <group ref="EG_ColorTransform" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="val" type="ST_SystemColorVal" use="required"/>
   <attribute name="lastClr" type="ST_HexBinary3" use="optional"/>
</complexType>
```

5.1.2.2.34 tint (Tint)

This element specifies a lighter version of its input color. A 10% tint is 10% of the input color combined with 90% white.

Parent Elements
hslClr (§5.1.2.2.13); prstClr (§5.1.2.2.22); schemeClr (§5.1.2.2.29); scrgbClr (§5.1.2.2.30); srgbClr (§5.1.2.2.32); sysClr (§5.1.2.2.33)

Attributes	Description

val (Value)	Specifies the opacity as expressed by a percentage value.
	
	[*Example*: The following represents a green solid fill which is 50% opaque
	
	`<a:solidFill>`
	`  <a:srgbClr val="00FF00">`
	`  <a:alpha val="50000"/>`
	`  </a:srgbClr>`
	`</a:solidFill>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_PositiveFixedPercentage simple type (§5.1.12.45).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PositiveFixedPercentage">
   <attribute name="val" type="ST_PositiveFixedPercentage" use="required"/>
</complexType>
```

5.1.3 Audio and Video

The Audio and Video portion of the DrawingML framework deals with all media of these two types that can be attached to objects within a document. Types of audio that can be represented within a file are CD audio, QuickTime audio, and any other generic audio. When dealing with generic audio there is the option for embedding it within the file and also linking it. The linking option is preferable if the size of the audio file is too large and will thus increase the size of the document by an undesirable amount. For video there are two types that can be represented and that is either a QuickTime movie or any other generic movie. When dealing with generic video there is only the option of linking to the media as video is too large to embed within a document.

5.1.3.1 audioCd (Audio from CD)

This element specifies the existence of Audio from a CD. This element is specified within the non-visual properties of an object. The audio must be attached to an object as this is how it is represented within the document. The actual playing of the sound however is done within the timing node list that is specified under the timing element.

[*Example*: Consider the following picture object that has an audio from a CD attached to it.

```
<p:pic>
  <p:nvPicPr>
    <p:cNvPr id="7" name="Rectangle 6">
      <a:hlinkClick r:id="" action="ppaction://media"/>
    </p:cNvPr>
    <p:cNvPicPr>
      <a:picLocks noRot="1"/>
    </p:cNvPicPr>
    <p:nvPr>
      <a:audioCd>
        <a:st track="1"/>
        <a:end track="3" time="65"/>
      </a:audioCd>
    </p:nvPr>
  </p:nvPicPr>
  …
</p:pic>
```

In the above example, we see that there is a single audioCD element attached to this picture. This picture is placed within the document just as a normal picture or shape would be. The id of this picture, namely 7 in this case, will be used to refer to this audioCD element from within the timing node list. For this example we see that the audio for this CD will start playing at the 0 second mark on the first track and will end on the 1 minute 5 second mark of the third track. *end example*]

Parent Elements
nvPr (§4.4.1.30)

Child Elements	Subclause
end (Audio End Time)	§5.1.3.3
extLst (Extension List)	§5.1.2.1.15
st (Audio Start Time)	§5.1.3.5

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AudioCD">
   <sequence>
      <element name="st" type="CT_AudioCDTime" minOccurs="1" maxOccurs="1"/>
      <element name="end" type="CT_AudioCDTime" minOccurs="1" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.3.2 audioFile (Audio from File)

This element specifies the existence of an audio file. This element is specified within the non-visual properties of an object. The audio must be attached to an object as this is how it is represented within the document. The actual playing of the audio however is done within the timing node list that is specified under the timing element.

[*Example*: Consider the following picture object that has an audio file attached to it.

```
<p:pic>
  <p:nvPicPr>
    <p:cNvPr id="7" name="Rectangle 6">
      <a:hlinkClick r:id="" action="ppaction://media"/>
    </p:cNvPr>
    <p:cNvPicPr>
      <a:picLocks noRot="1"/>
    </p:cNvPicPr>
    <p:nvPr>
      <a:audioFile r:link="rId1"/>
    </p:nvPr>
  </p:nvPicPr>
  …
</p:pic>
```

In the above example, we see that there is a single audioFile element attached to this picture. This picture is placed within the document just as a normal picture or shape would be. The id of this picture, namely 7 in this case, will be used to refer to this audioFile element from within the timing node list. The Linked relationship id will be used to retrieve the actual audio file for playback purposes. *end example*]

Parent Elements
nvPr (§4.4.1.30)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15

Attributes	Description
link (Linked Relationship ID) Namespace: …/officeDocument/2006/relationships	Specifies the identification information for a linked object. This attribute is used to specify the location of an object that does not reside within this file. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AudioFile">
   <sequence>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute ref="r:link" use="required"/>
</complexType>
```

5.1.3.3 end (Audio End Time)

This element specifies the end point for a CD Audio sound element. Encompassed within this element are the time and track at which the sound should halt its playback. This element will be used in conjunction with an Audio Start Time element to specify the time span for an entire audioCD sound element.

[*Example*: Consider the following DrawingML.

```
<a:audioCd>
  <a:st track="1" time="2"/>
  <a:end track="3" time="65"/>
</a:audioCd>
```

In the above example, the audioCD sound element shown will specify for a portion of audio spanning from 2 seconds into the first track to 1 minute, 5 seconds into the third track. *end example*]

Parent Elements
audioCd (§5.1.3.1)

Attributes	Description
time (Time)	Specifies the time in seconds that the CD Audio should be started at. If this attribute is omitted, then a value of 0 is assumed. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
track (Track)	Specifies which track of the CD this Audio will begin playing on. This attribute is required and cannot be omitted. The possible values for this attribute are defined by the XML Schema unsignedByte datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AudioCDTime">
   <attribute name="track" type="xsd:unsignedByte" use="required"/>
   <attribute name="time" type="xsd:unsignedInt" use="optional" default="0"/>
</complexType>
```

5.1.3.4 quickTimeFile (QuickTime from File)

This element specifies the existence of a QuickTime file. This element is specified within the non-visual properties of an object. The QuickTime file must be attached to an object as this is how it is represented within the document. The actual playing of the QuickTime however is done within the timing node list that is specified under the timing element.

[*Example*: Consider the following picture object that has a QuickTime file attached to it.

```
<p:pic>
  <p:nvPicPr>
    <p:cNvPr id="7" name="Rectangle 6">
      <a:hlinkClick r:id="" action="ppaction://media"/>
    </p:cNvPr>
    <p:cNvPicPr>
      <a:picLocks noRot="1"/>
    </p:cNvPicPr>
    <p:nvPr>
      <a:quickTimeFile r:link="rId1"/>
    </p:nvPr>
  </p:nvPicPr>
  …
</p:pic>
```

In the above example, we see that there is a single quickTimeFile element attached to this picture. This picture is placed within the document just as a normal picture or shape would be. The id of this picture, namely 7 in this case, will be used to refer to this quickTimeFile element from within the timing node list. The Linked relationship id will be used to retrieve the actual video file for playback purposes. *end example*]

Parent Elements
nvPr (§4.4.1.30)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15

Attributes	Description
link (Linked Relationship ID) Namespace: …/officeDocument/2006/relationships	Specifies the identification information for a linked object. This attribute is used to specify the location of an object that does not reside within this file. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_QuickTimeFile">
   <sequence>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute ref="r:link" use="required"/>
</complexType>
```

5.1.3.5 st (Audio Start Time)

This element specifies the start point for a CD Audio sound element. Encompassed within this element are the time and track at which the sound should begin its playback. This element will be used in conjunction with an Audio End Time element to specify the time span for an entire audioCD sound element.

[*Example*: Consider the following DrawingML.

```
<a:audioCd>
  <a:st track="1" time="2"/>
  <a:end track="3" time="65"/>
</a:audioCd>
```

In the above example, the audioCD sound element shown will specify for a portion of audio spanning from 2 seconds into the first track to 1 minute, 5 seconds into the third track. *end example*]

Parent Elements
audioCd (§5.1.3.1)

Attributes	Description
time (Time)	Specifies the time in seconds that the CD Audio should be started at. If this attribute is omitted, then a value of 0 is assumed. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
track (Track)	Specifies which track of the CD this Audio will begin playing on. This attribute is required and cannot be omitted. The possible values for this attribute are defined by the XML Schema unsignedByte datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AudioCDTime">
   <attribute name="track" type="xsd:unsignedByte" use="required"/>
   <attribute name="time" type="xsd:unsignedInt" use="optional" default="0"/>
</complexType>
```

5.1.3.6 videoFile (Video from File)

This element specifies the existence of a video file. This element is specified within the non-visual properties of an object. The video must be attached to an object as this is how it is represented within the document. The actual playing of the video however is done within the timing node list that is specified under the timing element.

[*Example*: Consider the following picture object that has a video attached to it.

```
<p:pic>
  <p:nvPicPr>
    <p:cNvPr id="7" name="Rectangle 6">
      <a:hlinkClick r:id="" action="ppaction://media"/>
    </p:cNvPr>
    <p:cNvPicPr>
      <a:picLocks noRot="1"/>
    </p:cNvPicPr>
    <p:nvPr>
      <a:videoFile r:link="rId1"/>
    </p:nvPr>
  </p:nvPicPr>
  …
</p:pic>
```

In the above example, we see that there is a single videoFile element attached to this picture. This picture is placed within the document just as a normal picture or shape would be. The id of this picture, namely 7 in this case, will be used to refer to this videoFile element from within the timing node list. The Linked relationship id will be used to retrieve the actual video file for playback purposes. *end example*]

Parent Elements
nvPr (§4.4.1.30)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15

Attributes	Description
link (Linked Relationship ID) Namespace: …/officeDocument/2006/relationships	Specifies the identification information for a linked video file. This attribute is used to specify the location of an object that does not reside within this file. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_VideoFile">
   <sequence>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute ref="r:link" use="required"/>
</complexType>
```

5.1.3.7 wavAudioFile (Audio from WAV File)

This element specifies the existence of an audio WAV file. This element is specified within the non-visual properties of an object. The audio must be attached to an object as this is how it is represented within the document. The actual playing of the audio however is done within the timing node list that is specified under the timing element.

[*Example*: Consider the following picture object that has an audio WAV file attached to it.

```
<p:pic>
  <p:nvPicPr>
    <p:cNvPr id="7" name="Rectangle 6">
      <a:hlinkClick r:id="" action="ppaction://media"/>
    </p:cNvPr>
    <p:cNvPicPr>
      <a:picLocks noRot="1"/>
    </p:cNvPicPr>
    <p:nvPr>
      <a:wavAudioFile r:embed="rId2"/>
    </p:nvPr>
  </p:nvPicPr>
  …
</p:pic>
```

In the above example, we see that there is a single wavAudioFile element attached to this picture. This picture is placed within the document just as a normal picture or shape would be. The id of this picture, namely 7 in this case, will be used to refer to this wavAudioFile element from within the timing node list. The Embedded relationship id will be used to retrieve the actual audio file for playback purposes. *end example*]

[*Note*: This element is generally used for the purposes of embedding audio files within the document. For linking to generic audio files the audioFile element should be used. *end note*]

Parent Elements
nvPr (§4.4.1.30)

Attributes	Description
builtIn (Recognized	Specifies whether or not this sound is a built-in sound. If this attribute is set to true then

Attributes	Description
Built-In Sound)	the generating application is alerted to check the name attribute specified for this sound in it's list of built-in sounds and can then surface a custom name or UI as needed. The possible values for this attribute are defined by the XML Schema boolean datatype.
embed (Embedded Audio File Relationship ID) Namespace: .../officeDocument/2006/relationships	Specifies the identification information for an embedded audio file. This attribute is used to specify the location of an object that resides locally within the file. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
name (Sound Name)	Specifies the original name or given short name for the corresponding sound. This is used to distinguish this sound from others by providing a human readable name for the attached sound should the user need to identify the sound among others within the UI. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EmbeddedWAVAudioFile">
   <attribute ref="r:embed" use="required"/>
   <attribute name="name" type="xsd:string" use="optional" default=""/>
   <attribute name="builtIn" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.1.4 Styles

Styles within DrawingML refer to the way a particular object (be it text or a shape, or anything else) is formatted. Different aspects, ranging from color, line type, fill, and effects applied to the object can be predefined within a theme. The main purpose of a theme is to define a style matrix from which a document can pull style information from in order to format the visual look of objects in a document.

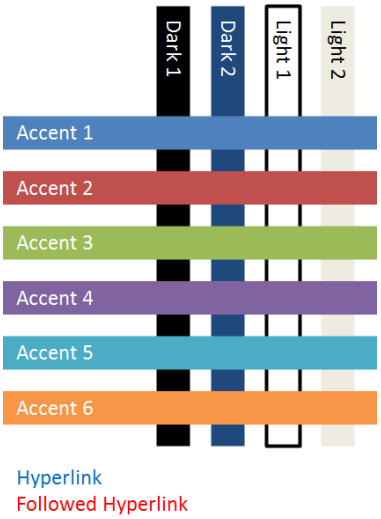
5.1.4.1 Styles

The elements in this section compose the basic definition of a style, including its associated colors, effect styles, line styles, fill styles, background styles, and font scheme.

5.1.4.1.1 accent1 (Accent 1)

This element defines a color that happens to be the accent 1 color. The set of twelve colors come together to form the color scheme for a theme.

[*Example*: Consider the following example of a set of colors that form a color scheme:



end example]

Parent Elements
clrScheme (§5.1.8.2)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

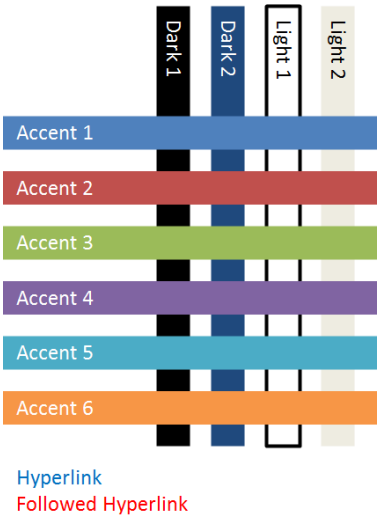
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

5.1.4.1.2 accent2 (Accent 2)

This element defines a color that happens to be the accent 2 color. The set of twelve colors come together to form the color scheme for a theme.

[*Example*: Consider the following example of a set of colors that form a color scheme:



end example]

Parent Elements
clrScheme (§5.1.8.2)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

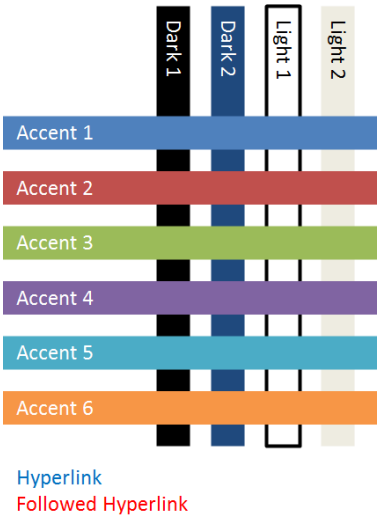
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

5.1.4.1.3 accent3 (Accent 3)

This element defines a color that happens to be the accent 3 color. The set of twelve colors come together to form the color scheme for a theme.

[*Example*: Consider the following example of a set of colors that form a color scheme:



end example]

Parent Elements
clrScheme (§5.1.8.2)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

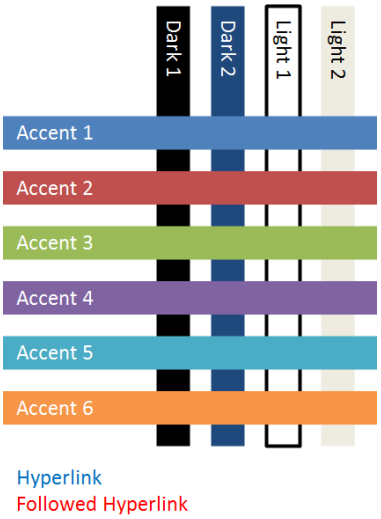
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

5.1.4.1.4 accent4 (Accent 4)

This element defines a color that happens to be the accent 4 color. The set of twelve colors come together to form the color scheme for a theme.

[*Example*: Consider the following example of a set of colors that form a color scheme:



end example]

Parent Elements
clrScheme (§5.1.8.2)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

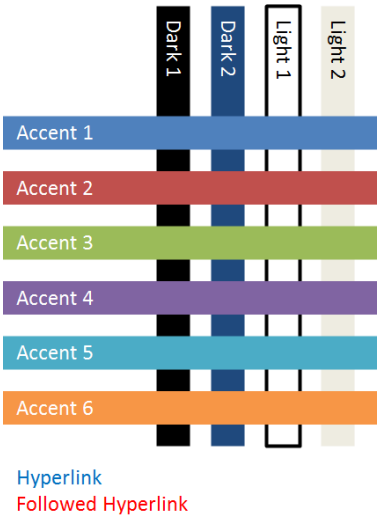
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

5.1.4.1.5 accent5 (Accent 5)

This element defines a color that happens to be the accent 5 color. The set of twelve colors come together to form the color scheme for a theme.

[*Example*: Consider the following example of a set of colors that form a color scheme:



end example]

Parent Elements
clrScheme (§5.1.8.2)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

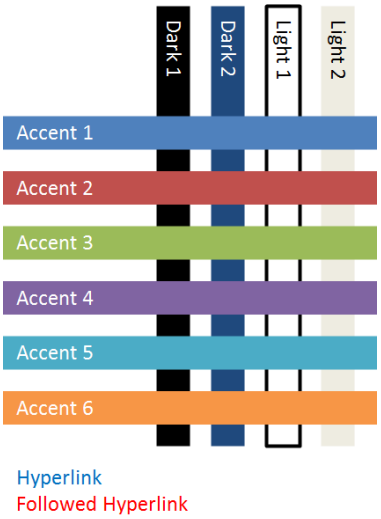
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

5.1.4.1.6 accent6 (Accent 6)

This element defines a color that happens to be the accent 1 color. The set of twelve colors come together to form the color scheme for a theme.

[*Example*: Consider the following example of a set of colors that form a color scheme:



end example]

Parent Elements
clrScheme (§5.1.8.2)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

5.1.4.1.7 bgFillStyleLst (Background Fill Style List)

This element defines a list of background fills that are used within a theme. The background fills consist of three fills, arranged in order from subtle to moderate to intense.

[*Example*: Consider the following example of a background fill style list within DrawingML:

```
<bgFillStyleLst>
  <solidFill>
…
  </solidFill>
  <gradFill rotWithShape="1">
…
  </gradFill>
  <blipFill>
…
  </blipFill>
</bgFillStyleLst>
```

In this example, we see that the list contains a solid fill for the subtle fill, a gradient fill for the moderate fill and an image fill for the intense background fill. *end example*]

Parent Elements
fmtScheme (§5.1.4.1.14)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
solidFill (Solid Fill)	§5.1.10.54

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BackgroundFillStyleList">
   <sequence>
      <group ref="EG_FillProperties" minOccurs="3" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.1.4.1.8 custClr (Custom color)

This element defines a custom color. The custom colors are used within a custom color list to define custom colors that are extra colors that can be appended to a theme. This is useful within corporate scenarios where there is a set corporate color palette from which to work.

Parent Elements
custClrLst (§5.1.8.3)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

Attributes	Description
name (Name)	The name of the color shown in the color picker. The possible values for this attribute are defined by the XML Schema string datatype.

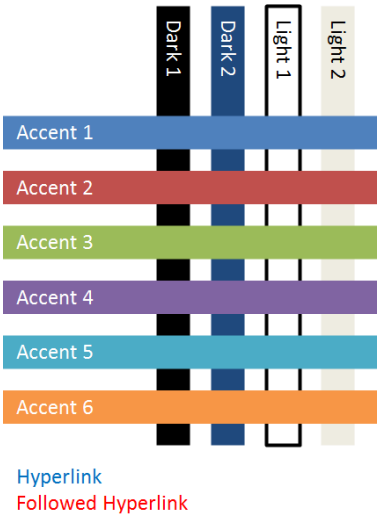
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustomColor">
   <sequence>
      <group ref="EG_ColorChoice" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="name" type="xsd:string" use="optional" default=""/>
</complexType>
```

5.1.4.1.9 dk1 (Dark 1)

This element defines a color that happens to be the dark 1 color. The set of twelve colors come together to form the color scheme for a theme.

[*Example*: Consider the following example of a set of colors that form a color scheme:



end example]

Parent Elements
clrScheme (§5.1.8.2)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

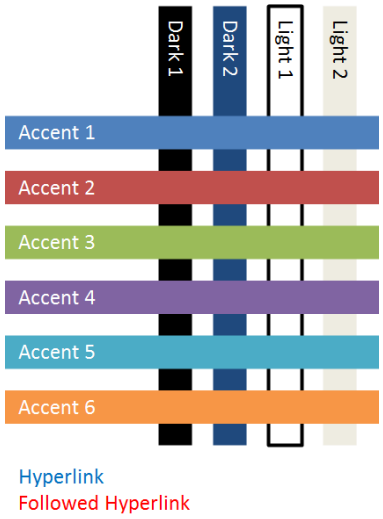
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

5.1.4.1.10 dk2 (Dark 2)

This element defines a color that happens to be the dark 2 color. The set of twelve colors come together to form the color scheme for a theme.

[*Example*: Consider the following example of a set of colors that form a color scheme:



end example]

Parent Elements
clrScheme (§5.1.8.2)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

5.1.4.1.11 effectStyle (Effect Style)

This element defines a set of effects and 3D properties that can be applied to an object.

[*Example*: Consider the following example of an effect style within DrawingML:

```
<effectStyle>
  <effectLst>
    <outerShdw blurRad="57150" dist="38100" dir="5400000" algn="ctr"
      rotWithShape="0">
      <schemeClr val="phClr">
        <shade val="9000"/>
        <satMod val="105000"/>
        <alpha val="48000"/>
      </schemeClr>
    </outerShdw>
  </effectLst>
</effectStyle>
```

In this example, an outer shadow is being applied to a shape as the moderate effect. *end example*]

Parent Elements

Parent Elements
effectStyleLst (§5.1.4.1.12)

Child Elements	Subclause
effectDag (Effect Container)	§5.1.10.25
effectLst (Effect Container)	§5.1.10.26
scene3d (3D Scene Properties)	§5.1.4.1.26
sp3d (Apply 3D shape properties)	§5.1.7.12

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EffectStyleItem">
   <sequence>
      <group ref="EG_EffectProperties" minOccurs="1" maxOccurs="1"/>
      <element name="scene3d" type="CT_Scene3D" minOccurs="0" maxOccurs="1"/>
      <element name="sp3d" type="CT_Shape3D" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.1.12 effectStyleLst (Effect Style List)

This element defines a set of three effect styles that create the effect style list for a theme. The effect styles are arranged in order of subtle to moderate to intense.

[*Example*: Consider the following example of an effect style list within DrawingML:

```
<effectStyleLst>
  <effectStyle>
    <effectLst>
      <outerShdw blurRad="57150" dist="38100" dir="5400000"
        algn="ctr" rotWithShape="0">
…
      </outerShdw>
    </effectLst>
  </effectStyle>
  <effectStyle>
    <effectLst>
      <outerShdw blurRad="57150" dist="38100" dir="5400000"
        algn="ctr" rotWithShape="0">
…
      </outerShdw>
    </effectLst>
  </effectStyle>
```

```
    <effectStyle>
      <effectLst>
        <outerShdw blurRad="57150" dist="38100" dir="5400000"
          algn="ctr" rotWithShape="0">
…
        </outerShdw>
      </effectLst>
      <scene3d>
…
      </scene3d>
      <sp3d prstMaterial="powder">
…
      </sp3d>
    </effectStyle>
  </effectStyleLst>
```

In this example, we see three effect styles defined. The first two (subtle and moderate) define an outer shadow as the effect, while the third effect style (intense) defines an outer shadow along with 3D properties which are to be applied to the object as well. *end example*]

Parent Elements
fmtScheme (§5.1.4.1.14)

Child Elements	Subclause
effectStyle (Effect Style)	§5.1.4.1.11

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EffectStyleList">
   <sequence>
      <element name="effectStyle" type="CT_EffectStyleItem" minOccurs="3" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.1.4.1.13 fillStyleLst (Fill Style List)

This element defines a set of three fill styles that are used within a theme. The three fill styles are arranged in order from subtle to moderate to intense.

[*Example*: Consider the following example of a fill style list within DrawingML:

```
  <fillStyleLst>
    <solidFill>
…
    </solidFill>
```

```
    <gradFill rotWithShape="1">
  …
    </gradFill>
    <gradFill rotWithShape="1">
  …
    </gradFill>
  </fillStyleLst>
```

In this example, we see three fill styles being defined within the fill style list. The first style is the subtle style and defines simply a solid fill. The second and third styles (moderate and intense fills respectively) define gradient fills. *end example*]

Parent Elements
fmtScheme (§5.1.4.1.14)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
solidFill (Solid Fill)	§5.1.10.54

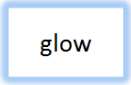
The following XML Schema fragment defines the contents of this element:

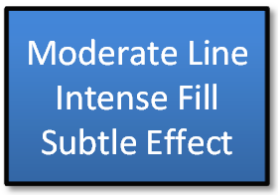
```
<complexType name="CT_FillStyleList">
   <sequence>
      <group ref="EG_FillProperties" minOccurs="3" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.1.4.1.14 fmtScheme (Format Scheme)

This element contains the background fill styles, effect styles, fill styles, and line styles which define the style matrix for a theme. The style matrix consists of subtle, moderate, and intense fills, lines, and effects. The background fills are not generally thought of to directly be associated with the matrix, but do play a role in the style of the overall document. Usually, a given object will choose a single line style, a single fill style, and a single effect style in order to define the overall final look of the object.

[*Example*: Consider the following example of the style matrix in use within DrawingML:

	Line	Fill	Effect
Subtle			
Moderate			
Intense			



In this example, we see a shape styled which utilizes different aspects from the above defined style matrix. *end example*]

Parent Elements
themeElements (§5.1.8.10); themeOverride (§5.1.8.12)

Child Elements	Subclause
bgFillStyleLst (Background Fill Style List)	§5.1.4.1.7
effectStyleLst (Effect Style List)	§5.1.4.1.12
fillStyleLst (Fill Style List)	§5.1.4.1.13
lnStyleLst (Line Style List)	§5.1.4.1.21

Attributes	Description
name (Name)	Defines the name for the format scheme. The name is simply a human readable string which identifies the format scheme in the user interface. The possible values for this attribute are defined by the XML Schema string datatype.

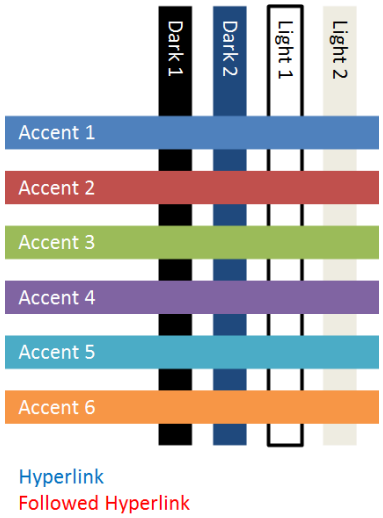
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_StyleMatrix">
    <sequence>
        <element name="fillStyleLst" type="CT_FillStyleList" minOccurs="1" maxOccurs="1"/>
        <element name="lnStyleLst" type="CT_LineStyleList" minOccurs="1" maxOccurs="1"/>
        <element name="effectStyleLst" type="CT_EffectStyleList" minOccurs="1" maxOccurs="1"/>
        <element name="bgFillStyleLst" type="CT_BackgroundFillStyleList" minOccurs="1" maxOccurs="1"/>
    </sequence>
    <attribute name="name" type="xsd:string" use="optional" default=""/>
</complexType>
```

5.1.4.1.15 folHlink (Followed Hyperlink)

This element defines a color that happens to be the followed hyperlink color. The set of twelve colors come together to form the color scheme for a theme.

[*Example*: Consider the following example of a set of colors that form a color scheme:



end example]

Parent Elements
clrScheme (§5.1.8.2)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

5.1.4.1.16 font (Font)

This element defines a font within the styles area of DrawingML. A font is defined by a script along with a typeface.

[*Example*: Consider the following example of a font in DrawingML:

```
<font script="Thai" typeface="Cordia New"/>
```

In this example, we see that the script 'Thai' is supposed to use the font face 'Cordia New'. *end example*]

Parent Elements
font (§5.1.4.2.13); majorFont (§5.1.4.1.24); minorFont (§5.1.4.1.25)

Attributes	Description
script (Script)	Specifies the script, or language, in which the typeface is supposed to be used. The possible values for this attribute are defined by the XML Schema string datatype.
typeface (Typeface)	Specifies the font face to use. The possible values for this attribute are defined by the ST_TextTypeface simple type (§5.1.12.81).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SupplementalFont">
   <attribute name="script" type="xsd:string" use="required"/>
   <attribute name="typeface" type="ST_TextTypeface" use="required"/>
</complexType>
```

5.1.4.1.17 fontRef (Font Reference)

This element represents a reference to a themed font. When used it specifies which themed font to use along with a choice of color.

[*Example*: Consider the following example of a font reference within DrawingML:

```
<fontRef idx="minor">
  <schemeClr val="tx1"/>
</fontRef>
```

In this example, we see a font referencing the minor font defined within the theme. *end example*]

Parent Elements
style (§5.6.2.30); style (§5.9.2.28); style (§4.4.1.43); style (§5.8.2.24); style (§5.1.2.1.37); tcTxStyle (§5.1.4.2.30)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

Attributes	Description
idx (Identifier)	Specifies the identifier of the font to reference. The possible values for this attribute are defined by the ST_FontCollectionIndex simple type (§5.1.12.23).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FontReference">
   <sequence>
      <group ref="EG_ColorChoice" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="idx" type="ST_FontCollectionIndex" use="required"/>
</complexType>
```

5.1.4.1.18 fontScheme (Font Scheme)

This element defines the font scheme within the theme. The font scheme consists of a pair of major and minor fonts for which to use in a document. The major font corresponds well with the heading areas of a document, and the minor font corresponds well with the normal text or paragraph areas.

[*Example*: Consider the following example of a font scheme within DrawingML:

```
<fontScheme name="sample">
  <majorFont>
…
  </majorFont>
```

```
    <minorFont>
  …
    </minorFont>
  </fontScheme>
```

In this example, we see the major and minor font lists within the font scheme that is named 'sample'. *end example*]

Parent Elements
themeElements (§5.1.8.10); themeOverride (§5.1.8.12)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
majorFont (Major Font)	§5.1.4.1.24
minorFont (Minor fonts)	§5.1.4.1.25

Attributes	Description
name (Name)	The name of the font scheme shown in the user interface. The possible values for this attribute are defined by the XML Schema string datatype.

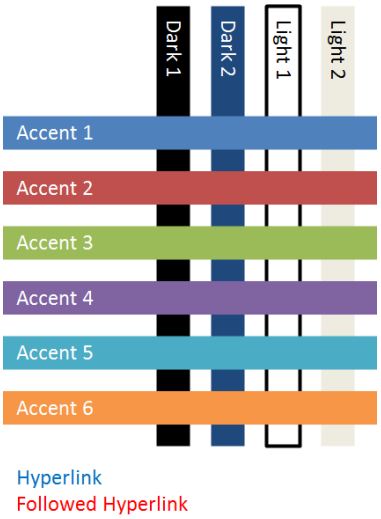
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FontScheme">
   <sequence>
      <element name="majorFont" type="CT_FontCollection" minOccurs="1" maxOccurs="1"/>
      <element name="minorFont" type="CT_FontCollection" minOccurs="1" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="name" type="xsd:string" use="required"/>
</complexType>
```

5.1.4.1.19 hlink (Hyperlink)

This element defines a color that happens to be the hyperlink color. The set of twelve colors come together to form the color scheme for a theme.

[*Example*: Consider the following example of a set of colors that form a color scheme:



end example]

Parent Elements
clrScheme (§5.1.8.2)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

5.1.4.1.20 lnDef (Line Default)

This element defines a default line that is used within a document.

[*Example*: Consider the following example of a default line defined in DrawingML:

```
<lnDef>
  <spPr/>
  <bodyPr/>
  <lstStyle/>
  <style>
    <lnRef idx="1">
      <schemeClr val="accent2"/>
    </lnRef>
    <fillRef idx="0">
      <schemeClr val="accent2"/>
    </fillRef>
    <effectRef idx="0">
      <schemeClr val="accent2"/>
    </effectRef>
    <fontRef idx="minor">
      <schemeClr val="tx1"/>
    </fontRef>
  </style>
</lnDef>
```

In this example, we see that the default line for the document is being defined as a themed line which references the subtle line style with idx equal to 1. *end example*]

Parent Elements
objectDefaults (§5.1.8.7)

Child Elements	Subclause
bodyPr (Body Properties)	§5.1.5.1.1
extLst (Extension List)	§5.1.2.1.15
lstStyle (Text List Styles)	§5.1.5.4.12
spPr (Shape Properties)	§5.1.2.1.35
style (Shape Style)	§5.1.2.1.37

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DefaultShapeDefinition">
   <sequence>
      <element name="spPr" type="CT_ShapeProperties" minOccurs="1" maxOccurs="1"/>
      <element name="bodyPr" type="CT_TextBodyProperties" minOccurs="1" maxOccurs="1"/>
      <element name="lstStyle" type="CT_TextListStyle" minOccurs="1" maxOccurs="1"/>
      <element name="style" type="CT_ShapeStyle" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.1.21 lnStyleLst (Line Style List)

This element defines a list of three line styles for use within a theme. The three line styles are arranged in order from subtle to moderate to intense versions of lines. This list makes up part of the style matrix.

[*Example*: Consider the following example of a line style list within DrawingML:

```
<lnStyleLst>
  <ln w="9525" cap="flat" cmpd="sng" algn="ctr">
    <solidFill>
      <schemeClr val="phClr">
        <shade val="50000"/>
        <satMod val="103000"/>
      </schemeClr>
    </solidFill>
    <prstDash val="solid"/>
  </ln>
  <ln w="25400" cap="flat" cmpd="sng" algn="ctr">
    <solidFill>
      <schemeClr val="phClr"/>
    </solidFill>
    <prstDash val="solid"/>
  </ln>
  <ln w="38100" cap="flat" cmpd="sng" algn="ctr">
    <solidFill>
      <schemeClr val="phClr"/>
    </solidFill>
    <prstDash val="solid"/>
  </ln>
</lnStyleLst>
```

In this example, we see three lines defined within a line style list. The first line corresponds to the subtle line, the second to the moderate, and the third corresponds to the intense line defined in the theme. *end example*]

Parent Elements
fmtScheme (§5.1.4.1.14)

Child Elements	Subclause
ln (Outline)	§5.1.2.1.24

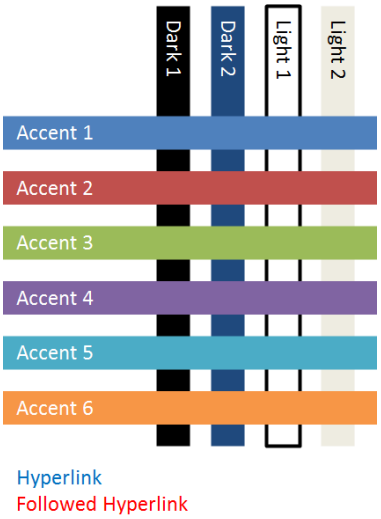
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LineStyleList">
    <sequence>
        <element name="ln" type="CT_LineProperties" minOccurs="3" maxOccurs="unbounded"/>
    </sequence>
</complexType>
```

5.1.4.1.22 lt1 (Light 1)

This element defines a color that happens to be the accent 1 color. The set of twelve colors come together to form the color scheme for a theme.

[*Example*: Consider the following example of a set of colors that form a color scheme:



end example]

Parent Elements
clrScheme (§5.1.8.2)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

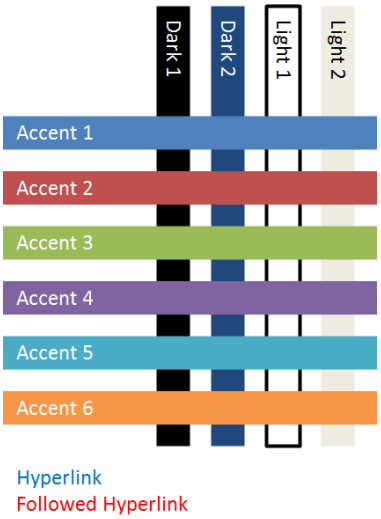
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

5.1.4.1.23 lt2 (Light 2)

This element defines a color that happens to be the accent 1 color. The set of twelve colors come together to form the color scheme for a theme.

[*Example*: Consider the following example of a set of colors that form a color scheme:



end example]

Parent Elements
clrScheme (§5.1.8.2)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

5.1.4.1.24 majorFont (Major Font)

This element defines the set of major fonts which are to be used under different languages or locals.

[*Example*: Consider the following example of the major fonts being defined within DrawingML:

```
<majorFont>
  <latin typeface="Calibri"/>
  <ea typeface="Arial"/>
  <cs typeface="Arial"/>
  <font script="Jpan" typeface="ＭＳ Ｐゴシック"/>
  <font script="Hang" typeface="HY중고딕"/>
  <font script="Hans" typeface="隶书"/>
  <font script="Hant" typeface="微軟正黑體"/>
  <font script="Arab" typeface="Traditional Arabic"/>
  <font script="Hebr" typeface="Arial"/>
  <font script="Thai" typeface="Cordia New"/>
  <font script="Ethi" typeface="Nyala"/>
  <font script="Beng" typeface="Vrinda"/>
  <font script="Gujr" typeface="Shruti"/>
  <font script="Khmr" typeface="DaunPenh"/>
  <font script="Knda" typeface="Tunga"/>
</majorFont>
```

In this example, we see the latin, east asian, and complex script fonts defined along with many fonts for different locals. *end example*]

Parent Elements
fontScheme (§5.1.4.1.18)

Child Elements	Subclause
cs (Complex Script Font)	§5.1.5.3.1
ea (East Asian Font)	§5.1.5.3.3
extLst (Extension List)	§5.1.2.1.15
font (Font)	§5.1.4.1.16

Child Elements	Subclause
latin (Latin Font)	§5.1.5.3.7

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FontCollection">
   <sequence>
      <element name="latin" type="CT_TextFont" minOccurs="1" maxOccurs="1"/>
      <element name="ea" type="CT_TextFont" minOccurs="1" maxOccurs="1"/>
      <element name="cs" type="CT_TextFont" minOccurs="1" maxOccurs="1"/>
      <element name="font" type="CT_SupplementalFont" minOccurs="0" maxOccurs="unbounded"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.1.25 minorFont (Minor fonts)

This element defines the set of minor fonts that are to be used under different languages or locals.

[*Example*: Consider the following example of the minor fonts being defined within DrawingML:

```
<minorFont>
  <latin typeface="Calibri"/>
  <ea typeface="Arial"/>
  <cs typeface="Arial"/>
  <font script="Jpan" typeface="ＭＳ Ｐゴシック"/>
  <font script="Hang" typeface="HY중고딕"/>
  <font script="Hans" typeface="隶书"/>
  <font script="Hant" typeface="微軟正黑體/>
  <font script="Arab" typeface="Traditional Arabic"/>
  <font script="Hebr" typeface="Arial"/>
  <font script="Thai" typeface="Cordia New"/>
  <font script="Ethi" typeface="Nyala"/>
  <font script="Beng" typeface="Vrinda"/>
  <font script="Gujr" typeface="Shruti"/>
  <font script="Khmr" typeface="DaunPenh"/>
  <font script="Knda" typeface="Tunga"/>
</minorFont>
```

In this example, we see the latin, east asian, and complex script fonts defined along with many fonts for different locals. *end example*]

Parent Elements
fontScheme (§5.1.4.1.18)

Child Elements	Subclause
cs (Complex Script Font)	§5.1.5.3.1
ea (East Asian Font)	§5.1.5.3.3
extLst (Extension List)	§5.1.2.1.15
font (Font)	§5.1.4.1.16
latin (Latin Font)	§5.1.5.3.7

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FontCollection">
   <sequence>
      <element name="latin" type="CT_TextFont" minOccurs="1" maxOccurs="1"/>
      <element name="ea" type="CT_TextFont" minOccurs="1" maxOccurs="1"/>
      <element name="cs" type="CT_TextFont" minOccurs="1" maxOccurs="1"/>
      <element name="font" type="CT_SupplementalFont" minOccurs="0" maxOccurs="unbounded"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.1.26 scene3d (3D Scene Properties)

This element defines optional scene-level 3D properties to apply to an object.

Parent Elements
bodyPr (§5.1.5.1.1); effectStyle (§5.1.4.1.11); grpSpPr (§5.8.2.14); grpSpPr (§4.4.1.20); grpSpPr (§5.1.2.1.22); grpSpPr (§5.6.2.17); spPr (§5.6.2.29); spPr (§5.7.2.198); spPr (§5.8.2.23); spPr (§5.1.2.1.35); spPr (§4.4.1.41); spPr (§5.9.3.7); spPr (§5.2.2.6)

Child Elements	Subclause
backdrop (Backdrop Plane)	§5.1.7.2
camera (Camera)	§5.1.7.5
extLst (Extension List)	§5.1.2.1.15
lightRig (Light Rig)	§5.1.7.9

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Scene3D">
   <sequence>
      <element name="camera" type="CT_Camera" minOccurs="1" maxOccurs="1"/>
      <element name="lightRig" type="CT_LightRig" minOccurs="1" maxOccurs="1"/>
      <element name="backdrop" type="CT_Backdrop" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.1.27 spDef (Shape Default)

This element defines the formatting that is associated with the default shape. The default formatting can be applied to a shape when it is initially inserted into a document.

[*Example*: Consider the following example of a shape default being used within DrawingML:

```
<spDef>
  <spPr>
    <solidFill>
      <schemeClr val="accent2">
        <shade val="75000"/>
      </schemeClr>
    </solidFill>
  </spPr>
  <bodyPr rtlCol="0" anchor="ctr"/>
  <lstStyle>
    <defPPr algn="ctr">
      <defRPr/>
    </defPPr>
  </lstStyle>
  <style>
    <lnRef idx="1">
      <schemeClr val="accent1"/>
    </lnRef>
    <fillRef idx="2">
      <schemeClr val="accent1"/>
    </fillRef>
    <effectRef idx="1">
      <schemeClr val="accent1"/>
    </effectRef>
    <fontRef idx="minor">
      <schemeClr val="dk1"/>
    </fontRef>
  </style>
</spDef>
```

In this example, we see a default shape which references a certain themed fill, line, effect, and font along with an override fill to these. *end example*]

Parent Elements
objectDefaults (§5.1.8.7)

Child Elements	Subclause
bodyPr (Body Properties)	§5.1.5.1.1
extLst (Extension List)	§5.1.2.1.15
lstStyle (Text List Styles)	§5.1.5.4.12
spPr (Shape Properties)	§5.1.2.1.35
style (Shape Style)	§5.1.2.1.37

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DefaultShapeDefinition">
   <sequence>
      <element name="spPr" type="CT_ShapeProperties" minOccurs="1" maxOccurs="1"/>
      <element name="bodyPr" type="CT_TextBodyProperties" minOccurs="1" maxOccurs="1"/>
      <element name="lstStyle" type="CT_TextListStyle" minOccurs="1" maxOccurs="1"/>
      <element name="style" type="CT_ShapeStyle" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.1.28 txDef (Text Default)

This element defines the default formatting which is applied to text in a document by default. The default formatting can and should be applied to the shape when it is initially inserted into a document.

[*Example*: Consider the following example of a text default being used within DrawingML:

```
<txDef>
  <spPr>
    <solidFill>
      <schemeClr val="accent2">
        <shade val="75000"/>
      </schemeClr>
    </solidFill>
  </spPr>
  <bodyPr rtlCol="0" anchor="ctr"/>
  <lstStyle>
    <defPPr algn="ctr">
      <defRPr/>
    </defPPr>
  </lstStyle>
  <style>
    <lnRef idx="1">
      <schemeClr val="accent1"/>
    </lnRef>
```

```
      <fillRef idx="2">
        <schemeClr val="accent1"/>
      </fillRef>
      <effectRef idx="1">
        <schemeClr val="accent1"/>
      </effectRef>
      <fontRef idx="minor">
        <schemeClr val="dk1"/>
      </fontRef>
    </style>
  </txDef>
```

In this example, we see a default text which references a certain themed fill, line, effect, and font along with an override fill to these. *end example*]

Parent Elements
objectDefaults (§5.1.8.7)

Child Elements	Subclause
bodyPr (Body Properties)	§5.1.5.1.1
extLst (Extension List)	§5.1.2.1.15
lstStyle (Text List Styles)	§5.1.5.4.12
spPr (Shape Properties)	§5.1.2.1.35
style (Shape Style)	§5.1.2.1.37

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DefaultShapeDefinition">
   <sequence>
      <element name="spPr" type="CT_ShapeProperties" minOccurs="1" maxOccurs="1"/>
      <element name="bodyPr" type="CT_TextBodyProperties" minOccurs="1" maxOccurs="1"/>
      <element name="lstStyle" type="CT_TextListStyle" minOccurs="1" maxOccurs="1"/>
      <element name="style" type="CT_ShapeStyle" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.2 Table Styles

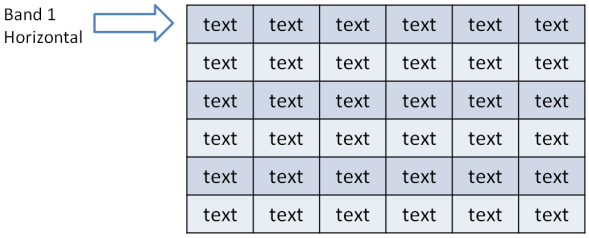
Table styles are responsible for the rapid formatting that can be applied to a table. This rapid formatting takes different parts of a table into account, such as if the first row or last row should be emphasized, or if there is some type of banding (row for example) present on the table. All of these different types of formatting can be defined within a table style

5.1.4.2.1 band1H (Band 1 Horizontal)

This element describes the formatting for the first row in horizontal banding. Two different row formatting are applied to the table alternating in order to create a banding effect on the table.

[*Example*: Consider the following example of band 1 horizontal being used within DrawingML:

```
<band1H>
  <tcStyle>
    <tcBdr/>
    <fill>
      <solidFill>
        <schemeClr val="accent1">
          <tint val="40000"/>
        </schemeClr>
      </solidFill>
    </fill>
  </tcStyle>
</band1H>
```



In this example, we set the fill to be a solid fill referencing the accent 1 color defined in the theme. *end example*]

Parent Elements
tableStyle (§5.1.6.9); tblStyle (§5.1.4.2.26)

Child Elements	Subclause
tcStyle (Table Cell Style)	§5.1.4.2.29
tcTxStyle (Table Cell Text Style)	§5.1.4.2.30

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TablePartStyle">
   <sequence>
      <element name="tcTxStyle" type="CT_TableStyleTextStyle" minOccurs="0" maxOccurs="1"/>
      <element name="tcStyle" type="CT_TableStyleCellStyle" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.2.2 band1V (Band 1 Vertical)

This element describes the formatting for the first row in vertical banding. Two different row formatting are applied to the table alternating in order to create a banding effect on the table.

[*Example*: Consider the following example of band 1 vertical being used within DrawingML:

```
<band1V>
  <tcStyle>
    <tcBdr/>
    <fill>
      <solidFill>
        <schemeClr val="accent1">
          <tint val="40000"/>
        </schemeClr>
      </solidFill>
    </fill>
  </tcStyle>
</band1V>
```

Band 1 Vertical

text	text	text	text	text	text
text	text	text	text	text	text
text	text	text	text	text	text
text	text	text	text	text	text
text	text	text	text	text	text
text	text	text	text	text	text

In this example, we set the fill to be a solid fill referencing the accent 1 color defined in the theme. *end example*]

Parent Elements
tableStyle (§5.1.6.9); tblStyle (§5.1.4.2.26)

Child Elements	Subclause
tcStyle (Table Cell Style)	§5.1.4.2.29
tcTxStyle (Table Cell Text Style)	§5.1.4.2.30

The following XML Schema fragment defines the contents of this element:

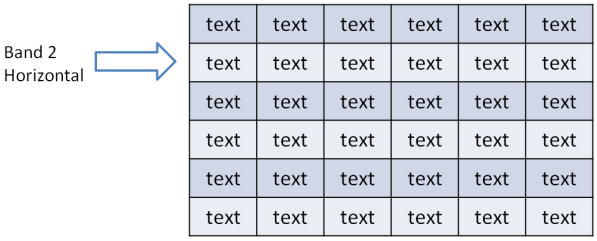
```
<complexType name="CT_TablePartStyle">
   <sequence>
      <element name="tcTxStyle" type="CT_TableStyleTextStyle" minOccurs="0" maxOccurs="1"/>
      <element name="tcStyle" type="CT_TableStyleCellStyle" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.2.3 band2H (Band 2 Horizontal)

This element describes the formatting for the second row in horizontal banding. Two different row formatting are applied to the table alternating in order to create a banding effect on the table.

[*Example*: Consider the following example of band 2 horizontal being used within DrawingML:

```
<band2H>
  <tcStyle>
    <tcBdr/>
    <fill>
      <solidFill>
        <schemeClr val="accent2">
          <tint val="40000"/>
        </schemeClr>
      </solidFill>
    </fill>
  </tcStyle>
</band2H>
```



In this example, we set the fill to be a solid fill referencing the accent 2 color defined in the theme. *end example*]

Parent Elements
tableStyle (§5.1.6.9); tblStyle (§5.1.4.2.26)

Child Elements	Subclause
tcStyle (Table Cell Style)	§5.1.4.2.29
tcTxStyle (Table Cell Text Style)	§5.1.4.2.30

The following XML Schema fragment defines the contents of this element:

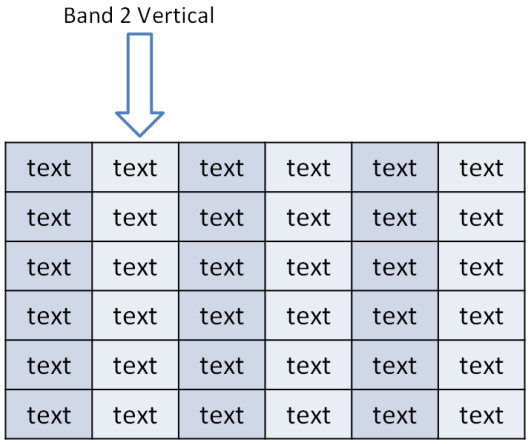
```
<complexType name="CT_TablePartStyle">
   <sequence>
      <element name="tcTxStyle" type="CT_TableStyleTextStyle" minOccurs="0" maxOccurs="1"/>
      <element name="tcStyle" type="CT_TableStyleCellStyle" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.2.4 band2V (Band 2 Vertical)

This element describes the formatting for the second row in vertical banding. Two different row formatting are applied to the table alternating in order to create a banding effect on the table.

[*Example*: Consider the following example of band 2 vertical being used within DrawingML:

```
<band2V>
  <tcStyle>
    <tcBdr/>
    <fill>
      <solidFill>
        <schemeClr val="accent2">
          <tint val="40000"/>
        </schemeClr>
      </solidFill>
    </fill>
  </tcStyle>
</band2V>
```



In this example, we set the fill to be a solid fill referencing the accent 2 color defined in the theme. *end example*]

Parent Elements
tableStyle (§5.1.6.9); tblStyle (§5.1.4.2.26)

Child Elements	Subclause
tcStyle (Table Cell Style)	§5.1.4.2.29
tcTxStyle (Table Cell Text Style)	§5.1.4.2.30

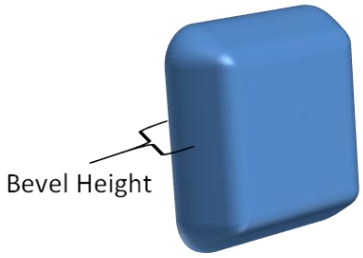
The following XML Schema fragment defines the contents of this element:

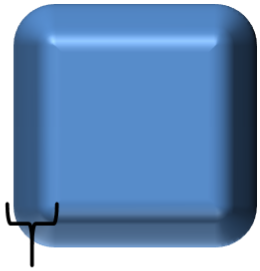
```
<complexType name="CT_TablePartStyle">
   <sequence>
      <element name="tcTxStyle" type="CT_TableStyleTextStyle" minOccurs="0" maxOccurs="1"/>
      <element name="tcStyle" type="CT_TableStyleCellStyle" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.2.5 bevel (Bevel)

This element defines the properties of the bevel associated with the 3D effect applied to a cell in a table.

Parent Elements
cell3D (§5.1.6.1)

Attributes	Description
h (Height)	Specifies the height of the bevel, or how far above the shape it is applied. [*Example*: Consider the following example bevel  In this example, we see the height of an example bevel on a shape. *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).
prst (Preset Bevel)	Specifies the preset bevel type which defines the look of the bevel. The possible values for this attribute are defined by the ST_BevelPresetType simple type (§5.1.12.9).
w (Width)	Specifies the width of the bevel, or how far into the shape it is applied.

Attributes	Description
	[*Example*: Consider the following example bevel  Bevel Width In this example, we see the width of an example bevel on a shape. *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Bevel">
   <attribute name="w" type="ST_PositiveCoordinate" use="optional" default="76200"/>
   <attribute name="h" type="ST_PositiveCoordinate" use="optional" default="76200"/>
   <attribute name="prst" type="ST_BevelPresetType" use="optional" default="circle"/>
</complexType>
```

5.1.4.2.6 bottom (Bottom Border)

This element defines the line properties associated with the bottom border in a table cell.

[*Example*: Consider the following example of the bottom border in use within DrawingML:

```
<bottom>
  <ln w="12700" cmpd="sng">
    <solidFill>
      <schemeClr val="accent1"/>
    </solidFill>
  </ln>
</bottom>
```

In this example, we see the bottom border on a table cell to be a single 1pt line which is colored accent 1. *end example*]

Parent Elements
tcBdr (§5.1.4.2.28)

Child Elements	Subclause

Child Elements	Subclause
ln (Outline)	§5.1.2.1.24
lnRef (Line Reference)	§5.1.4.2.19

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ThemeableLineStyle">
   <choice>
      <element name="ln" type="CT_LineProperties" minOccurs="1" maxOccurs="1"/>
      <element name="lnRef" type="CT_StyleMatrixReference" minOccurs="1" maxOccurs="1"/>
   </choice>
</complexType>
```

5.1.4.2.7 effect (Effect)

This element defines the effect that can be applied to a table as a whole through a table style.

[*Example*: Consider the following example of an effect in use within DrawingML:

```
<effect>
  <effectLst>
    <glow rad="228600">
      <schemeClr val="accent1">
        <satMod val="175000"/>
        <alpha val="40000"/>
      </schemeClr>
    </glow>
  </effectLst>
</effect>
```

In this example, we see a glow being defined within the table style that will be applied to the table as a whole. *end example*]

Parent Elements
tblBg (§5.1.4.2.25)

Child Elements	Subclause
effectDag (Effect Container)	§5.1.10.25
effectLst (Effect Container)	§5.1.10.26

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EffectProperties">
   <sequence>
      <group ref="EG_EffectProperties" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.2.8 effectRef (Effect Reference)

This element defines a reference to an effect style within the style matrix. The idx attribute refers the index of an effect style within the effectStyleLst element.

Parent Elements
style (§5.6.2.30); style (§5.9.2.28); style (§4.4.1.43); style (§5.8.2.24); style (§5.1.2.1.37); tblBg (§5.1.4.2.25)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

Attributes	Description
idx (Style Matrix Index)	Specifies the style matrix index of the style referred to. The possible values for this attribute are defined by the ST_StyleMatrixColumnIndex simple type (§5.1.12.57).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_StyleMatrixReference">
   <sequence>
      <group ref="EG_ColorChoice" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="idx" type="ST_StyleMatrixColumnIndex" use="required"/>
</complexType>
```

5.1.4.2.9 fill (Fill)

This element defines the fill that is applied to the table as a whole. The background of the table can contain a single fill that is the entire size of the table. This can allow for gradient fills, or image fills, which span the entire size of the table.

[*Example*: Consider the following example of a fill on a table background in DrawingML:

```
<fill>
  <gradFill flip="none" rotWithShape="1">
    <gsLst>
      <gs pos="0">
        <schemeClr val="accent2">
          <shade val="75000"/>
        </schemeClr>
      </gs>
      <gs pos="100000">
        <schemeClr val="accent2">
          <shade val="75000"/>
          <tint val="20000"/>
        </schemeClr>
      </gs>
    </gsLst>
    <lin ang="2700000" scaled="1"/>
    <tileRect/>
  </gradFill>
</fill>
```

In this example, we apply a gradient fill to the entire table on the background shape of the table. *end example*]

Parent Elements
tblBg (§5.1.4.2.25); tcStyle (§5.1.4.2.29)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
solidFill (Solid Fill)	§5.1.10.54

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FillProperties">
   <sequence>
      <group ref="EG_FillProperties" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.2.10 fillRef (Fill Reference)

This element defines a reference to a fill style within the style matrix. The idx attribute refers to the index of a fill style or background fill style within the presentation's style matrix, defined by the fmtScheme element. A value of 0 or 1000 indicates no background, values 1-999 refer to the index of a fill style within the fillStyleLst element, and values 1001 and above refer to the index of a background fill style within the bgFillStyleLst element. The value 1001 corresponds to the first background fill style, 1002 to the second background fill style, and so on.

[*Example*:

```
<a:fillRef idx="2">
  <a:schemeClr val="accent2"/>
</a:fillRef>
```

The above code indicates the object is to have the style's second fill style using the accent2 color of the color scheme.

end example]

[*Example*:

```
<a:fillRef idx="1001">
  <a:schemeClr val="accent2"/>
</a:fillRef>
```

The above code indicates the object is to have the style's first background fill style using the accent2 color of the color scheme.

end example]

Parent Elements
style (§5.6.2.30); style (§5.9.2.28); style (§4.4.1.43); style (§5.8.2.24); style (§5.1.2.1.37); tblBg (§5.1.4.2.25); tcStyle (§5.1.4.2.29)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

Attributes	Description
idx (Style Matrix Index)	Specifies the style matrix index of the style referred to. The possible values for this attribute are defined by the ST_StyleMatrixColumnIndex simple type (§5.1.12.57).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_StyleMatrixReference">
   <sequence>
      <group ref="EG_ColorChoice" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="idx" type="ST_StyleMatrixColumnIndex" use="required"/>
</complexType>
```

5.1.4.2.11 firstCol (First Column)

This element defines the cell formatting which can be applied to the first column of the table.

[*Example*: Consider the following example of first column formatting within DrawingML:

```
<firstCol>
  <tcTxStyle b="on">
    <fontRef idx="minor">
      <scrgbClr r="0" g="0" b="0"/>
    </fontRef>
    <schemeClr val="lt1"/>
  </tcTxStyle>
  <tcStyle>
    <tcBdr/>
    <fill>
      <solidFill>
        <schemeClr val="accent1"/>
      </solidFill>
    </fill>
  </tcStyle>
</firstCol>
```

First Column

text	text	text	text	text	text
text	text	text	text	text	text
text	text	text	text	text	text
text	text	text	text	text	text
text	text	text	text	text	text
text	text	text	text	text	text

In this example, we define the first column cell fills to be accent 1 along with the text properties to be bold when first column formatting is enabled through the user interface. *end example*]

Parent Elements
tableStyle (§5.1.6.9); tblStyle (§5.1.4.2.26)

Child Elements	Subclause
tcStyle (Table Cell Style)	§5.1.4.2.29
tcTxStyle (Table Cell Text Style)	§5.1.4.2.30

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TablePartStyle">
   <sequence>
      <element name="tcTxStyle" type="CT_TableStyleTextStyle" minOccurs="0" maxOccurs="1"/>
      <element name="tcStyle" type="CT_TableStyleCellStyle" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.2.12 firstRow (First Row)

This element defines the cell formatting which can be applied to the first row of the table.

[*Example*: Consider the following example of first row formatting within DrawingML:

```
<firstRow>
  <tcTxStyle b="on">
    <fontRef idx="minor">
      <scrgbClr r="0" g="0" b="0"/>
    </fontRef>
    <schemeClr val="lt1"/>
  </tcTxStyle>
```

```
    <tcStyle>
      <tcBdr/>
      <fill>
        <solidFill>
          <schemeClr val="accent1"/>
        </solidFill>
      </fill>
    </tcStyle>
  </firstRow>
```

In this example, we define the first row cell fills to be accent 1 along with the text properties to be bold when first row formatting is enabled through the user interface. *end example*]

Parent Elements
tableStyle (§5.1.6.9); tblStyle (§5.1.4.2.26)

Child Elements	Subclause
tcStyle (Table Cell Style)	§5.1.4.2.29
tcTxStyle (Table Cell Text Style)	§5.1.4.2.30

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TablePartStyle">
   <sequence>
      <element name="tcTxStyle" type="CT_TableStyleTextStyle" minOccurs="0" maxOccurs="1"/>
      <element name="tcStyle" type="CT_TableStyleCellStyle" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.2.13 font (Font)

This element defines the font to be used within a given table cell text style. This element allows for exact definition of the font within the table style instead of referencing a themed font.

Parent Elements
tcTxStyle (§5.1.4.2.30)

Child Elements	Subclause
cs (Complex Script Font)	§5.1.5.3.1
ea (East Asian Font)	§5.1.5.3.3
extLst (Extension List)	§5.1.2.1.15
font (Font)	§5.1.4.1.16
latin (Latin Font)	§5.1.5.3.7

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FontCollection">
   <sequence>
      <element name="latin" type="CT_TextFont" minOccurs="1" maxOccurs="1"/>
      <element name="ea" type="CT_TextFont" minOccurs="1" maxOccurs="1"/>
      <element name="cs" type="CT_TextFont" minOccurs="1" maxOccurs="1"/>
      <element name="font" type="CT_SupplementalFont" minOccurs="0" maxOccurs="unbounded"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.2.14 insideH (Inside Horizontal Border)

This element defines the line properties associated with the inner horizontal borders in a table.

[*Example*: Consider the following example of the inner horizontal borders in use within DrawingML:

```
<insideH>
  <ln w="12700" cmpd="sng">
    <solidFill>
      <schemeClr val="accent1"/>
    </solidFill>
  </ln>
</insideH>
```

In this example, we see the inner horizontal borders in a table to be a single 1pt line which is colored accent 1. *end example*]

Parent Elements
tcBdr (§5.1.4.2.28)

Child Elements	Subclause
ln (Outline)	§5.1.2.1.24
lnRef (Line Reference)	§5.1.4.2.19

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ThemeableLineStyle">
   <choice>
      <element name="ln" type="CT_LineProperties" minOccurs="1" maxOccurs="1"/>
      <element name="lnRef" type="CT_StyleMatrixReference" minOccurs="1" maxOccurs="1"/>
   </choice>
</complexType>
```

5.1.4.2.15 insideV (Inside Vertical Border)

This element defines the line properties associated with the inner vertical borders in a table.

[*Example*: Consider the following example of the inside vertical borders in use within DrawingML:

```
<insideV>
  <ln w="12700" cmpd="sng">
    <solidFill>
      <schemeClr val="accent1"/>
    </solidFill>
  </ln>
</insideV>
```

In this example, we see the inner vertical borders in a table to be a single 1pt line which is colored accent 1. *end example*]

Parent Elements
tcBdr (§5.1.4.2.28)

Child Elements	Subclause
ln (Outline)	§5.1.2.1.24
lnRef (Line Reference)	§5.1.4.2.19

The following XML Schema fragment defines the contents of this element:

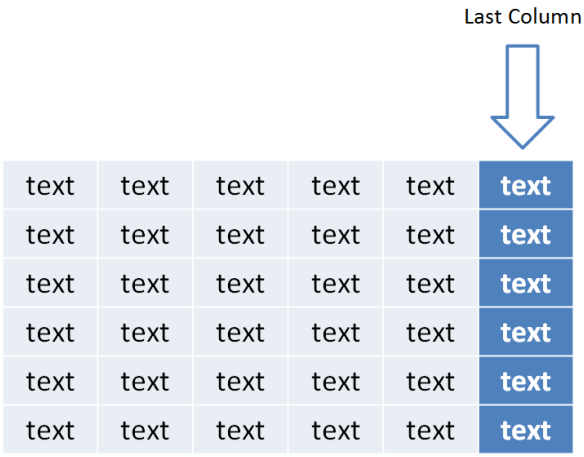
```
<complexType name="CT_ThemeableLineStyle">
   <choice>
      <element name="ln" type="CT_LineProperties" minOccurs="1" maxOccurs="1"/>
      <element name="lnRef" type="CT_StyleMatrixReference" minOccurs="1" maxOccurs="1"/>
   </choice>
</complexType>
```

5.1.4.2.16 lastCol (Last Column)

This element defines the cell formatting which can be applied to the last column of the table.

[*Example*: Consider the following example of last column formatting within DrawingML:

```
<lastCol>
  <tcTxStyle b="on">
    <fontRef idx="minor">
      <scrgbClr r="0" g="0" b="0"/>
    </fontRef>
    <schemeClr val="lt1"/>
  </tcTxStyle>
  <tcStyle>
    <tcBdr/>
    <fill>
      <solidFill>
        <schemeClr val="accent1"/>
      </solidFill>
    </fill>
  </tcStyle>
</lastCol>
```



In this example, we define the last column cell fills to be accent 1 along with the text properties to be bold when last column formatting is enabled through the user interface. *end example*]

Parent Elements
tableStyle (§5.1.6.9); tblStyle (§5.1.4.2.26)

Child Elements	Subclause
tcStyle (Table Cell Style)	§5.1.4.2.29
tcTxStyle (Table Cell Text Style)	§5.1.4.2.30

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TablePartStyle">
   <sequence>
      <element name="tcTxStyle" type="CT_TableStyleTextStyle" minOccurs="0" maxOccurs="1"/>
      <element name="tcStyle" type="CT_TableStyleCellStyle" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.2.17 lastRow (Last Row)

This element defines the cell formatting which can be applied to the last row of the table.

[*Example*: Consider the following example of last row formatting within DrawingML:

```
<lastRow>
  <tcTxStyle b="on">
    <fontRef idx="minor">
      <scrgbClr r="0" g="0" b="0"/>
    </fontRef>
    <schemeClr val="lt1"/>
  </tcTxStyle>
  <tcStyle>
    <tcBdr/>
    <fill>
      <solidFill>
        <schemeClr val="accent1"/>
      </solidFill>
    </fill>
  </tcStyle>
</lastRow>
```

In this example, we define the last row cell fills to be accent 1 along with the text properties to be bold when last row formatting is enabled through the user interface. *end example*]

Parent Elements
tableStyle (§5.1.6.9); tblStyle (§5.1.4.2.26)

Child Elements	Subclause
tcStyle (Table Cell Style)	§5.1.4.2.29
tcTxStyle (Table Cell Text Style)	§5.1.4.2.30

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TablePartStyle">
   <sequence>
      <element name="tcTxStyle" type="CT_TableStyleTextStyle" minOccurs="0" maxOccurs="1"/>
      <element name="tcStyle" type="CT_TableStyleCellStyle" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.2.18 left (Left Border)

This element defines the line properties associated with the left border in a table cell.

[*Example*: Consider the following example of the left border in use within DrawingML:

```
<left>
  <ln w="12700" cmpd="sng">
    <solidFill>
      <schemeClr val="accent1"/>
    </solidFill>
  </ln>
</left>
```

In this example, we see the left border on a table cell to be a single 1pt line which is colored accent 1. *end example*]

Parent Elements
tcBdr (§5.1.4.2.28)

Child Elements	Subclause
ln (Outline)	§5.1.2.1.24
lnRef (Line Reference)	§5.1.4.2.19

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ThemeableLineStyle">
   <choice>
      <element name="ln" type="CT_LineProperties" minOccurs="1" maxOccurs="1"/>
      <element name="lnRef" type="CT_StyleMatrixReference" minOccurs="1" maxOccurs="1"/>
   </choice>
</complexType>
```

5.1.4.2.19 lnRef (Line Reference)

This element defines a reference to a line style within the style matrix. The idx attribute refers the index of a line style within the fillStyleLst element.

Parent Elements
bottom (§5.1.4.2.6); insideH (§5.1.4.2.14); insideV (§5.1.4.2.15); left (§5.1.4.2.18); right (§5.1.4.2.22); style (§5.6.2.30); style (§5.9.2.28); style (§4.4.1.43); style (§5.8.2.24); style (§5.1.2.1.37); tl2br (§5.1.4.2.31); top (§5.1.4.2.32); tr2bl (§5.1.4.2.33)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

Attributes	Description
idx (Style Matrix Index)	Specifies the style matrix index of the style referred to. The possible values for this attribute are defined by the ST_StyleMatrixColumnIndex simple type (§5.1.12.57).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_StyleMatrixReference">
   <sequence>
      <group ref="EG_ColorChoice" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="idx" type="ST_StyleMatrixColumnIndex" use="required"/>
</complexType>
```

5.1.4.2.20 neCell (Northeast Cell)

This element defies the formatting for the cell in the northeast corner of a table when both the first row formatting and last column formatting are enabled. This formatting is only applied to the single cell which overlaps between the two formatting options.

[*Example*: Consider the following example of the northeast cell formatting within DrawingML:

```
<neCell>
  <tcTxStyle b="on">
    <fontRef idx="minor">
      <scrgbClr r="0" g="0" b="0"/>
    </fontRef>
    <schemeClr val="lt1"/>
  </tcTxStyle>
  <tcStyle>
    <tcBdr/>
    <fill>
      <solidFill>
        <schemeClr val="accent1"/>
      </solidFill>
    </fill>
  </tcStyle>
</neCell>
```

In this example, we specifically set the northeast cell to contain bold text with a solid cell fill in the color of accent 1. *end example*]

Parent Elements
tableStyle (§5.1.6.9); tblStyle (§5.1.4.2.26)

Child Elements	Subclause
tcStyle (Table Cell Style)	§5.1.4.2.29
tcTxStyle (Table Cell Text Style)	§5.1.4.2.30

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TablePartStyle">
   <sequence>
      <element name="tcTxStyle" type="CT_TableStyleTextStyle" minOccurs="0" maxOccurs="1"/>
      <element name="tcStyle" type="CT_TableStyleCellStyle" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.2.21 nwCell (Northwest Cell)

This element defies the formatting for the cell in the northwest corner of a table when both the first row formatting and first column formatting are enabled. This formatting is only applied to the single cell which overlaps between the two formatting options.

[*Example*: Consider the following example of the northwest cell formatting within DrawingML:

```
<nwCell>
  <tcTxStyle b="on">
    <fontRef idx="minor">
      <scrgbClr r="0" g="0" b="0"/>
    </fontRef>
    <schemeClr val="lt1"/>
  </tcTxStyle>
  <tcStyle>
    <tcBdr/>
    <fill>
      <solidFill>
        <schemeClr val="accent1"/>
      </solidFill>
    </fill>
  </tcStyle>
</nwCell>
```

In this example, we specifically set the northwest cell to contain bold text with a solid cell fill in the color of accent 1. *end example*]

Parent Elements
tableStyle (§5.1.6.9); tblStyle (§5.1.4.2.26)

Child Elements	Subclause
tcStyle (Table Cell Style)	§5.1.4.2.29
tcTxStyle (Table Cell Text Style)	§5.1.4.2.30

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TablePartStyle">
   <sequence>
      <element name="tcTxStyle" type="CT_TableStyleTextStyle" minOccurs="0" maxOccurs="1"/>
      <element name="tcStyle" type="CT_TableStyleCellStyle" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.2.22 right (Right Border)

This element defines the line properties associated with the right border in a table cell.

[*Example*: Consider the following example of the right border in use within DrawingML:

```
<right>
  <ln w="12700" cmpd="sng">
    <solidFill>
      <schemeClr val="accent1"/>
    </solidFill>
  </ln>
</right>
```

In this example, we see the right border on a table cell to be a single 1pt line which is colored accent 1. *end example*]

Parent Elements
tcBdr (§5.1.4.2.28)

Child Elements	Subclause
ln (Outline)	§5.1.2.1.24
lnRef (Line Reference)	§5.1.4.2.19

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ThemeableLineStyle">
   <choice>
      <element name="ln" type="CT_LineProperties" minOccurs="1" maxOccurs="1"/>
      <element name="lnRef" type="CT_StyleMatrixReference" minOccurs="1" maxOccurs="1"/>
   </choice>
</complexType>
```

5.1.4.2.23 seCell (Southeast Cell)

This element defies the formatting for the cell in the southeast corner of a table when both the last row formatting and last column formatting are enabled. This formatting is only applied to the single cell which overlaps between the two formatting options.

[*Example*: Consider the following example of the southeast cell formatting within DrawingML:

```
<seCell>
  <tcTxStyle b="on">
    <fontRef idx="minor">
      <scrgbClr r="0" g="0" b="0"/>
    </fontRef>
    <schemeClr val="lt1"/>
  </tcTxStyle>
```

```
    <tcStyle>
      <tcBdr/>
      <fill>
        <solidFill>
          <schemeClr val="accent1"/>
        </solidFill>
      </fill>
    </tcStyle>
  </seCell>
```

In this example, we specifically set the southeast cell to contain bold text with a solid cell fill in the color of accent 1. *end example*]

Parent Elements
tableStyle (§5.1.6.9); tblStyle (§5.1.4.2.26)

Child Elements	Subclause
tcStyle (Table Cell Style)	§5.1.4.2.29
tcTxStyle (Table Cell Text Style)	§5.1.4.2.30

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TablePartStyle">
   <sequence>
      <element name="tcTxStyle" type="CT_TableStyleTextStyle" minOccurs="0" maxOccurs="1"/>
      <element name="tcStyle" type="CT_TableStyleCellStyle" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.2.24 swCell (Southwest Cell)

This element defies the formatting for the cell in the southwest corner of a table when both the last row formatting and first column formatting are enabled. This formatting is only applied to the single cell which overlaps between the two formatting options.

[*Example*: Consider the following example of the southwest cell formatting within DrawingML:

```
  <swCell>
    <tcTxStyle b="on">
      <fontRef idx="minor">
        <scrgbClr r="0" g="0" b="0"/>
      </fontRef>
      <schemeClr val="lt1"/>
    </tcTxStyle>
```

```
    <tcStyle>
      <tcBdr/>
      <fill>
        <solidFill>
          <schemeClr val="accent1"/>
        </solidFill>
      </fill>
    </tcStyle>
  </swCell>
```

In this example, we specifically set the southwest cell to contain bold text with a solid cell fill in the color of accent 1. *end example*]

Parent Elements
tableStyle (§5.1.6.9); tblStyle (§5.1.4.2.26)

Child Elements	Subclause
tcStyle (Table Cell Style)	§5.1.4.2.29
tcTxStyle (Table Cell Text Style)	§5.1.4.2.30

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TablePartStyle">
   <sequence>
      <element name="tcTxStyle" type="CT_TableStyleTextStyle" minOccurs="0" maxOccurs="1"/>
      <element name="tcStyle" type="CT_TableStyleCellStyle" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.2.25 tblBg (Table Background)

This element defines the formatting options which can be applied to the table background shape. The background shape is the same size as the entire table and can hold a fill or an effect which spans the entire table.

[*Example*: Consider the following example of a table background in use within DrawingML:

```
  <tblBg>
    <fillRef idx="2">
      <schemeClr val="accent1"/>
    </fillRef>
    <effectRef idx="1">
      <schemeClr val="accent1"/>
    </effectRef>
  </tblBg>
```

In this example, we see that there is a themed fill and themed effect being applied to the table background through the table style. *end example*]

Parent Elements
tableStyle (§5.1.6.9); tblStyle (§5.1.4.2.26)

Child Elements	Subclause
effect (Effect)	§5.1.4.2.7
effectRef (Effect Reference)	§5.1.4.2.8
fill (Fill)	§5.1.4.2.9
fillRef (Fill Reference)	§5.1.4.2.10

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TableBackgroundStyle">
   <sequence>
      <group ref="EG_ThemeableFillStyle" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_ThemeableEffectStyle" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.2.26 tblStyle (Table Style)

This is the root element for a table style. Within the table style are different formatting options available in order to apply a table.

Parent Elements
tblStyleLst (§5.1.4.2.27)

Child Elements	Subclause
band1H (Band 1 Horizontal)	§5.1.4.2.1
band1V (Band 1 Vertical)	§5.1.4.2.2
band2H (Band 2 Horizontal)	§5.1.4.2.3
band2V (Band 2 Vertical)	§5.1.4.2.4
extLst (Extension List)	§5.1.2.1.15
firstCol (First Column)	§5.1.4.2.11
firstRow (First Row)	§5.1.4.2.12
lastCol (Last Column)	§5.1.4.2.16
lastRow (Last Row)	§5.1.4.2.17
neCell (Northeast Cell)	§5.1.4.2.20

Child Elements	Subclause
nwCell (Northwest Cell)	§5.1.4.2.21
seCell (Southeast Cell)	§5.1.4.2.23
swCell (Southwest Cell)	§5.1.4.2.24
tblBg (Table Background)	§5.1.4.2.25
wholeTbl (Whole Table)	§5.1.4.2.34

Attributes	Description
styleId (Style ID)	Specifies a GUID identifying the table style in a unique manner. The possible values for this attribute are defined by the ST_Guid simple type (§5.1.12.27).
styleName (Name)	Specifies the name of the table style which can show up in the user interface identifying the style to a user. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TableStyle">
   <sequence>
      <element name="tblBg" type="CT_TableBackgroundStyle" minOccurs="0" maxOccurs="1"/>
      <element name="wholeTbl" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="band1H" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="band2H" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="band1V" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="band2V" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="lastCol" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="firstCol" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="lastRow" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="seCell" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="swCell" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="firstRow" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="neCell" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="nwCell" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="styleId" type="ST_Guid" use="required"/>
   <attribute name="styleName" type="xsd:string" use="required"/>
</complexType>
```

5.1.4.2.27 tblStyleLst (Table Style List)

This element is simply a list of table styles which are used within a document.

[*Example*: Consider the following example of a table style list within DrawingML:

```
<tblStyleLst def="{5C22544A-7EE6-4342-B048-85BDC9FD1C3A}">
  <tblStyle styleId="{5C22544A-7EE6-4342-B048-85BDC9FD1C3A}"
    styleName="Medium Style 2 - Accent 1">
    …
  </tblStyle>
  <tblStyle styleId="{3C2FFA5D-87B4-456A-9821-1D502468CF0F}"
    styleName="Themed Style 1 - Accent 1">
    …
  </tblStyle>
</tblStyleLst>
```

In this example, we see two table styles defined along with the default being specified. *end example*]

Parent Elements
Root element of DrawingML Table Styles part

Child Elements	Subclause
tblStyle (Table Style)	§5.1.4.2.26

Attributes	Description
def (Default)	The GUID corresponding to the default table style in the list of table styles. This default can be used when a table is initially inserted into a document. The possible values for this attribute are defined by the ST_Guid simple type (§5.1.12.27).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TableStyleList">
   <sequence>
      <element name="tblStyle" type="CT_TableStyle" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="def" type="ST_Guid" use="required"/>
</complexType>
```

5.1.4.2.28 tcBdr (Table Cell Borders)

This element defines the borders for the cells within a table.

[*Example*: Consider the following example of table cell borders being used within DrawingML:

```
<tcBdr>
  <left>
    <lnRef idx="1">
      <schemeClr val="accent1"/>
    </lnRef>
  </left>
  <right>
    <lnRef idx="1">
      <schemeClr val="accent1"/>
    </lnRef>
  </right>
  <top>
    <lnRef idx="1">
      <schemeClr val="accent1"/>
    </lnRef>
  </top>
  <bottom>
    <lnRef idx="2">
      <schemeClr val="lt1"/>
    </lnRef>
  </bottom>
  <insideH>
    <ln>
      <noFill/>
    </ln>
  </insideH>
  <insideV>
    <ln>
      <noFill/>
    </ln>
  </insideV>
</tcBdr>
```

In this example, we define borders for the bottom, top, right, and left borders of the table cells. *end example*]

Parent Elements
tcStyle (§5.1.4.2.29)

Child Elements	Subclause
bottom (Bottom Border)	§5.1.4.2.6
extLst (Extension List)	§5.1.2.1.15
insideH (Inside Horizontal Border)	§5.1.4.2.14

Child Elements	Subclause
insideV (Inside Vertical Border)	§5.1.4.2.15
left (Left Border)	§5.1.4.2.18
right (Right Border)	§5.1.4.2.22
tl2br (Top Left to Bottom Right Border)	§5.1.4.2.31
top (Top Border)	§5.1.4.2.32
tr2bl (Top Right to Bottom Left Border)	§5.1.4.2.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TableCellBorderStyle">
   <sequence>
      <element name="left" type="CT_ThemeableLineStyle" minOccurs="0" maxOccurs="1"/>
      <element name="right" type="CT_ThemeableLineStyle" minOccurs="0" maxOccurs="1"/>
      <element name="top" type="CT_ThemeableLineStyle" minOccurs="0" maxOccurs="1"/>
      <element name="bottom" type="CT_ThemeableLineStyle" minOccurs="0" maxOccurs="1"/>
      <element name="insideH" type="CT_ThemeableLineStyle" minOccurs="0" maxOccurs="1"/>
      <element name="insideV" type="CT_ThemeableLineStyle" minOccurs="0" maxOccurs="1"/>
      <element name="tl2br" type="CT_ThemeableLineStyle" minOccurs="0" maxOccurs="1"/>
      <element name="tr2bl" type="CT_ThemeableLineStyle" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.2.29 tcStyle (Table Cell Style)

This element defines the style for a give cell in a table.

[*Example*: Consider the following example of a table cell style in use within DrawingML:

```
<tcStyle>
  <tcBdr>
    …
  </tcBdr>
  <fill>
    …
  </fill>
</tcStyle>
```

In this example, we see that a set of borders for the cell along with a cell fill are being defined. *end example*]

Parent Elements
band1H (§5.1.4.2.1); band1V (§5.1.4.2.2); band2H (§5.1.4.2.3); band2V (§5.1.4.2.4); firstCol (§5.1.4.2.11); firstRow (§5.1.4.2.12); lastCol (§5.1.4.2.16); lastRow (§5.1.4.2.17); neCell (§5.1.4.2.20); nwCell (§5.1.4.2.21); seCell (§5.1.4.2.23); swCell (§5.1.4.2.24); wholeTbl (§5.1.4.2.34)

Child Elements	Subclause
cell3D (Cell 3-D)	§5.1.6.1
fill (Fill)	§5.1.4.2.9
fillRef (Fill Reference)	§5.1.4.2.10
tcBdr (Table Cell Borders)	§5.1.4.2.28

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TableStyleCellStyle">
   <sequence>
      <element name="tcBdr" type="CT_TableCellBorderStyle" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_ThemeableFillStyle" minOccurs="0" maxOccurs="1"/>
      <element name="cell3D" type="CT_Cell3D" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.4.2.30 tcTxStyle (Table Cell Text Style)

This element defines the text properties associated with the text contained within a table cell.

[*Example*: Consider the following example of a table cell text style in use within DrawingML:

```
<tcTxStyle b="on">
  <fontRef idx="minor">
    <scrgbClr r="0" g="0" b="0"/>
  </fontRef>
  <schemeClr val="lt1"/>
</tcTxStyle>
```

In this example, we define the text within the cell to be bold and reference the themed minor font and to also be the light 1 color. *end example*]

Parent Elements
band1H (§5.1.4.2.1); band1V (§5.1.4.2.2); band2H (§5.1.4.2.3); band2V (§5.1.4.2.4); firstCol (§5.1.4.2.11); firstRow (§5.1.4.2.12); lastCol (§5.1.4.2.16); lastRow (§5.1.4.2.17); neCell (§5.1.4.2.20); nwCell (§5.1.4.2.21); seCell (§5.1.4.2.23); swCell (§5.1.4.2.24); wholeTbl (§5.1.4.2.34)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
font (Font)	§5.1.4.2.13
fontRef (Font Reference)	§5.1.4.1.17
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29

Child Elements	Subclause
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

Attributes	Description
b (Bold)	Specifies if the text is to be bolded. The possible values for this attribute are defined by the ST_OnOffStyleType simple type (§5.1.12.36).
i (Italic)	Specifies if the text is to be italicized. The possible values for this attribute are defined by the ST_OnOffStyleType simple type (§5.1.12.36).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TableStyleTextStyle">
   <sequence>
      <group ref="EG_ThemeableFontStyles" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_ColorChoice" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="b" type="ST_OnOffStyleType" use="optional" default="def"/>
   <attribute name="i" type="ST_OnOffStyleType" use="optional" default="def"/>
</complexType>
```

5.1.4.2.31 tl2br (Top Left to Bottom Right Border)

This element defines the line properties associated with the border which goes from the top-left to the bottom-right corner in a table cell.

[*Example*: Consider the following example of the top border in use within DrawingML:

```
<tl2br>
  <ln w="12700" cmpd="sng">
    <solidFill>
      <schemeClr val="accent1"/>
    </solidFill>
  </ln>
</tl2br>
```

In this example, we see the border on a table cell to be a single 1pt line which is colored accent 1. *end example*]

Parent Elements
tcBdr (§5.1.4.2.28)

Child Elements	Subclause
ln (Outline)	§5.1.2.1.24
lnRef (Line Reference)	§5.1.4.2.19

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ThemeableLineStyle">
   <choice>
      <element name="ln" type="CT_LineProperties" minOccurs="1" maxOccurs="1"/>
      <element name="lnRef" type="CT_StyleMatrixReference" minOccurs="1" maxOccurs="1"/>
   </choice>
</complexType>
```

5.1.4.2.32 top (Top Border)

This element defines the line properties associated with the top border in a table cell.

[*Example*: Consider the following example of the top border in use within DrawingML:

```
<top>
  <ln w="12700" cmpd="sng">
    <solidFill>
      <schemeClr val="accent1"/>
    </solidFill>
  </ln>
</top>
```

In this example, we see the top border on a table cell to be a single 1pt line which is colored accent 1. *end example*]

Parent Elements
tcBdr (§5.1.4.2.28)

Child Elements	Subclause
ln (Outline)	§5.1.2.1.24
lnRef (Line Reference)	§5.1.4.2.19

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ThemeableLineStyle">
   <choice>
      <element name="ln" type="CT_LineProperties" minOccurs="1" maxOccurs="1"/>
      <element name="lnRef" type="CT_StyleMatrixReference" minOccurs="1" maxOccurs="1"/>
   </choice>
</complexType>
```

5.1.4.2.33 tr2bl (Top Right to Bottom Left Border)

This element defines the line properties associated with the border which goes from the top-right to the bottom-left corner in a table cell.

[*Example*: Consider the following example of the top border in use within DrawingML:

```
<tr2bl>
  <ln w="12700" cmpd="sng">
    <solidFill>
      <schemeClr val="accent1"/>
    </solidFill>
  </ln>
</tr2bl>
```

In this example, we see the border on a table cell to be a single 1pt line which is colored accent 1. *end example*]

Parent Elements
tcBdr (§5.1.4.2.28)

Child Elements	Subclause
ln (Outline)	§5.1.2.1.24
lnRef (Line Reference)	§5.1.4.2.19

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ThemeableLineStyle">
   <choice>
      <element name="ln" type="CT_LineProperties" minOccurs="1" maxOccurs="1"/>
      <element name="lnRef" type="CT_StyleMatrixReference" minOccurs="1" maxOccurs="1"/>
   </choice>
</complexType>
```

5.1.4.2.34 wholeTbl (Whole Table)

This element contains formatting options which are applied to the table as a whole when it is in its default state with no formatting options (first row, last row, etc) enabled.

[*Example*: Consider the following example of whole table being used within DrawingML:

```
<wholeTbl>
  <tcTxStyle>
    …
  </tcTxStyle>
  <tcStyle>
    …
  </tcStyle>
</wholeTbl>
```

In this example, we see definitions for the text and the cells within the table. *end example*]

Parent Elements
tableStyle (§5.1.6.9); tblStyle (§5.1.4.2.26)

Child Elements	Subclause
tcStyle (Table Cell Style)	§5.1.4.2.29
tcTxStyle (Table Cell Text Style)	§5.1.4.2.30

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TablePartStyle">
   <sequence>
      <element name="tcTxStyle" type="CT_TableStyleTextStyle" minOccurs="0" maxOccurs="1"/>
      <element name="tcStyle" type="CT_TableStyleCellStyle" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.5 Paragraphs and Rich Formatting

The Paragraphs and Rich Formatting portion of the DrawingML framework stores text and related formatting information for a text body contained within a shape. Formatting for text within a shape can be broken down into three levels of precision, namely body, paragraph, and run formatting properties.

5.1.5.1 Body Formatting

Being the highest level of formatting available within a shape, the body properties allow for the manipulation of the text area as a whole. This means that all paragraphs and runs of text for the shape in question would be encompassed within here and, therefore, follow the text body style defined here.

[*Example*: Consider a shape that has three paragraphs within it, each with a different sized text. If this shape is resized to be smaller, then the text will no longer fit the same way within the shape. Thus, we see that to maintain visual quality the size must be changed.

Rather than try to change each of the paragraphs to a smaller font size to fit in the shape, just use a body-level format property such as the normAutofit. This will apply to all text within the shape and scale the text by a certain percentage in order to fit within the shape.

```
<p:txBody>
  <a:bodyPr>
    <a:normAutofit fontScale="20000" lnSpcReduction="20000"/>
  </a:bodyPr>
  …
  <a:p>
  …
  </a:p>
  <a:p>
  …
  </a:p>
  <a:p>
  …
  </a:p>
</p:txBody>
```

end example]

5.1.5.1.1 bodyPr (Body Properties)

This element defines the body properties for the text body within a shape.

[*Example*: Consider a shape with a text body that has some formatting properties associated with it. For the formatting of text body properties, the bodyPr element should be used as follows:

```
<p:sp>
  …
  <p:txBody>
    <a:bodyPr>
      (text body properties)
    </a:bodyPr>
  …
  </p:txBody>
</p:sp>
```

end example]

Parent Elements
lnDef (§5.1.4.1.20); rich (§5.7.2.157); spDef (§5.1.4.1.27); t (§5.9.3.8); txBody (§5.8.2.26); txBody (§5.1.2.1.40); txBody (§5.6.2.33); txBody (§4.4.1.47); txDef (§5.1.4.1.28); txPr (§5.7.2.217)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
flatTx (No text in 3D scene)	§5.1.7.8

Child Elements	Subclause
noAutofit (No AutoFit)	§5.1.5.1.2
normAutofit (Normal AutoFit)	§5.1.5.1.3
prstTxWarp (Preset Text Warp)	§5.1.11.19
scene3d (3D Scene Properties)	§5.1.4.1.26
sp3d (Apply 3D shape properties)	§5.1.7.12
spAutoFit (Shape AutoFit)	§5.1.5.1.4

Attributes	Description

anchor (Anchor)	Specifies the anchoring position of the txBody within the shape. If this attribute is omitted, then a value of t, or top is implied.
	
	[*Example*: Consider the following DrawingML:
	
	`<p:txBody>`
	`  <a:bodyPr anchor="ctr" … />`
	`  …`
	`</p:txBody>`
	
	Here the text is vertically aligned in the center of the shape within which it is contained. *end example*]
	
	The possible values for this attribute are defined by the ST_TextAnchoringType simple type (§5.1.12.60).

anchorCtr (Anchor Center)	Specifies the centering of the text box. The way it works fundamentally is to determine the smallest possible "bounds box" for the text and then to center that "bounds box" accordingly. Note that this is different than paragraph alignment, which aligns the text within the "bounds box" for the text. This flag is compatible with all of the different types of anchoring. If this attribute is omitted, then a value of 0, or off is implied.
	
	[*Example*: The text within this shape has been both vertically centered with the anchor attribute and horizontally centered with the anchorCtr attribute.
	
	`<p:txBody>`
	`  <a:bodyPr anchor="ctr" anchorCtr="1" … />`
	`  …`
	`</p:txBody>`
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

bIns (Bottom Inset)	Specifies the bottom inset of the bounding rectangle. Insets are used just as internal margins for text boxes within shapes. If this attribute is omitted, a value of 45720 or 0.05 inches is implied.

Attributes	Description

	[*Example*: Consider the following DrawingML:
	`<p:txBody>`
	`<a:bodyPr lIns="91440" tIns="91440"`
	`rIns="91440" bIns="91440" … />`
	`…`
	`</p:txBody>`
	
	The text box having the above body properties will have inset margins of 91440 or 0.1 inches on all four sides. *end example*]
	
	The possible values for this attribute are defined by the ST_Coordinate32 simple type (§5.1.12.17).

compatLnSpc (Compatible Line Spacing)	Specifies that the line spacing for this text body will be decided in a simplistic manner using the font scene. If this attribute is omitted, a value of 0 or false is implied. The possible values for this attribute are defined by the XML Schema boolean datatype.
forceAA (Force Anti-Alias)	Forces the text to be rendered anti-aliased regardless of the font size. Certain fonts may appear grainy around their edges unless they are anti-aliased. Therefore this attribute allows for the specifying of which bodies of text should always be anti-aliased and which ones should not. If this attribute is omitted, then a value of 0, or off is implied. The possible values for this attribute are defined by the XML Schema boolean datatype.

fromWordArt (From WordArt)	Specifies that text within this textbox is converted text from a WordArt object. This is more of a backwards compatibility attribute that is useful to the application from a tracking perspective. WordArt was the former way to apply text effects and therefore this attribute is useful in document conversion scenarios. If this attribute is omitted, then a value of 0, or off is implied.
	
	[*Example*: Consider the following DrawingML:
	`<p:txBody>`
	`<a:bodyPr wrap="none" fromWordArt="1" …`
	`/>`
	`…`
	`</p:txBody>`
	end example]
	Because of the presence of the fromWordArt attribute the text within this shape can be mapped back to the corresponding WordArt during document conversion. *end example*]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

horzOverflow (Text Horizontal Overflow)	Determines whether the text can flow out of the bounding box horizontally. This is used to determine what will happen in the event that the text within a shape is too large for the bounding box it is contained within. If this attribute is omitted, then a value of overflow is implied.

Attributes	Description

	[*Example*: Consider the case where we have multiply paragraphs within a shape and the second is greater in length and causes text to flow outside the shape. By applying the clip value of the horzOverflow attribute as a body property this overflowing text will now be cut off instead of extending beyond the bounds of the shape.
	
	`<p:txBody>`
	`  <a:bodyPr horzOverflow="clip" … />`
	`  …`
	`  <a:p>`
	`  …`
	`  (Some text)`
	`  …`
	`  </a:p>`
	`  <a:p>`
	`  …`
	`  (Some more text)`
	`  …`
	`  </a:p>`
	`</p:txBody>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TextHorzOverflowType simple type (§5.1.12.69).

lIns (Left Inset)	Specifies the left inset of the bounding rectangle. Insets are used just as internal margins for text boxes within shapes. If this attribute is omitted, then a value of 91440 or 0.1 inches is implied.
	
	[*Example*: Consider the following DrawingML:
	
	`<p:txBody>`
	`  <a:bodyPr lIns="91440" tIns="91440" rIns="91440"`
	`  bIns="91440" … />`
	`  …`
	`</p:txBody>`
	
	The text box having the above body properties will have inset margins of 91440 or 0.1 inches on all four sides. *end example*]
	
	The possible values for this attribute are defined by the ST_Coordinate32 simple type (§5.1.12.17).

numCol (Number of Columns)	Specifies the number of columns of text in the bounding rectangle. When applied to a text run this property takes the width of the bounding box for the text and divides it by the number of columns specified. These columns are then treated as overflow containers in that when the previous column has been filled with text the next column will act as the repository for additional text. When all columns have been filled and text still remains

Attributes	Description

	then the overflow properties set for this text body will be used and the text will be reflowed to make room for additional text. If this attribute is omitted, then a value of 1 is implied.
	
	[*Example*: Consider the case where a text area would need to be split up into four separate columns. Then simply specifying one paragraph with one run of text is enough to describe four columns of text here.
	
	`<p:txBody>`
	`  <a:bodyPr numCol="4" … />`
	`  <a:p>`
	`  <a:r>`
	`  …`
	`  (Some text)`
	`  …`
	`  </a:r>`
	`  </a:p>`
	`</p:txBody>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TextColumnCount simple type (§5.1.12.65).

rIns (Right Inset)	Specifies the right inset of the bounding rectangle. Insets are used just as internal margins for text boxes within shapes. If this attribute is omitted, then a value of 91440 or 0.1 inches is implied.
	
	[*Example*: Consider the following DrawingML:
	
	`<p:txBody>`
	`  <a:bodyPr lIns="91440" tIns="91440"`
	`  rIns="91440" bIns="91440" … />`
	`  …`
	`</p:txBody>`
	
	The text box having the above body properties will have inset margins of 91440 or 0.1 inches on all four sides. *end example*]
	
	The possible values for this attribute are defined by the ST_Coordinate32 simple type (§5.1.12.17).

rot (Rotation)	Specifies the rotation that is being applied to the text within the bounding box. If it not specified, the rotation of the accompanying shape is used. If it is specified, then this is applied independently from the shape. That is the shape can have a rotation applied in addition to the text itself having a rotation applied to it. If this attribute is omitted, then a value of 0, is implied.

Attributes	Description

	[*Example*: Consider the case where a shape has a rotation of 5400000, or 90 degrees clockwise applied to it. In addition to this, the text body itself has a rotation of -5400000, or 90 degrees counter-clockwise applied to it. Then the resulting shape would appear to be rotated but the text within it would appear as though it had not been rotated at all. The DrawingML specifying this would look like the following:
	
	`<p:sp>`
	`  <p:spPr>`
	`  <a:xfrm rot="5400000">`
	`  …`
	`  </a:xfrm>`
	`  </p:spPr>`
	`  …`
	`  <p:txBody>`
	`  <a:bodyPr rot="-5400000" … />`
	`  …`
	`  (Some text)`
	`  …`
	`  </p:txBody>`
	`</p:sp>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_Angle simple type (§5.1.12.3).

rtlCol (Columns Right-To-Left)	Specifies whether columns are used in a right-to-left or left-to-right order. The usage of this attribute only sets the column order that is used to determine which column overflow text should go to next. If this attribute is omitted, then a value of 0, or off is implied in which case text will start in the leftmost column and flow to the right. [*Note*: This attribute in no way determines the direction of text but merely the direction in which multiple columns are used. *end note*] The possible values for this attribute are defined by the XML Schema boolean datatype.
spcCol (Space Between Columns)	Specifies the space between text columns in the text area. This should only apply when there is more than 1 column present. If this attribute is omitted, then a value of 0 is implied. The possible values for this attribute are defined by the ST_PositiveCoordinate32 simple type (§5.1.12.43).
spcFirstLastPara (Paragraph Spacing)	Specifies whether the before and after paragraph spacing defined by the user is to be respected. While the spacing between paragraphs is helpful, it is additionally useful to be able to set a flag as to whether this spacing is to be followed at the edges of the text body, in other words the first and last paragraphs in the text body. More precisely since this is a text body level property it should only effect the before paragraph spacing of the first paragraph and the after paragraph spacing of the last paragraph for a given text body. If this attribute is omitted, then a value of 0, or false is implied.

Attributes	Description

	[*Example*: Consider the case where spacing has been defined between multiple paragraphs within a text body using the spcBef and spcAft paragraph spacing attributes. For this text body however the user would like to not have this followed for the edge paragraphs and thus we have the following DrawingML.
	
	`<p:txBody>`
	`  <a:bodyPr spcFirstLastPara="0" … />`
	`  …`
	`  <a:p>`
	`  <a:pPr>`
	`  <a:spcBef>`
	`  <a:spcPts val="1800"/>`
	`  </a:spcBef>`
	`  <a:spcAft>`
	`  <a:spcPts val="600"/>`
	`  </a:spcAft>`
	`  </a:pPr>`
	`  …`
	`  (Some text)`
	`  …`
	`  </a:p>`
	`  <a:p>`
	`  <a:pPr>`
	`  <a:spcBef>`
	`  <a:spcPts val="1800"/>`
	`  </a:spcBef>`
	`  <a:spcAft>`
	`  <a:spcPts val="600"/>`
	`  </a:spcAft>`
	`  </a:pPr>`
	`  …`
	`  (Some text)`
	`  …`
	`  </a:p>`
	`  …`
	`  </p:txBody>`
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

tIns (Top Inset)	Specifies the top inset of the bounding rectangle. Insets are used just as internal margins for text boxes within shapes. If this attribute is omitted, then a value of 45720 or 0.05 inches is implied. [*Example*: Consider the following DrawingML: `  <p:txBody>`

Attributes	Description
	<a:bodyPr lIns="91440" tIns="91440" rIns="91440" bIns="91440" … /> … </p:txBody> The text box having the above body properties will have inset margins of 91440 or 0.1 inches on all four sides. *end example*] The possible values for this attribute are defined by the ST_Coordinate32 simple type (§5.1.12.17).
upright (Text Upright)	Specifies whether text should remain upright, regardless of the transform applied to it and the accompanying shape transform. If this attribute is omitted, then a value of 0, or false will be implied. [*Example*: Consider text that has been rotated within the text body but has the upright flag set. <p:txBody> <a:bodyPr upright="1" rot="5400000" …/> … (Some text) … </p:txBody> This text will appear as though no transform has been applied to it. *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
vert (Vertical Text)	Determines if the text within the given text body should be displayed vertically. If this attribute is omitted, then a value of horz, or no vertical text is implied. [*Example*: Consider the case where the user needs to display text that appears vertical and has a right to left flow with respect to its columns. <p:txBody> <a:bodyPr vert="wordArtVertRtl" … /> … <a:p> … <a:t>This is</a:t> … </a:p> <a:p> … <a:t>some text.</a:t> … </a:p>

Attributes	Description
	</p:txBody> In the above sample DrawingML there are two paragraphs denoting a separation between the text otherwise which are known as either a line or paragraph break. Because wordArtVertRtl is used here this text will not only be displayed in a stacked manner flowing from top to bottom but also have the first paragraph be displayed to the right of the second. This is because it is both vertical text and right to left. *end example*] The possible values for this attribute are defined by the ST_TextVerticalType simple type (§5.1.12.83).
vertOverflow (Text Vertical Overflow)	Determines whether the text can flow out of the bounding box vertically. This is used to determine what will happen in the event that the text within a shape is too large for the bounding box it is contained within. If this attribute is omitted, then a value of overflow is implied. [*Example*: Consider the case where we have multiply paragraphs within a shape and the second causes text to flow outside the shape. By applying the clip value of the vertOverflow attribute as a body property this overflowing text will now be cut off instead of extending beyond the bounds of the shape. <p:txBody> <a:bodyPr vertOverflow="clip" … /> … <a:p> … (Some text) … </a:p> <a:p> … (Some longer text) … </a:p> </p:txBody> *end example*] The possible values for this attribute are defined by the ST_TextVertOverflowType simple type (§5.1.12.84).
wrap (Text Wrapping Type)	Specifies the wrapping options to be used for this text body. If this attribute is omitted, then a value of square is implied which will wrap the text using the bounding text box. The possible values for this attribute are defined by the ST_TextWrappingType simple type (§5.1.12.85).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextBodyProperties">
   <sequence>
      <element name="prstTxWarp" type="CT_PresetTextShape" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextAutofit" minOccurs="0" maxOccurs="1"/>
      <element name="scene3d" type="CT_Scene3D" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_Text3D" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="rot" type="ST_Angle" use="optional"/>
   <attribute name="spcFirstLastPara" type="xsd:boolean" use="optional"/>
   <attribute name="vertOverflow" type="ST_TextVertOverflowType" use="optional"/>
   <attribute name="horzOverflow" type="ST_TextHorzOverflowType" use="optional"/>
   <attribute name="vert" type="ST_TextVerticalType" use="optional"/>
   <attribute name="wrap" type="ST_TextWrappingType" use="optional"/>
   <attribute name="lIns" type="ST_Coordinate32" use="optional"/>
   <attribute name="tIns" type="ST_Coordinate32" use="optional"/>
   <attribute name="rIns" type="ST_Coordinate32" use="optional"/>
   <attribute name="bIns" type="ST_Coordinate32" use="optional"/>
   <attribute name="numCol" type="ST_TextColumnCount" use="optional"/>
   <attribute name="spcCol" type="ST_PositiveCoordinate32" use="optional"/>
   <attribute name="rtlCol" type="xsd:boolean" use="optional"/>
   <attribute name="fromWordArt" type="xsd:boolean" use="optional"/>
   <attribute name="anchor" type="ST_TextAnchoringType" use="optional"/>
   <attribute name="anchorCtr" type="xsd:boolean" use="optional"/>
   <attribute name="forceAA" type="xsd:boolean" use="optional"/>
   <attribute name="upright" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="compatLnSpc" type="xsd:boolean" use="optional"/>
</complexType>
```

5.1.5.1.2 noAutofit (No AutoFit)

This element specifies that text within the text body should not be auto-fit to the bounding box. Auto-fitting is when text within a text box is scaled in order to remain inside the text box. If this element is omitted, then noAutofit or auto-fit off is implied.

[*Example*: Consider a text box where the user wishes to have the text extend outside the bounding box. The following DrawingML would describe this.

```
<p:txBody>
  <a:bodyPr wrap="none" rtlCol="0">
    <a:noAutofit/>
  </a:bodyPr>
  <a:p>
    …
    <a:t>Some text</a:t>
    …
  </a:p>
</p:txBody>
```

end example]

Parent Elements
bodyPr (§5.1.5.1.1)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextNoAutofit"/>
```

5.1.5.1.3 normAutofit (Normal AutoFit)

This element specifies that text within the text body should be normally auto-fit to the bounding box. Auto-fitting is when text within a text box is scaled in order to remain inside the text box. If this element is omitted, then noAutofit or auto-fit off is implied.

[*Example*: Consider the situation where a user is building a diagram and needs to have the text for each shape that they are using stay within the bounds of the shape. An easy way this might be done is by using normAutofit. The following DrawingML will illustrate how this might be accomplished.

```
<p:sp>
  <p:txBody>
    <a:bodyPr rtlCol="0" anchor="ctr">
      <a:normAutofit fontScale="92000" lnSpcReduction="20000"/>
    </a:bodyPr>
    …
    <a:p>
      …
      <a:t>Diagram Object 1</a:t>
      …
    </a:p>
  </p:txBody>
</p:sp>
<p:sp>
  <p:txBody>
    <a:bodyPr rtlCol="0" anchor="ctr">
      <a:normAutofit fontScale="92000" lnSpcReduction="20000"/>
    </a:bodyPr>
    …
    <a:p>
      …
      <a:t>Diagram Object 2</a:t>
      …
    </a:p>
  </p:txBody>
</p:sp>
```

In the above example there are two shapes that have normAutofit turned on so that when the user types more text within the shape that the text will actually resize to accommodate the new data. For the application to know how and to what degree the text should be resized two attributes are set for the auto-fit resize logic. *end example*]

Parent Elements
bodyPr (§5.1.5.1.1)

Attributes	Description
fontScale (Font Scale)	Specifies the percentage of the original font size to which each run in the text body is scaled. In order to auto-fit text within a bounding box it is sometimes necessary to decrease the font size by a certain percentage. Using this attribute the font within a text box can be scaled based on the value provided. A value of 100000 is treated as 100% while a value of 1000 is consequently 1%. If this attribute is omitted, then a value of 100000 or 100% is implied. The possible values for this attribute are defined by the ST_TextFontScalePercent simple type (§5.1.12.67).
lnSpcReduction (Line Space Reduction)	Specifies the percentage amount by which the line spacing of each paragraph in the text body is reduced. The reduction is applied by subtracting it from the original line spacing value. Using this attribute the vertical spacing between the lines of text can be scaled by a percent amount. A value of 100000 is treated as 100% while a value of 1000 is consequently 1%. If this attribute is omitted, then a value of 0 or 0% is implied. [*Note*: This attribute applies only to paragraphs with percentage line spacing. *end note*] The possible values for this attribute are defined by the ST_TextSpacingPercent simple type (§5.1.12.77).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextNormalAutofit">
   <attribute name="fontScale" type="ST_TextFontScalePercent" use="optional" default="100000"/>
   <attribute name="lnSpcReduction" type="ST_TextSpacingPercent" use="optional" default="0"/>
</complexType>
```

5.1.5.1.4 spAutoFit (Shape AutoFit)

This element specifies that a shape should be auto-fit to fully contain the text described within it. Auto-fitting is when text within a shape is scaled in order to contain all the text inside. If this element is omitted, then noAutofit or auto-fit off is implied.

[*Example*: Consider the situation where a user is building a diagram and needs to have the text for each shape that they are using stay within the bounds of the shape. An easy way this might be done is by using spAutofit. The following DrawingML will illustrate how this might be accomplished.

```
<p:sp>
  <p:txBody>
    <a:bodyPr rtlCol="0" anchor="ctr">
      <a:spAutoFit/>
    </a:bodyPr>
    …
    <a:p>
      …
      <a:t>Diagram Object 1</a:t>
      …
    </a:p>
  </p:txBody>
</p:sp>
<p:sp>
  <p:txBody>
    <a:bodyPr rtlCol="0" anchor="ctr">
      <a:spAutoFit/>
    </a:bodyPr>
    …
    <a:p>
      …
      <a:t>Diagram Object 2</a:t>
      …
    </a:p>
  </p:txBody>
</p:sp>
```

In the above example there are two shapes that have spAutoFit turned on so that when the user types more text within the shape that the shape will actually resize to accommodate the new data. *end example*]

Parent Elements
bodyPr (§5.1.5.1.1)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextShapeAutofit"/>
```

5.1.5.2 Paragraph Formatting

This level of formatting allows for more granular control of text within a shape. Properties here apply to all text residing within the corresponding paragraph. This intermediate property level allows freedom to assign what would seem like lower level properties to a larger group of text. Along with this the paragraph property level also allows what would seem like larger group properties to a more granular set of text. This makes for a property level that is quite versatile in its ability to define formatting on text within a shape.

[*Example*: For instance consider the case where a paragraph of text would need to have bullets applied to it. At first one might think that this formatting must be done at the text run level as it may seem run specific. Much to the contrary this is a paragraph level property and is applied to multiple runs of text. As an example we have once again our three paragraphs with the second having bullets applied to it.

```
<a:p>
…
</a:p>
<a:p>
  <a:pPr>
    <a:buFont typeface="Wingdings"/>
    <a:buChar typeface="ü"/>
  <a:pPr>
  <a:r>
    <a:rPr lang="en-US" dirty="0" smtClean="0"/>
    <a:t>This Paragraph of Text Will Have a Bullet.</a:t>
  </a:r>
</a:p>
<a:p>
…
</a:p>
```

Here we see that the paragraph will be formatted to have character bullets for each new line of text that is encountered. In particular this paragraph will have the "ü" character applied which in the "Wingdings" font is the checkmark character. The other paragraphs will not be effected by this paragraph's bullet formatting and should have their text remain unformatted.

end example]

5.1.5.2.1 br (Text Line Break)

This element specifies the existence of a vertical line break between two runs of text within a paragraph. In addition to specifying a vertical space between two runs of text, this element may also have run properties specified via the rPr child element. This sets the formatting of text for the line break so that if text is later inserted there that a new run can be generated with the correct formatting.

[*Example*: Consider the DrawingML shown below.

```
<p:txBody>
  …
  <a:p>
    <a:r>
      …
      <a:t>Text Run 1.</a:t>
      …
    </a:r>
    <a:br/>
    <a:r>
      …
      <a:t>Text Run 2.</a:t>
      …
    </a:r>
  </a:p>
</p:txBody>
```

This paragraph will have two runs of text laid out in a vertical fashion with a line break in between them. This line break acts much like a carriage return would within a normal run of text. *end example*]

Parent Elements
p (§5.1.5.2.6)

Child Elements	Subclause
rPr (Text Run Properties)	§5.1.5.3.9

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextLineBreak">
   <sequence>
      <element name="rPr" type="CT_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.5.2.2 defPPr (Default Paragraph Style)

This element specifies the paragraph properties that are to be applied when no other paragraph properties have been specified. If this attribute is omitted, then it is left to the application to decide the set of default paragraph properties that should be applied.

[*Example*: Consider the DrawingML shown below.

```
<p:txBody>
  …
  <a:lstStyle>
    <a:defPPr>
      <a:buNone/>
    </a:defPPr>
  </a:lstStyle>
  <a:p>
    …
    <a:t>Sample Text</a:t>
    …
  </a:p>
</p:txBody>
```

The above paragraph will follow the properties described in defPPr if no overriding properties are specified within the pPr element. *end example*]

Parent Elements
bodyStyle (§4.4.1.5); defaultTextStyle (§4.3.1.7); lstStyle (§5.1.5.4.12); notesStyle (§4.4.1.25); otherStyle (§4.4.1.32); titleStyle (§4.4.1.45)

Child Elements	Subclause
buAutoNum (Auto-Numbered Bullet)	§5.1.5.4.1
buBlip (Picture Bullet)	§5.1.5.4.2
buChar (Character Bullet)	§5.1.5.4.3
buClr (Color Specified)	§5.1.5.4.4
buClrTx (Follow Text)	§5.1.5.4.5
buFont (Specified)	§5.1.5.4.6
buFontTx (Follow text)	§5.1.5.4.7
buNone (No Bullet)	§5.1.5.4.8
buSzPct (Bullet Size Percentage)	§5.1.5.4.9
buSzPts (Bullet Size Points)	§5.1.5.4.10
buSzTx (Bullet Size Follows Text)	§5.1.5.4.11
defRPr (Default Text Run Properties)	§5.1.5.3.2
extLst (Extension List)	§5.1.2.1.15
lnSpc (Line Spacing)	§5.1.5.2.5
spcAft (Space After)	§5.1.5.2.8
spcBef (Space Before)	§5.1.5.2.9
tabLst (Tab List)	§5.1.5.2.13

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>algn (Alignment)</td><td>

Specifies the alignment that is to be applied to the paragraph. Possible values for this include left, right, centered, justified and distributed. If this attribute is omitted, then a value of left is implied.

Sample text Sample text

[*Example*: Consider the case where the user wishes to have two columns of text that have a justified alignment, much like text within a book. The following DrawingML could describe this.

```
<p:txBody>
  <a:bodyPr numCol="2" spcCol="914400"…/>
    <a:normAutofit/>
    </a:bodyPr>
  …
  <a:p>
    <a:pPr marL="0" algn="just">
      <a:buNone/>
    </a:pPr>
    …
    <a:t>Sample Text …</a:t>
    …
  </a:p>
</p:txBody>
```

end example]

The possible values for this attribute are defined by the ST_TextAlignType simple type (§5.1.12.59).

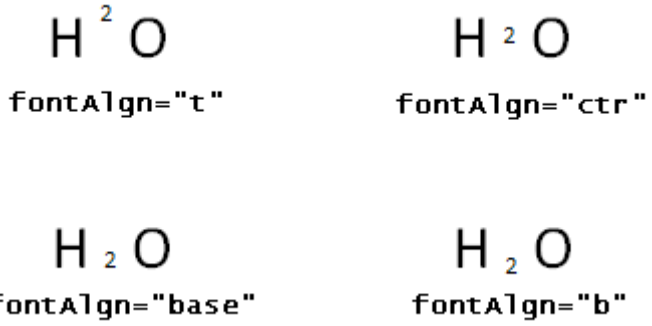
</td></tr>
<tr><td>defTabSz (Default Tab Size)</td><td>Specifies the default size for a tab character within this paragraph. This attribute should be used to describe the spacing of tabs within the paragraph instead of a leading indentation tab. For indentation tabs there are the marL and indent attributes to assist with this.</td></tr>
</table>

Attributes	Description

	[*Example*: Consider the case where a paragraph contains numerous tabs that need to be of a specific size. The following DrawingML would describe this.
	
	`<p:txBody>`
	`  …`
	`  <a:p>`
	`  <a:pPr defTabSz="376300" …/>`
	`  …`
	`  <a:t>Sample Text …</a:t>`
	`  …`
	`  </a:p>`
	`</p:txBody>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_Coordinate32 simple type (§5.1.12.17).

eaLnBrk (East Asian Line Break)	Specifies whether an East Asian word can be broken in half and wrapped onto the next line without a hyphen being added. To determine whether an East Asian word can be broken the presentation application would use the kinsoku settings here. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable East Asian words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.
	
	[*Example*: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in whole on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.
	
	`<p:txBody>`
	`  …`
	`  <a:p>`
	`  <a:pPr eaLnBrk="0" …/>`
	`  …`
	`  <a:t>Sample text (Long word)</a:t>`
	`  …`
	`  </a:p>`
	`</p:txBody>`
	
	Sample text Sample text Sample text supercalifr
	agilisticexpialidocious

Attributes	Description
	Sample text Sample text Sample text supercalifragilisticexpialidocious *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.

fontAlgn (Font Alignment)	Determines where vertically on a line of text the actual words are positioned. This deals with vertical placement of the characters with respect to the baselines. For instance having text anchored to the top baseline, anchored to the bottom baseline, centered in between, etc. To understand this attribute and it's use it is helpful to understand what baselines are. A diagram describing these different cases is shown below. If this attribute is omitted, then a value of base is implied.
	
	[*Example*: Consider the case where the user wishes to represent the chemical compound of a water molecule. For this they will need to make sure the H, the 2, and the O are all in the correct position and are of the correct size. The results below can be achieved through the DrawingML shown below.
	
	 
	
	`<a:txtBody>`
	`…`
	`<a:pPr fontAlgn="b" …/>`
	`…`
	`<a:r>`
	`<a:rPr …/>`
	`<a:t>H </a:t>`
	`</a:r>`
	`<a:r>`
	`<a:rPr sz="1200" …/>`
	`<a:t>2</a:t>`
	`</a:r>`
	`<a:r>`
	`<a:rPr …/>`
	`<a:t>O</a:t>`
	`</a:r>`

Attributes	Description
	… </p:txBody> *end example*] The possible values for this attribute are defined by the ST_TextFontAlignType simple type (§5.1.12.66).
hangingPunct (Hanging Punctuation)	Specifies whether punctuation is to be forcefully laid out on a line of text or put on a different line of text. That is, if there is punctuation at the end of a run of text that should be carried over to a separate line does it actually get carried over. A true value will allow for hanging punctuation forcing the punctuation to not be carried over and a value of false will allow the punctuation to be carried onto the next text line. If this attribute is omitted, then a value of 0, or false is implied. The possible values for this attribute are defined by the XML Schema boolean datatype.

indent (Indent)	Specifies the indent size that will be applied to the first line of text in the paragraph. An indentation of 0 will be considered to be at the same location as marL attribute. If this attribute is omitted, then a value of -342900 is implied.
	
	Here is some Sample text
	
	[*Example*: Consider the scenario where the user now wanted to add a paragraph indentation to the first line of text in their two column format book.
	
	<p:txBody>
	  <a:bodyPr numCol="2" spcCol="914400"…/>
	    <a:normAutofit/>
	    </a:bodyPr>
	  …
	  <a:p>
	    <a:pPr marL="0" indent="571500" algn="just">
	      <a:buNone/>
	    </a:pPr>
	    …

Attributes	Description
	<a:t>Here is some…</a:t> … </a:p> </p:txBody> By adding the indent attribute the user has effectively added a first line indent to this paragraph of text. *end example*] The possible values for this attribute are defined by the ST_TextIndent simple type (§5.1.12.70).
latinLnBrk (Latin Line Break)	Specifies whether a Latin word can be broken in half and wrapped onto the next line without a hyphen being added. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable Latin words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied. [*Example*: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in whole on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows. <p:txBody> … <a:p> <a:pPr latinLnBrk="0" …/> … <a:t>Sample text (Long word)</a:t> … </a:p> </p:txBody> Sample text Sample text Sample text supercalifr agilisticexpialidocious Sample text Sample text Sample text supercalifragilisticexpialidocious *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
lvl (Level)	Specifies the particular level text properties that this paragraph will follow. The value for this attribute is numerical and formats the text according to the corresponding level paragraph properties that are listed within the lstStyle element. Since there are nine

Attributes	Description

	separate level properties defined, this tag will have an effective range of 0-8 = 9 available values.
	
	[*Example*: Consider the following DrawingML. This would specify that this paragraph should follow the lvl2pPr formatting style because once again lvl="1" is considered to be level 2.
	
	`<p:txBody>`
	`…`
	`<a:p>`
	`<a:pPr lvl="1" …/>`
	`…`
	`<a:t>Sample text</a:t>`
	`…`
	`</a:p>`
	`</p:txBody>`
	
	end example]
	
	[*Note*: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The rule here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. *end note*]
	
	The possible values for this attribute are defined by the ST_TextIndentLevelType simple type (§5.1.12.71).

marL (Left Margin)	Specifies the left margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marL attributes are additive with respect to the text position. If this attribute is omitted, then a value of 347663 is implied. The possible values for this attribute are defined by the ST_TextMargin simple type (§5.1.12.73).
marR (Right Margin)	Specifies the right margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marR attributes are additive with respect to the text position. If this attribute is omitted, then a value of 0 is implied. The possible values for this attribute are defined by the ST_TextMargin simple type (§5.1.12.73).
rtl (Right To Left)	Specifies whether the text is right-to-left or left-to-right in its flow direction. If this attribute is omitted, then a value of 0, or left-to-right is implied.

Attributes	Description

	[*Example*: Consider the scenario where the user wanted text to flow from right to left. That is within the bounding text box the first word would be right aligned with each additional word being written to the left of the previous while continuing to flow top to bottom. The DrawingML to describe this might be as follows.
	
	`<p:txBody>`
	`…`
	`<a:p>`
	`<a:pPr rtl="1" …/>`
	`…`
	`<a:t>Sample text…</a:t>`
	`…`
	`</a:p>`
	`</p:txBody>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextParagraphProperties">
   <sequence>
      <element name="lnSpc" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <element name="spcBef" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <element name="spcAft" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletColor" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletSize" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletTypeface" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBullet" minOccurs="0" maxOccurs="1"/>
      <element name="tabLst" type="CT_TextTabStopList" minOccurs="0" maxOccurs="1"/>
      <element name="defRPr" type="CT_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="marL" type="ST_TextMargin" use="optional"/>
   <attribute name="marR" type="ST_TextMargin" use="optional"/>
   <attribute name="lvl" type="ST_TextIndentLevelType" use="optional"/>
   <attribute name="indent" type="ST_TextIndent" use="optional"/>
   <attribute name="algn" type="ST_TextAlignType" use="optional"/>
   <attribute name="defTabSz" type="ST_Coordinate32" use="optional"/>
   <attribute name="rtl" type="xsd:boolean" use="optional"/>
   <attribute name="eaLnBrk" type="xsd:boolean" use="optional"/>
   <attribute name="fontAlgn" type="ST_TextFontAlignType" use="optional"/>
   <attribute name="latinLnBrk" type="xsd:boolean" use="optional"/>
   <attribute name="hangingPunct" type="xsd:boolean" use="optional"/>
</complexType>
```

5.1.5.2.3 endParaRPr (End Paragraph Run Properties)

This element specifies the text run properties that are to be used if another run is inserted after the last run specified. This effectively saves the run property state so that it may be applied when the user enters additional text. If this element is omitted, then the application may determine which default properties to apply. It is recommended that this element be specified at the end of the list of text runs within the paragraph so that an orderly list is maintained.

Parent Elements
p (§5.1.5.2.6)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
cs (Complex Script Font)	§5.1.5.3.1
ea (East Asian Font)	§5.1.5.3.3
effectDag (Effect Container)	§5.1.10.25
effectLst (Effect Container)	§5.1.10.26
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35
highlight (Highlight Color)	§5.1.5.3.4
hlinkClick (Click Hyperlink)	§5.1.5.3.5
hlinkMouseOver (Mouse-Over Hyperlink)	§5.1.5.3.6
latin (Latin Font)	§5.1.5.3.7
ln (Outline)	§5.1.2.1.24
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
solidFill (Solid Fill)	§5.1.10.54
sym (Symbol Font)	§5.1.5.3.10
uFill (Underline Fill)	§5.1.5.3.12
uFillTx (Underline Fill Properties Follow Text)	§5.1.5.3.13
uLn (Underline Stroke)	§5.1.5.3.14
uLnTx (Underline Follows Text)	§5.1.5.3.15

Attributes	Description
altLang (Alternative Language)	Specifies the alternate language to use when the generating application is displaying the user interface controls. If this attribute is omitted, than the lang attribute will be used

Attributes	Description
	here. The possible values for this attribute are defined by the ST_TextLanguageID simple type (§5.1.12.72).

b (Bold)	Specifies whether a run of text will be formatted as bold text. If this attribute is omitted, than a value of 0, or false is assumed.
	[*Example*: Consider the DrawingML shown below.
	`<a:p>`
	`…`
	`<a:rPr b="1"/>`
	`…`
	`<a:t>Some Text</a:t>`
	`…`
	`</a:p>`
	The above run of text will be formatted as bold text. *end example*]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

baseline (Baseline)	Specifies the baseline for both the superscript and subscript fonts. The size is specified using a percentage where 1000 is equal to 1 percent of the font size and 100000 is equal to 100 percent font of the font size. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
bmk (Bookmark Link Target)	Specifies the link target name that is used to reference to the proper link properties in a custom XML part within the document. The possible values for this attribute are defined by the XML Schema string datatype.
cap (Capitalization)	Specifies the capitalization that is to be applied to the text run. This is a render-only modification and does not affect the actual characters stored in the text run. This attribute is also distinct from the toggle function where the actual characters stored in the text run are changed. The possible values for this attribute are defined by the ST_TextCapsType simple type (§5.1.12.64).
dirty (Dirty)	Specifies that the content of a text run has changed since the proofing tools have last been run. Effectively this flags text that must be checked again by the generating application for mistakes such as spelling, grammar, etc. The possible values for this attribute are defined by the XML Schema boolean datatype.
err (Spelling Error)	Specifies that when this run of text was checked for spelling, grammar, etc. that a mistake was indeed found. This allows the generating application to effectively save the state of the mistakes within the document instead of having to perform a full pass check upon opening the document.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema boolean datatype.

i (Italics)	Specifies whether a run of text will be formatted as italic text. If this attribute is omitted, than a value of 0, or false is assumed.
	
	[*Example*: Consider the DrawingML shown below.
	
	`<a:p>`
	`…`
	`<a:rPr i="1"/>`
	`…`
	`<a:t>Some Text</a:t>`
	`…`
	`</a:p>`
	
	The above run of text will be formatted as italic text. *end example*]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

kern (Kerning)	Specifies the minimum font size at which character kerning will occur for this text run. Whole points are specified in increments of 100 starting with 100 being a point size of 1. For instance a font point size of 12 would be 1200 and a font point size of 12.5 would be 1250. If this attribute is omitted, than kerning will occur for all font sizes down to a 0 point font. The possible values for this attribute are defined by the ST_TextNonNegativePoint simple type (§5.1.12.74).
kumimoji (Kumimoji)	Specifies whether the numbers contained within vertical text will continue vertically with the text or whether they are to be displayed horizontally while the surrounding characters continue in a vertical fashion. If this attribute is omitted, than a value of 0, or false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
lang (Language ID)	Specifies the language to be used when the generating application is displaying the user interface controls. If this attribute is omitted, than the generating application may select a language of its choice. The possible values for this attribute are defined by the ST_TextLanguageID simple type (§5.1.12.72).
noProof (No Proofing)	Specifies that a run of text has been selected by the user to not be checked for mistakes. Therefore if there are spelling, grammar, etc mistakes within this text the generating application should ignore them. The possible values for this attribute are defined by the XML Schema boolean datatype.
normalizeH (Normalize Heights)	Specifies the normalization of height that is to be applied to the text run. This is a render-only modification and does not affect the actual characters stored in the text run. This

Attributes	Description
	attribute is also distinct from the toggle function where the actual characters stored in the text run are changed. If this attribute is omitted, than a value of 0, or false will be assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
smtClean (SmartTag Clean)	Specifies whether or not a text run has been checked for smart tags. This attribute acts much like the dirty attribute dose for the checking of spelling, grammar, etc. A value of true here indicates to the generating application that this text run should be checked for smart tags. If this attribute is omitted, than a value of 0, or false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
smtId (SmartTag ID)	Specifies the reference id for the smart tag. This id corresponds to a link within this slides relationship file. The following of this link within the relationship file will result in the actual smart tag information for this piece of text. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
spc (Spacing)	Specifies the spacing between characters within a text run. This spacing is specified numerically and should be consistently applied across the entire run of text by the generating application. Whole points are specified in increments of 100 starting with 100 being a point size of 1. For instance a font point size of 12 would be 1200 and a font point size of 12.5 would be 1250. If this attribute is omitted than a value of 0 or no adjustment is assumed. The possible values for this attribute are defined by the ST_TextPoint simple type (§5.1.12.75).

strike (Strikethrough)	Specifies whether a run of text will be formatted as strikethrough text. If this attribute is omitted, than no strikethrough is assumed.
	
	[*Example*: Consider the DrawingML shown below.
	
	`<a:p>`
	`…`
	`<a:rPr strike="sngStrike"/>`
	`…`
	`<a:t>Some Text</a:t>`
	`…`
	`</a:p>`
	
	The above run of text will be formatted as single strikethrough text. *end example*]
	
	The possible values for this attribute are defined by the ST_TextStrikeType simple type (§5.1.12.79).

sz (Font Size)	Specifies the size of text within a text run. Whole points are specified in increments of 100 starting with 100 being a point size of 1. For instance a font point size of 12 would be

Attributes	Description

	1200 and a font point size of 12.5 would be 1250. If this attribute is omitted, than the value in defRPr should be used.
	
	[*Example*: Consider the DrawingML shown below.
	
	`<a:p>`
	`  …`
	`  <a:rPr sz="1200"/>`
	`  …`
	`  <a:t>Some Text</a:t>`
	`  …`
	`</a:p>`
	
	The above run of text will be formatted with a 12 point text size. *end example*]
	
	The possible values for this attribute are defined by the ST_TextFontSize simple type (§5.1.12.68).

u (Underline)	Specifies whether a run of text will be formatted as underlined text. If this attribute is omitted, than no underline is assumed.
	
	[*Example*: Consider the DrawingML shown below.
	
	`<a:p>`
	`  …`
	`  <a:rPr u="sng"/>`
	`  …`
	`  <a:t>Some Text</a:t>`
	`  …`
	`</a:p>`
	
	The above run of text will be formatted as single underline text. *end example*]
	
	The possible values for this attribute are defined by the ST_TextUnderlineType simple type (§5.1.12.82).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextCharacterProperties">
   <sequence>
      <element name="ln" type="CT_LineProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_FillProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_EffectProperties" minOccurs="0" maxOccurs="1"/>
      <element name="highlight" type="CT_Color" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextUnderlineLine" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextUnderlineFill" minOccurs="0" maxOccurs="1"/>
      <element name="latin" type="CT_TextFont" minOccurs="0" maxOccurs="1"/>
      <element name="ea" type="CT_TextFont" minOccurs="0" maxOccurs="1"/>
      <element name="cs" type="CT_TextFont" minOccurs="0" maxOccurs="1"/>
      <element name="sym" type="CT_TextFont" minOccurs="0" maxOccurs="1"/>
      <element name="hlinkClick" type="CT_Hyperlink" minOccurs="0" maxOccurs="1"/>
      <element name="hlinkMouseOver" type="CT_Hyperlink" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="kumimoji" type="xsd:boolean" use="optional"/>
   <attribute name="lang" type="ST_TextLanguageID" use="optional"/>
   <attribute name="altLang" type="ST_TextLanguageID" use="optional"/>
   <attribute name="sz" type="ST_TextFontSize" use="optional"/>
   <attribute name="b" type="xsd:boolean" use="optional"/>
   <attribute name="i" type="xsd:boolean" use="optional"/>
   <attribute name="u" type="ST_TextUnderlineType" use="optional"/>
   <attribute name="strike" type="ST_TextStrikeType" use="optional"/>
   <attribute name="kern" type="ST_TextNonNegativePoint" use="optional"/>
   <attribute name="cap" type="ST_TextCapsType" use="optional"/>
   <attribute name="spc" type="ST_TextPoint" use="optional"/>
   <attribute name="normalizeH" type="xsd:boolean" use="optional"/>
   <attribute name="baseline" type="ST_Percentage" use="optional"/>
   <attribute name="noProof" type="xsd:boolean" use="optional"/>
   <attribute name="dirty" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="err" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="smtClean" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="smtId" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="bmk" type="xsd:string" use="optional"/>
</complexType>
```

5.1.5.2.4 fld (Text Field)

This element specifies a text field which contains generated text that the application should update periodically. Each piece of text when it is generated is given a unique identification number that is used to refer to a specific field. At the time of creation the text field is also specified to be of a certain type which indicates the type of text that should be used to update this field. This update type is used so that all applications that did not create this text field may still know what type of text it should be updated with. Thus the new application can then attach an update type to the text field id for continual updating.

[*Example*: Consider a slide within a presentation that needs to have the slide number placed on the slide. The following DrawingML may be used to describe such a situation.

```
<p:txBody>
  <a:bodyPr/>
    <a:lstStyle/>
    <a:p>
      <a:fld id="{424CEEAC-8F67-4238-9622-1B74DC6E8318}" type="slidenum">
      <a:rPr lang="en-US" smtClean="0"/>
      <a:pPr/>
      <a:t>3</a:t>
    </a:fld>
    <a:endParaRPr lang="en-US"/>
  </a:p>
</p:txBody>
```

end example]

Parent Elements
p (§5.1.5.2.6)

Child Elements	Subclause
pPr (Text Paragraph Properties)	§5.1.5.2.7
rPr (Text Run Properties)	§5.1.5.3.9
t (Text String)	§5.1.5.3.11

Attributes	Description
id (Field ID)	Specifies the unique to this document, host specified token that is used to identify the field. This token is generated when the text field is created and persists in the file as the same token until the text field is removed. Any application should check the document for conflicting tokens before assigning a new token to a text field. The possible values for this attribute are defined by the ST_Guid simple type (§5.1.12.27).
type (Field Type)	Specifies the type of update text that should be used within this text field. This is needed in addition to the text field id because a new application that did not create this document must be able to know what update should be assigned to a specific text field id. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextField">
   <sequence>
      <element name="rPr" type="CT_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>
      <element name="pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="t" type="xsd:string" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="id" type="ST_Guid" use="required"/>
   <attribute name="type" type="xsd:string" use="optional"/>
</complexType>
```

5.1.5.2.5 lnSpc (Line Spacing)

This element specifies the vertical line spacing that is to be used within a paragraph. This may be specified in two different ways, percentage spacing and font point spacing. If this element is omitted then the spacing between two lines of text should be determined by the point size of the largest piece of text within a line.

[*Example*: Consider the DrawingML shown below.

```
<p:txBody>
  <a:p>
    <a:pPr>
      <a:lnSpc>
        <a:spcPct val="200000"/>
      </a:lnSpc>
    </a:pPr>
    <a:r>
      <a:rPr lang="en-US" dirty="0" smtClean="0"/>
      <a:t>Some</a:t>
    </a:r>
    <a:br>
      <a:rPr lang="en-US" smtClean="0"/>
    </a:br>
    <a:r>
      <a:rPr lang="en-US" dirty="0" smtClean="0"/>
      <a:t>Text</a:t>
    </a:r>
  </a:p>
</p:txBody>
```

This paragraph will have two lines of text that will have percentage based vertical spacing. This kind of spacing should change based on the size of the text involved as its size is calculated as a percentage of this. *end example*]

Parent Elements
defPPr (§5.1.5.2.2); lvl1pPr (§5.1.5.4.13); lvl2pPr (§5.1.5.4.14); lvl3pPr (§5.1.5.4.15); lvl4pPr (§5.1.5.4.16);

Parent Elements
lvl5pPr (§5.1.5.4.17); lvl6pPr (§5.1.5.4.18); lvl7pPr (§5.1.5.4.19); lvl8pPr (§5.1.5.4.20); lvl9pPr (§5.1.5.4.21); pPr (§5.1.5.2.7)

Child Elements	Subclause
spcPct (Spacing Percent)	§5.1.5.2.10
spcPts (Spacing Points)	§5.1.5.2.11

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextSpacing">
   <choice>
      <element name="spcPct" type="CT_TextSpacingPercent"/>
      <element name="spcPts" type="CT_TextSpacingPoint"/>
   </choice>
</complexType>
```

5.1.5.2.6 p (Text Paragraphs)

This element specifies the presence of a paragraph of text within the containing text body. The paragraph is the highest level text separation mechanism within a text body. A paragraph may contain text paragraph properties associated with the paragraph. If no properties are listed then properties specified in the defPPr element are used.

[*Example*: Consider the case where the user would like to describe a text body that will contain two paragraphs. The requirement for these paragraphs is that one be right aligned and the other left aligned. The following DrawingML would specify a text body such as this.

```
<p:txBody>
    …
    <a:p>
    <a:pPr algn="r">
    </a:pPr>
    …
    <a:t>Some text</a:t>
    …
  </a:p>
  <a:p>
    <a:pPr algn="l">
    </a:pPr>
    …
    <a:t>Some text</a:t>
    …
  </a:p>
</p:txBody>
```

end example]

Parent Elements
rich (§5.7.2.157); t (§5.9.3.8); txBody (§5.8.2.26); txBody (§5.1.2.1.40); txBody (§5.6.2.33); txBody (§4.4.1.47); txPr (§5.7.2.217)

Child Elements	Subclause
br (Text Line Break)	§5.1.5.2.1
endParaRPr (End Paragraph Run Properties)	§5.1.5.2.3
fld (Text Field)	§5.1.5.2.4
pPr (Text Paragraph Properties)	§5.1.5.2.7
r (Text Run)	§5.1.5.3.8

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextParagraph">
   <sequence>
      <element name="pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextRun" minOccurs="0" maxOccurs="unbounded"/>
      <element name="endParaRPr" type="CT_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.5.2.7 pPr (Text Paragraph Properties)

This element contains all paragraph level text properties for the containing paragraph. These paragraph properties should override any and all conflicting properties that are associated with the paragraph in question.

[*Example*: Consider the DrawingML shown below.

```
<a:p>
  <a:pPr marL="0" algn="ctr">
    <a:buNone/>
  </a:pPr>
  …
  <a:t>Some Text</a:t>
  …
</a:p>
```

The paragraph described above will be formatting with a left margin of 0 and will have all of text runs contained within it centered about the horizontal median of the bounding box for the text body. *end example*]

[*Note*: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The rule here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr

elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. *end note*]

Parent Elements
fld (§5.1.5.2.4); p (§5.1.5.2.6)

Child Elements	Subclause
buAutoNum (Auto-Numbered Bullet)	§5.1.5.4.1
buBlip (Picture Bullet)	§5.1.5.4.2
buChar (Character Bullet)	§5.1.5.4.3
buClr (Color Specified)	§5.1.5.4.4
buClrTx (Follow Text)	§5.1.5.4.5
buFont (Specified)	§5.1.5.4.6
buFontTx (Follow text)	§5.1.5.4.7
buNone (No Bullet)	§5.1.5.4.8
buSzPct (Bullet Size Percentage)	§5.1.5.4.9
buSzPts (Bullet Size Points)	§5.1.5.4.10
buSzTx (Bullet Size Follows Text)	§5.1.5.4.11
defRPr (Default Text Run Properties)	§5.1.5.3.2
extLst (Extension List)	§5.1.2.1.15
lnSpc (Line Spacing)	§5.1.5.2.5
spcAft (Space After)	§5.1.5.2.8
spcBef (Space Before)	§5.1.5.2.9
tabLst (Tab List)	§5.1.5.2.13

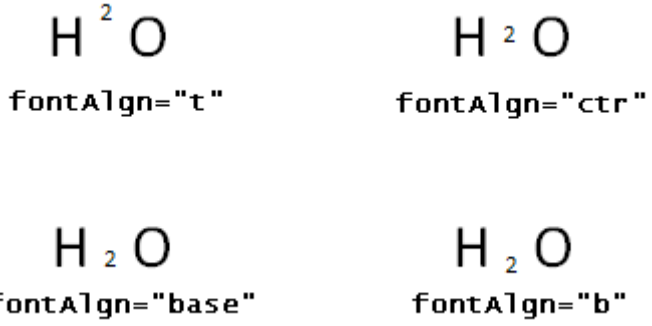
Attributes	Description
algn (Alignment)	Specifies the alignment that is to be applied to the paragraph. Possible values for this include left, right, centered, justified and distributed. If this attribute is omitted, then a value of left is implied.

Attributes	Description
	Sample text [*Example*: Consider the case where the user wishes to have two columns of text that have a justified alignment, much like text within a book. The following DrawingML could describe this. `<p:txBody>` `  <a:bodyPr numCol="2" spcCol="914400"…/>` `    <a:normAutofit/>` `    </a:bodyPr>` `  …` `  <a:p>` `    <a:pPr marL="0" algn="just">` `      <a:buNone/>` `    </a:pPr>` `    …` `    <a:t>Sample Text …</a:t>` `    …` `  </a:p>` `</p:txBody>` *end example*] The possible values for this attribute are defined by the ST_TextAlignType simple type (§5.1.12.59).

defTabSz (Default Tab Size)	Specifies the default size for a tab character within this paragraph. This attribute should be used to describe the spacing of tabs within the paragraph instead of a leading indentation tab. For indentation tabs there are the marL and indent attributes to assist with this.
	
	[*Example*: Consider the case where a paragraph contains numerous tabs that need to be of a specific size. The following DrawingML would describe this.
	
	`<p:txBody>`
	`  …`

Attributes	Description
	<a:p> <a:pPr defTabSz="376300" …/> … <a:t>Sample Text …</a:t> … </a:p> </p:txBody> *end example*] The possible values for this attribute are defined by the ST_Coordinate32 simple type (§5.1.12.17).
eaLnBrk (East Asian Line Break)	Specifies whether an East Asian word can be broken in half and wrapped onto the next line without a hyphen being added. To determine whether an East Asian word can be broken the presentation application would use the kinsoku settings here. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable East Asian words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied. [*Example*: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in whole on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows. <p:txBody> … <a:p> <a:pPr eaLnBrk="0" …/> … <a:t>Sample text (Long word)</a:t> … </a:p> </p:txBody> Sample text Sample text Sample text supercalifr agilisticexpialidocious Sample text Sample text Sample text supercalifragilisticexpialidocious *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description

fontAlgn (Font Alignment)	Determines where vertically on a line of text the actual words are positioned. This deals with vertical placement of the characters with respect to the baselines. For instance having text anchored to the top baseline, anchored to the bottom baseline, centered in between, etc. To understand this attribute and it's use it is helpful to understand what baselines are. A diagram describing these different cases is shown below. If this attribute is omitted, then a value of base is implied.
	
	[*Example*: Consider the case where the user wishes to represent the chemical compound of a water molecule. For this they will need to make sure the H, the 2, and the O are all in the correct position and are of the correct size. The results below can be achieved through the DrawingML shown below.
	
	 
	
	`<a:txtBody>`
	`…`
	`<a:pPr fontAlgn="b" …/>`
	`…`
	`<a:r>`
	`<a:rPr …/>`
	`<a:t>H </a:t>`
	`</a:r>`
	`<a:r>`
	`<a:rPr sz="1200" …/>`
	`<a:t>2</a:t>`
	`</a:r>`
	`<a:r>`
	`<a:rPr …/>`
	`<a:t>O</a:t>`
	`</a:r>`
	`…`
	`</p:txBody>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TextFontAlignType simple type (§5.1.12.66).

Attributes	Description
hangingPunct (Hanging Punctuation)	Specifies whether punctuation is to be forcefully laid out on a line of text or put on a different line of text. That is, if there is punctuation at the end of a run of text that should be carried over to a separate line does it actually get carried over. A true value will allow for hanging punctuation forcing the punctuation to not be carried over and a value of false will allow the punctuation to be carried onto the next text line. If this attribute is omitted, then a value of 0, or false is implied. The possible values for this attribute are defined by the XML Schema boolean datatype.

indent (Indent)	Specifies the indent size that will be applied to the first line of text in the paragraph. An indentation of 0 will be considered to be at the same location as marL attribute. If this attribute is omitted, then a value of -342900 is implied.
	
	Here is some text Sample text
	
	[*Example*: Consider the scenario where the user now wanted to add a paragraph indentation to the first line of text in their two column format book.
	
	`<p:txBody>`
	`<a:bodyPr numCol="2" spcCol="914400"…/>`
	`<a:normAutofit/>`
	`</a:bodyPr>`
	`…`
	`<a:p>`
	`<a:pPr marL="0" indent="571500" algn="just">`
	`<a:buNone/>`
	`</a:pPr>`
	`…`
	`<a:t>Here is some…</a:t>`
	`…`
	`</a:p>`
	`</p:txBody>`
	
	By adding the indent attribute the user has effectively added a first line indent to this paragraph of text. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_TextIndent simple type (§5.1.12.70).

latinLnBrk (Latin Line Break)	Specifies whether a Latin word can be broken in half and wrapped onto the next line without a hyphen being added. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable Latin words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.
	
	[*Example*: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in whole on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.
	
	`<p:txBody>`
	`…`
	`<a:p>`
	`<a:pPr latinLnBrk="0" …/>`
	`…`
	`<a:t>Sample text (Long word)</a:t>`
	`…`
	`</a:p>`
	`</p:txBody>`
	
	Sample text Sample text Sample text supercalifr agilisticexpialidocious
	
	Sample text Sample text Sample text supercalifragilisticexpialidocious
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

lvl (Level)	Specifies the particular level text properties that this paragraph will follow. The value for this attribute is numerical and formats the text according to the corresponding level paragraph properties that are listed within the lstStyle element. Since there are nine separate level properties defined, this tag will have an effective range of 0-8 = 9 available values. [*Example*: Consider the following DrawingML. This would specify that this paragraph should follow the lvl2pPr formatting style because once again lvl="1" is considered to be level 2. `<p:txBody>`

Attributes	Description
	… <a:p> <a:pPr lvl="1" …/> … <a:t>Sample text</a:t> … </a:p> </p:txBody> *end example*] [*Note*: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The rule here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. *end note*] The possible values for this attribute are defined by the ST_TextIndentLevelType simple type (§5.1.12.71).
marL (Left Margin)	Specifies the left margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marL attributes are additive with respect to the text position. If this attribute is omitted, then a value of 347663 is implied. The possible values for this attribute are defined by the ST_TextMargin simple type (§5.1.12.73).
marR (Right Margin)	Specifies the right margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marR attributes are additive with respect to the text position. If this attribute is omitted, then a value of 0 is implied. The possible values for this attribute are defined by the ST_TextMargin simple type (§5.1.12.73).
rtl (Right To Left)	Specifies whether the text is right-to-left or left-to-right in its flow direction. If this attribute is omitted, then a value of 0, or left-to-right is implied. [*Example*: Consider the scenario where the user wanted text to flow from right to left. That is within the bounding text box the first word would be right aligned with each additional word being written to the left of the previous while continuing to flow top to bottom. The DrawingML to describe this might be as follows. <p:txBody> … <a:p>

Attributes	Description
	<a:pPr rtl="1" …/> … <a:t>Sample text…</a:t> … </a:p> </p:txBody> *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextParagraphProperties">
   <sequence>
      <element name="lnSpc" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <element name="spcBef" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <element name="spcAft" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletColor" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletSize" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletTypeface" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBullet" minOccurs="0" maxOccurs="1"/>
      <element name="tabLst" type="CT_TextTabStopList" minOccurs="0" maxOccurs="1"/>
      <element name="defRPr" type="CT_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="marL" type="ST_TextMargin" use="optional"/>
   <attribute name="marR" type="ST_TextMargin" use="optional"/>
   <attribute name="lvl" type="ST_TextIndentLevelType" use="optional"/>
   <attribute name="indent" type="ST_TextIndent" use="optional"/>
   <attribute name="algn" type="ST_TextAlignType" use="optional"/>
   <attribute name="defTabSz" type="ST_Coordinate32" use="optional"/>
   <attribute name="rtl" type="xsd:boolean" use="optional"/>
   <attribute name="eaLnBrk" type="xsd:boolean" use="optional"/>
   <attribute name="fontAlign" type="ST_TextFontAlignType" use="optional"/>
   <attribute name="latinLnBrk" type="xsd:boolean" use="optional"/>
   <attribute name="hangingPunct" type="xsd:boolean" use="optional"/>
</complexType>
```

5.1.5.2.8 spcAft (Space After)

This element specifies the amount of vertical white space that will be present after a paragraph. This space is specified in either percentage or points via the child elements spcPct and spcPts.

[*Example*: Consider the DrawingML shown below.

```
<p:txBody>
  …
  <a:p>
    <a:pPr …>
      <a:spcBef>
        <a:spcPts val="1800"/>
      </a:spcBef>
      <a:spcAft>
        <a:spcPts val="600"/>
      </a:spcAft>
    </a:pPr>
    …
    <a:t>Sample Text</a:t>
    …
  </a:p>
  …
</p:txBody>
```

The above paragraph of text will be formatted to have a spacing both before and after the paragraph text. The spacing before will be a size of 18 points, or value=1800 and the spacing after will be a size of 6 points, or value=600. *end example*]

Parent Elements
defPPr (§5.1.5.2.2); lvl1pPr (§5.1.5.4.13); lvl2pPr (§5.1.5.4.14); lvl3pPr (§5.1.5.4.15); lvl4pPr (§5.1.5.4.16); lvl5pPr (§5.1.5.4.17); lvl6pPr (§5.1.5.4.18); lvl7pPr (§5.1.5.4.19); lvl8pPr (§5.1.5.4.20); lvl9pPr (§5.1.5.4.21); pPr (§5.1.5.2.7)

Child Elements	Subclause
spcPct (Spacing Percent)	§5.1.5.2.10
spcPts (Spacing Points)	§5.1.5.2.11

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextSpacing">
   <choice>
      <element name="spcPct" type="CT_TextSpacingPercent"/>
      <element name="spcPts" type="CT_TextSpacingPoint"/>
   </choice>
</complexType>
```

5.1.5.2.9 spcBef (Space Before)

This element specifies the amount of vertical white space that will be present before a paragraph. This space is specified in either percentage or points via the child elements spcPct and spcPts.

[*Example*: Consider the DrawingML shown below.

```
<p:txBody>
  …
  <a:p>
    <a:pPr …>
      <a:spcBef>
        <a:spcPts val="1800"/>
      </a:spcBef>
      <a:spcAft>
        <a:spcPts val="600"/>
      </a:spcAft>
    </a:pPr>
    …
    <a:t>Sample Text</a:t>
    …
  </a:p>
  …
</p:txBody>
```

The above paragraph of text will be formatted to have a spacing both before and after the paragraph text. The spacing before will be a size of 18 points, or value=1800 and the spacing after will be a size of 6 points, or value=600. *end example*]

Parent Elements
defPPr (§5.1.5.2.2); lvl1pPr (§5.1.5.4.13); lvl2pPr (§5.1.5.4.14); lvl3pPr (§5.1.5.4.15); lvl4pPr (§5.1.5.4.16); lvl5pPr (§5.1.5.4.17); lvl6pPr (§5.1.5.4.18); lvl7pPr (§5.1.5.4.19); lvl8pPr (§5.1.5.4.20); lvl9pPr (§5.1.5.4.21); pPr (§5.1.5.2.7)

Child Elements	Subclause
spcPct (Spacing Percent)	§5.1.5.2.10
spcPts (Spacing Points)	§5.1.5.2.11

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextSpacing">
   <choice>
      <element name="spcPct" type="CT_TextSpacingPercent"/>
      <element name="spcPts" type="CT_TextSpacingPoint"/>
   </choice>
</complexType>
```

5.1.5.2.10 spcPct (Spacing Percent)

This element specifies the amount of white space that is to be used between lines and paragraphs in the form of a percentage of the text size. The text size that is used to calculate the spacing here is the text for each run, with the largest text size having precedence. That is if there is a run of text with 10 point font and within the same paragraph on the same line there is a run of text with a 12 point font size then the 12 point should be used to calculate the spacing to be used.

[*Example*: Consider the DrawingML shown below.

```
<p:txBody>
  …
  <a:p>
    <a:pPr …>
      <a:spcBef>
        <a:spcPct val="200000"/>
      </a:spcBef>
    </a:pPr>
    …
    <a:t>Sample Text</a:t>
    …
  </a:p>
  …
</p:txBody>
```

The above paragraph of text will be formatted to have a spacing before the paragraph text. This spacing will be 200% of the size of the largest text on each line. *end example*]

Parent Elements
lnSpc (§5.1.5.2.5); spcAft (§5.1.5.2.8); spcBef (§5.1.5.2.9)

Attributes	Description
val (Value)	Specifies the percentage of the size that the white space should be. It is specified here in terms of 100% being equal to 100000 and 1% being specified in increments of 1000. The possible values for this attribute are defined by the ST_TextSpacingPercent simple type (§5.1.12.77).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextSpacingPercent">
   <attribute name="val" type="ST_TextSpacingPercent" use="required"/>
</complexType>
```

5.1.5.2.11 spcPts (Spacing Points)

This element specifies the amount of white space that is to be used between lines and paragraphs in the form of a text point size. The size is specified using points where 100 is equal to 1 point font and 1200 is equal to 12 point.

[*Example*: Consider the DrawingML shown below.

```
<p:txBody>
  …
  <a:p>
    <a:pPr …>
      <a:spcBef>
        <a:spcPts val="1400"/>
      </a:spcBef>
    </a:pPr>
    …
    <a:t>Sample Text</a:t>
    …
  </a:p>
  …
</p:txBody>
```

The above paragraph of text will be formatted to have a spacing before the paragraph text. This spacing will be a size of 14 points due to val="1400". *end example*]

Parent Elements
lnSpc (§5.1.5.2.5); spcAft (§5.1.5.2.8); spcBef (§5.1.5.2.9)

Attributes	Description
val (Value)	Specifies the size of the white space in point size. Whole points are specified in increments of 100 starting with 100 being a point size of 1. For instance a font point size of 12 would be 1200 and a font point size of 12.5 would be 1250. The possible values for this attribute are defined by the ST_TextSpacingPoint simple type (§5.1.12.78).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextSpacingPoint">
   <attribute name="val" type="ST_TextSpacingPoint" use="required"/>
</complexType>
```

5.1.5.2.12 tab (Tab Stop)

This element specifies a single tab stop to be used on a line of text when there are one or more tab characters present within the text. When there is more than one present than they should be utilized in increasing position order which is specified via the pos attribute.

[*Example*: Consider the DrawingML shown below.

```
<p:txBody>
  …
  <a:p>
    <a:pPr …>
      <a:tabLst>
        <a:tab pos="2292350" algn="l"/>
        <a:tab pos="2627313" algn="l"/>
        <a:tab pos="2743200" algn="l"/>
        <a:tab pos="2974975" algn="l"/>
      </a:tabLst>
    </a:pPr>
    …
    <a:t>Sample Text</a:t>
    …
  </a:p>
  …
</p:txBody>
```

The paragraph within which this <a:tab> information resides will have a total of 4 unique tab stops that should be listed in order of increasing position. Along with specifying the tab position each tab allows for the specifying of an alignment. *end example*]

Parent Elements
tabLst (§5.1.5.2.13)

Attributes	Description
algn (Tab Alignment)	Specifies the alignment that is to be applied to text using this tab stop. If this attribute is omitted then the application default for the generating application. The possible values for this attribute are defined by the ST_TextTabAlignType simple type (§5.1.12.80).
pos (Tab Position)	Specifies the position of the tab stop relative to the left margin. If this attribute is omitted then the application default for tab stops will be used. The possible values for this attribute are defined by the ST_Coordinate32 simple type (§5.1.12.17).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextTabStop">
   <attribute name="pos" type="ST_Coordinate32" use="optional"/>
   <attribute name="algn" type="ST_TextTabAlignType" use="optional"/>
</complexType>
```

5.1.5.2.13 tabLst (Tab List)

This element specifies the list of all tab stops that are to be used within a paragraph. These tabs should be used when describing any custom tab stops within the document. If these are not specified then the default tab stops of the generating application should be used.

Parent Elements
defPPr (§5.1.5.2.2); lvl1pPr (§5.1.5.4.13); lvl2pPr (§5.1.5.4.14); lvl3pPr (§5.1.5.4.15); lvl4pPr (§5.1.5.4.16); lvl5pPr (§5.1.5.4.17); lvl6pPr (§5.1.5.4.18); lvl7pPr (§5.1.5.4.19); lvl8pPr (§5.1.5.4.20); lvl9pPr (§5.1.5.4.21); pPr (§5.1.5.2.7)

Child Elements	Subclause
tab (Tab Stop)	§5.1.5.2.12

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextTabStopList">
   <sequence>
      <element name="tab" type="CT_TextTabStop" minOccurs="0" maxOccurs="32"/>
   </sequence>
</complexType>
```

5.1.5.3 Run Formatting

Run level formatting is the most granular property level and allows for the specifying of all low level text properties. The text run is what all paragraphs are derived from and thus specifying various properties per run will allow for a diversely formatted text paragraph.

[*Example*: Consider the case where have multiple runs within a paragraph and you wish to apply bold to only one of them without having to split up the text into higher level XML groups. To do this we would simply apply the bold run property to the text run that we wish to format as shown below.

```
<a:r>
…
</a:r>
<a:r>
  <a:rPr lang="en-US" b="1" dirty="0" smtClean="0"/>
  <a:t>This text will be bold</a:t>
</a:r>
```

```
<a:r>
…
</a:r>
```

end example]

5.1.5.3.1 cs (Complex Script Font)

This element specifies that a complex script font be used for a specific run of text. This font will be specified with a typeface attribute much like the others but will specifically be classified as a complex script font.

[*Example*: Consider the DrawingML shown below.

```
<a:r>
  <a:rPr …>
    <a:cs typeface="Sample Font"/>
  </a:rPr>
  <a:t>Sample Text</a:t>
</a:r>
```

The above run of text will be rendered using the complex script font "Sample Font". *end example*]

Parent Elements
defRPr (§5.1.5.3.2); endParaRPr (§5.1.5.2.3); font (§5.1.4.2.13); majorFont (§5.1.4.1.24); minorFont (§5.1.4.1.25); rPr (§5.1.5.3.9)

Attributes	Description
charset (Similar Character Set)	Specifies the most similar character set to the one being used. This is useful if the generating application cannot use the current font and must choose a similar one. The possible values for this attribute are defined by the XML Schema byte datatype.
panose (Panose Setting)	Specifies the panose standard setting that will be used to determine the closest matching font by any generating application that employs this method. The possible values for this attribute are defined by the ST_Panose simple type (§5.1.12.37).
pitchFamily (Similar Font Family)	Specifies the most similar font family to the one being used. This is useful if the generating application cannot use the current font and must choose a similar one. The possible values for this attribute are defined by the XML Schema byte datatype.
typeface (Text Typeface)	Specifies the typeface, or name of the font that is to be used for a bullet. The typeface used here should be selected from the font list of the generating application. The possible values for this attribute are defined by the ST_TextTypeface simple type (§5.1.12.81).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextFont">
   <attribute name="typeface" type="ST_TextTypeface"/>
   <attribute name="panose" type="ST_Panose" use="optional"/>
   <attribute name="pitchFamily" type="xsd:byte" use="optional" default="0"/>
   <attribute name="charset" type="xsd:byte" use="optional" default="1"/>
</complexType>
```

5.1.5.3.2 defRPr (Default Text Run Properties)

This element contains all default run level text properties for the text runs within a containing paragraph. These properties are to be used when overriding properties have not been defined within the rPr element.

[*Example*: Consider the DrawingML shown below.

```
<a:p>
  …
  <a:rPr u="sng"/>
  …
  <a:t>Some Text</a:t>
  …
</a:p>
```

The run of text described above will be formatting with a single underline of text matching color. *end example*]

Parent Elements
defPPr (§5.1.5.2.2); lvl1pPr (§5.1.5.4.13); lvl2pPr (§5.1.5.4.14); lvl3pPr (§5.1.5.4.15); lvl4pPr (§5.1.5.4.16); lvl5pPr (§5.1.5.4.17); lvl6pPr (§5.1.5.4.18); lvl7pPr (§5.1.5.4.19); lvl8pPr (§5.1.5.4.20); lvl9pPr (§5.1.5.4.21); pPr (§5.1.5.2.7)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
cs (Complex Script Font)	§5.1.5.3.1
ea (East Asian Font)	§5.1.5.3.3
effectDag (Effect Container)	§5.1.10.25
effectLst (Effect Container)	§5.1.10.26
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35
highlight (Highlight Color)	§5.1.5.3.4
hlinkClick (Click Hyperlink)	§5.1.5.3.5
hlinkMouseOver (Mouse-Over Hyperlink)	§5.1.5.3.6

Child Elements	Subclause
latin (Latin Font)	§5.1.5.3.7
ln (Outline)	§5.1.2.1.24
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
solidFill (Solid Fill)	§5.1.10.54
sym (Symbol Font)	§5.1.5.3.10
uFill (Underline Fill)	§5.1.5.3.12
uFillTx (Underline Fill Properties Follow Text)	§5.1.5.3.13
uLn (Underline Stroke)	§5.1.5.3.14
uLnTx (Underline Follows Text)	§5.1.5.3.15

Attributes	Description
altLang (Alternative Language)	Specifies the alternate language to use when the generating application is displaying the user interface controls. If this attribute is omitted, than the lang attribute will be used here. The possible values for this attribute are defined by the ST_TextLanguageID simple type (§5.1.12.72).

b (Bold)	Specifies whether a run of text will be formatted as bold text. If this attribute is omitted, than a value of 0, or false is assumed.
	[*Example*: Consider the DrawingML shown below.
	`<a:p>`
	`…`
	`<a:rPr b="1"/>`
	`…`
	`<a:t>Some Text</a:t>`
	`…`
	`</a:p>`
	The above run of text will be formatted as bold text. *end example*]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

baseline (Baseline)	Specifies the baseline for both the superscript and subscript fonts. The size is specified using a percentage where 1000 is equal to 1 percent of the font size and 100000 is equal to 100 percent font of the font size. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
bmk (Bookmark Link Target)	Specifies the link target name that is used to reference to the proper link properties in a custom XML part within the document.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema string datatype.
cap (Capitalization)	Specifies the capitalization that is to be applied to the text run. This is a render-only modification and does not affect the actual characters stored in the text run. This attribute is also distinct from the toggle function where the actual characters stored in the text run are changed. The possible values for this attribute are defined by the ST_TextCapsType simple type (§5.1.12.64).
dirty (Dirty)	Specifies that the content of a text run has changed since the proofing tools have last been run. Effectively this flags text that must be checked again by the generating application for mistakes such as spelling, grammar, etc. The possible values for this attribute are defined by the XML Schema boolean datatype.
err (Spelling Error)	Specifies that when this run of text was checked for spelling, grammar, etc. that a mistake was indeed found. This allows the generating application to effectively save the state of the mistakes within the document instead of having to perform a full pass check upon opening the document. The possible values for this attribute are defined by the XML Schema boolean datatype.

i (Italics)	Specifies whether a run of text will be formatted as italic text. If this attribute is omitted, than a value of 0, or false is assumed.
	
	[*Example*: Consider the DrawingML shown below.
	
	`<a:p>`
	`…`
	`<a:rPr i="1"/>`
	`…`
	`<a:t>Some Text</a:t>`
	`…`
	`</a:p>`
	
	The above run of text will be formatted as italic text. *end example*]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

kern (Kerning)	Specifies the minimum font size at which character kerning will occur for this text run. Whole points are specified in increments of 100 starting with 100 being a point size of 1. For instance a font point size of 12 would be 1200 and a font point size of 12.5 would be 1250. If this attribute is omitted, than kerning will occur for all font sizes down to a 0 point font. The possible values for this attribute are defined by the ST_TextNonNegativePoint simple type (§5.1.12.74).
kumimoji	Specifies whether the numbers contained within vertical text will continue vertically with

Attributes	Description
(Kumimoji)	the text or whether they are to be displayed horizontally while the surrounding characters continue in a vertical fashion. If this attribute is omitted, than a value of 0, or false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
lang (Language ID)	Specifies the language to be used when the generating application is displaying the user interface controls. If this attribute is omitted, than the generating application may select a language of its choice. The possible values for this attribute are defined by the ST_TextLanguageID simple type (§5.1.12.72).
noProof (No Proofing)	Specifies that a run of text has been selected by the user to not be checked for mistakes. Therefore if there are spelling, grammar, etc mistakes within this text the generating application should ignore them. The possible values for this attribute are defined by the XML Schema boolean datatype.
normalizeH (Normalize Heights)	Specifies the normalization of height that is to be applied to the text run. This is a render-only modification and does not affect the actual characters stored in the text run. This attribute is also distinct from the toggle function where the actual characters stored in the text run are changed. If this attribute is omitted, than a value of 0, or false will be assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
smtClean (SmartTag Clean)	Specifies whether or not a text run has been checked for smart tags. This attribute acts much like the dirty attribute dose for the checking of spelling, grammar, etc. A value of true here indicates to the generating application that this text run should be checked for smart tags. If this attribute is omitted, than a value of 0, or false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
smtId (SmartTag ID)	Specifies the reference id for the smart tag. This id corresponds to a link within this slides relationship file. The following of this link within the relationship file will result in the actual smart tag information for this piece of text. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
spc (Spacing)	Specifies the spacing between characters within a text run. This spacing is specified numerically and should be consistently applied across the entire run of text by the generating application. Whole points are specified in increments of 100 starting with 100 being a point size of 1. For instance a font point size of 12 would be 1200 and a font point size of 12.5 would be 1250. If this attribute is omitted than a value of 0 or no adjustment is assumed. The possible values for this attribute are defined by the ST_TextPoint simple type (§5.1.12.75).

Attributes	Description

strike (Strikethrough)	Specifies whether a run of text will be formatted as strikethrough text. If this attribute is omitted, than no strikethrough is assumed.
	
	[*Example*: Consider the DrawingML shown below.
	
	`<a:p>`
	`…`
	`<a:rPr strike="sngStrike"/>`
	`…`
	`<a:t>Some Text</a:t>`
	`…`
	`</a:p>`
	
	The above run of text will be formatted as single strikethrough text. *end example*]
	
	The possible values for this attribute are defined by the ST_TextStrikeType simple type (§5.1.12.79).

sz (Font Size)	Specifies the size of text within a text run. Whole points are specified in increments of 100 starting with 100 being a point size of 1. For instance a font point size of 12 would be 1200 and a font point size of 12.5 would be 1250. If this attribute is omitted, than the value in defRPr should be used.
	
	[*Example*: Consider the DrawingML shown below.
	
	`<a:p>`
	`…`
	`<a:rPr sz="1200"/>`
	`…`
	`<a:t>Some Text</a:t>`
	`…`
	`</a:p>`
	
	The above run of text will be formatted with a 12 point text size. *end example*]
	
	The possible values for this attribute are defined by the ST_TextFontSize simple type (§5.1.12.68).

u (Underline)	Specifies whether a run of text will be formatted as underlined text. If this attribute is omitted, than no underline is assumed.
	
	[*Example*: Consider the DrawingML shown below.
	
	`<a:p>`
	`…`
	`<a:rPr u="sng"/>`
	`…`
	`<a:t>Some Text</a:t>`

Attributes	Description
	… </a:p> The above run of text will be formatted as single underline text. *end example*] The possible values for this attribute are defined by the ST_TextUnderlineType simple type (§5.1.12.82).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextCharacterProperties">
   <sequence>
      <element name="ln" type="CT_LineProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_FillProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_EffectProperties" minOccurs="0" maxOccurs="1"/>
      <element name="highlight" type="CT_Color" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextUnderlineLine" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextUnderlineFill" minOccurs="0" maxOccurs="1"/>
      <element name="latin" type="CT_TextFont" minOccurs="0" maxOccurs="1"/>
      <element name="ea" type="CT_TextFont" minOccurs="0" maxOccurs="1"/>
      <element name="cs" type="CT_TextFont" minOccurs="0" maxOccurs="1"/>
      <element name="sym" type="CT_TextFont" minOccurs="0" maxOccurs="1"/>
      <element name="hlinkClick" type="CT_Hyperlink" minOccurs="0" maxOccurs="1"/>
      <element name="hlinkMouseOver" type="CT_Hyperlink" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="kumimoji" type="xsd:boolean" use="optional"/>
   <attribute name="lang" type="ST_TextLanguageID" use="optional"/>
   <attribute name="altLang" type="ST_TextLanguageID" use="optional"/>
   <attribute name="sz" type="ST_TextFontSize" use="optional"/>
   <attribute name="b" type="xsd:boolean" use="optional"/>
   <attribute name="i" type="xsd:boolean" use="optional"/>
   <attribute name="u" type="ST_TextUnderlineType" use="optional"/>
   <attribute name="strike" type="ST_TextStrikeType" use="optional"/>
   <attribute name="kern" type="ST_TextNonNegativePoint" use="optional"/>
   <attribute name="cap" type="ST_TextCapsType" use="optional"/>
   <attribute name="spc" type="ST_TextPoint" use="optional"/>
   <attribute name="normalizeH" type="xsd:boolean" use="optional"/>
   <attribute name="baseline" type="ST_Percentage" use="optional"/>
   <attribute name="noProof" type="xsd:boolean" use="optional"/>
   <attribute name="dirty" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="err" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="smtClean" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="smtId" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="bmk" type="xsd:string" use="optional"/>
</complexType>
```

5.1.5.3.3 ea (East Asian Font)

This element specifies that an East Asian font be used for a specific run of text. This font will be specified with a typeface attribute much like the others but will specifically be classified as an East Asian font.

[*Example*: Consider the DrawingML shown below.

```
<a:r>
  <a:rPr …>
    <a:ea typeface="Sample Font"/>
  </a:rPr>
  <a:t>Sample Text</a:t>
</a:r>
```

The above run of text will be rendered using the East Asian font "Sample Font". *end example*]

Parent Elements
defRPr (§5.1.5.3.2); endParaRPr (§5.1.5.2.3); font (§5.1.4.2.13); majorFont (§5.1.4.1.24); minorFont (§5.1.4.1.25); rPr (§5.1.5.3.9)

Attributes	Description
charset (Similar Character Set)	Specifies the most similar character set to the one being used. This is useful if the generating application cannot use the current font and must choose a similar one. The possible values for this attribute are defined by the XML Schema byte datatype.
panose (Panose Setting)	Specifies the panose standard setting that will be used to determine the closest matching font by any generating application that employs this method. The possible values for this attribute are defined by the ST_Panose simple type (§5.1.12.37).
pitchFamily (Similar Font Family)	Specifies the most similar font family to the one being used. This is useful if the generating application cannot use the current font and must choose a similar one. The possible values for this attribute are defined by the XML Schema byte datatype.
typeface (Text Typeface)	Specifies the typeface, or name of the font that is to be used for a bullet. The typeface used here should be selected from the font list of the generating application. The possible values for this attribute are defined by the ST_TextTypeface simple type (§5.1.12.81).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextFont">
   <attribute name="typeface" type="ST_TextTypeface"/>
   <attribute name="panose" type="ST_Panose" use="optional"/>
   <attribute name="pitchFamily" type="xsd:byte" use="optional" default="0"/>
   <attribute name="charset" type="xsd:byte" use="optional" default="1"/>
</complexType>
```

5.1.5.3.4 highlight (Highlight Color)

This element specifies the highlight color that will be present for a run of text.

[*Example*: Consider the DrawingML shown below.

```
<p:txBody>
  …
  <a:p>
    <a:r>
      <a:rPr …>
        <a:highlight>
          <a:srgbClr val="FFFF00"/>
        </a:highlight>
      </a:rPr>
      …
      <a:t>Sample Text</a:t>
      …
    </a:r>
  </a:p>
  …
</p:txBody>
```

The above run of text will have a yellow highlight color as specified by the srgbClr child element. *end example*]

Parent Elements
defRPr (§5.1.5.3.2); endParaRPr (§5.1.5.2.3); rPr (§5.1.5.3.9)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

5.1.5.3.5 hlinkClick (Click Hyperlink)

Specifies the on-click hyperlink information to be applied to a run of text. When the hyperlink text is clicked the link will be fetched.

[*Example*: Consider the DrawingML shown below.

```
<p:txBody>
  …
  <a:p>
    <a:r>
      <a:rPr …>
        <a:hlinkClick r:id="rId2" tooltip="Some Sample Text"/>
      </a:rPr>
      …
      <a:t>Sample Text</a:t>
      …
    </a:r>
  </a:p>
  …
</p:txBody>
```

The above run of text will be a hyperlink that points to the resource pointed at by rId2 within this slides relationship file. Additionally this text should display a tooltip when the mouse is hovered over the run of text. *end example*]

Parent Elements
cNvPr (§5.2.2.3); cNvPr (§5.8.2.7); cNvPr (§4.4.1.12); cNvPr (§5.6.2.8); cNvPr (§5.1.2.1.8); defRPr (§5.1.5.3.2); docPr (§5.5.2.5); endParaRPr (§5.1.5.2.3); rPr (§5.1.5.3.9)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
snd (Hyperlink Sound)	§5.1.2.1.32

Attributes	Description
action (Action Setting)	Specifies an action that is to be taken when this hyperlink is activated. This may be used to specify a slide to be navigated to or a script of code to be run. The possible values for this attribute are defined by the XML Schema string datatype.
endSnd (End Sounds)	Specifies if the URL in question should stop all sounds that are playing when it is clicked. The possible values for this attribute are defined by the XML Schema boolean datatype.

Attributes	Description
highlightClick (Highlight Click)	Specifies if this attribute has already been used within this document. That is when a hyperlink has already been visited that this attribute would be utilized so the generating application may determine the color of this text. If this attribute is omitted, then a value of 0 or false is implied. The possible values for this attribute are defined by the XML Schema boolean datatype.
history (Add Hyperlink to Page History)	Specifies whether to add this URI to the history when navigating to it. This allows for the viewing of this presentation without the storing of history information on the viewing machine. If this attribute is omitted, then a value of 1, or true is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
id (Drawing Object Hyperlink Target) Namespace: .../officeDocument/2006/relationships	Specifies the relationship id that when looked up in this slides relationship file will contain the target of this hyperlink. This attribute cannot be omitted. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
invalidUrl (Invalid URL)	Specifies the URL when it has been determined by the generating application that the URL is invalid. That is the generating application may still store the URL but it is known that this URL is not correct. The possible values for this attribute are defined by the XML Schema string datatype.
tgtFrame (Target Frame)	Specifies the target frame that is to be used when opening this hyperlink. When the hyperlink is activated this attribute will be used to determine if a new window must be launched for viewing or if an existing one may be used. If this attribute is omitted, than a new window will be opened. The possible values for this attribute are defined by the XML Schema string datatype.
tooltip (Hyperlink Tooltip)	Specifies the tooltip that should be displayed when the hyperlink text is hovered over with the mouse. If this attribute is omitted, than the hyperlink text itself may be displayed. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Hyperlink">
   <sequence>
      <element name="snd" type="CT_EmbeddedWAVAudioFile" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute ref="r:id" use="optional"/>
   <attribute name="invalidUrl" type="xsd:string" use="optional" default=""/>
   <attribute name="action" type="xsd:string" use="optional" default=""/>
   <attribute name="tgtFrame" type="xsd:string" use="optional" default=""/>
   <attribute name="tooltip" type="xsd:string" use="optional" default=""/>
   <attribute name="history" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="highlightClick" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="endSnd" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.1.5.3.6 hlinkMouseOver (Mouse-Over Hyperlink)

Specifies the mouse-over hyperlink information to be applied to a run of text. When the mouse is hovered over this hyperlink text the link will be fetched.

[*Example*: Consider the DrawingML shown below.

```
<p:txBody>
  …
  <a:p>
    <a:r>
      <a:rPr …>
        <a:hlinkMouseOver r:id="rId2" tooltip="Some Sample Text"/>
      </a:rPr>
      …
      <a:t>Sample Text</a:t>
      …
    </a:r>
  </a:p>
  …
</p:txBody>
```

The above run of text will be a hyperlink that points to the resource pointed at by rId2 within this slides relationship file. Additionally this text should display a tooltip when the mouse is hovered over the run of text. *end example*]

Parent Elements
defRPr (§5.1.5.3.2); endParaRPr (§5.1.5.2.3); rPr (§5.1.5.3.9)

Child Elements	Subclause

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
snd (Hyperlink Sound)	§5.1.2.1.32

Attributes	Description
action (Action Setting)	Specifies an action that is to be taken when this hyperlink is activated. This may be used to specify a slide to be navigated to or a script of code to be run. The possible values for this attribute are defined by the XML Schema string datatype.
endSnd (End Sounds)	Specifies if the URL in question should stop all sounds that are playing when it is clicked. The possible values for this attribute are defined by the XML Schema boolean datatype.
highlightClick (Highlight Click)	Specifies if this attribute has already been used within this document. That is when a hyperlink has already been visited that this attribute would be utilized so the generating application may determine the color of this text. If this attribute is omitted, then a value of 0 or false is implied. The possible values for this attribute are defined by the XML Schema boolean datatype.
history (Add Hyperlink to Page History)	Specifies whether to add this URI to the history when navigating to it. This allows for the viewing of this presentation without the storing of history information on the viewing machine. If this attribute is omitted, then a value of 1, or true is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
id (Drawing Object Hyperlink Target) Namespace: .../officeDocument/2006/relationships	Specifies the relationship id that when looked up in this slides relationship file will contain the target of this hyperlink. This attribute cannot be omitted. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
invalidUrl (Invalid URL)	Specifies the URL when it has been determined by the generating application that the URL is invalid. That is the generating application may still store the URL but it is known that this URL is not correct. The possible values for this attribute are defined by the XML Schema string datatype.
tgtFrame (Target Frame)	Specifies the target frame that is to be used when opening this hyperlink. When the hyperlink is activated this attribute will be used to determine if a new window must be launched for viewing or if an existing one may be used. If this attribute is omitted, than a new window will be opened. The possible values for this attribute are defined by the XML Schema string datatype.
tooltip (Hyperlink Tooltip)	Specifies the tooltip that should be displayed when the hyperlink text is hovered over with the mouse. If this attribute is omitted, than the hyperlink text itself may be

Attributes	Description
	displayed. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Hyperlink">
   <sequence>
      <element name="snd" type="CT_EmbeddedWAVAudioFile" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute ref="r:id" use="optional"/>
   <attribute name="invalidUrl" type="xsd:string" use="optional" default=""/>
   <attribute name="action" type="xsd:string" use="optional" default=""/>
   <attribute name="tgtFrame" type="xsd:string" use="optional" default=""/>
   <attribute name="tooltip" type="xsd:string" use="optional" default=""/>
   <attribute name="history" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="highlightClick" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="endSnd" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.1.5.3.7 latin (Latin Font)

This element specifies that a Latin font be used for a specific run of text. This font will be specified with a typeface attribute much like the others but will specifically be classified as a Latin font.

[*Example*: Consider the DrawingML shown below.

```
<a:r>
  <a:rPr …>
    <a:latin typeface="Sample Font"/>
  </a:rPr>
  <a:t>Sample Text</a:t>
</a:r>
```

The above run of text will be rendered using the Latin font "Sample Font". *end example*]

Parent Elements
defRPr (§5.1.5.3.2); endParaRPr (§5.1.5.2.3); font (§5.1.4.2.13); majorFont (§5.1.4.1.24); minorFont (§5.1.4.1.25); rPr (§5.1.5.3.9)

Attributes	Description
charset (Similar Character Set)	Specifies the most similar character set to the one being used. This is useful if the generating application cannot use the current font and must choose a similar one. The possible values for this attribute are defined by the XML Schema byte datatype.

Attributes	Description
panose (Panose Setting)	Specifies the panose standard setting that will be used to determine the closest matching font by any generating application that employs this method. The possible values for this attribute are defined by the ST_Panose simple type (§5.1.12.37).
pitchFamily (Similar Font Family)	Specifies the most similar font family to the one being used. This is useful if the generating application cannot use the current font and must choose a similar one. The possible values for this attribute are defined by the XML Schema byte datatype.
typeface (Text Typeface)	Specifies the typeface, or name of the font that is to be used for a bullet. The typeface used here should be selected from the font list of the generating application. The possible values for this attribute are defined by the ST_TextTypeface simple type (§5.1.12.81).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextFont">
   <attribute name="typeface" type="ST_TextTypeface"/>
   <attribute name="panose" type="ST_Panose" use="optional"/>
   <attribute name="pitchFamily" type="xsd:byte" use="optional" default="0"/>
   <attribute name="charset" type="xsd:byte" use="optional" default="1"/>
</complexType>
```

5.1.5.3.8 r (Text Run)

This element specifies the presence of a run of text within the containing text body. The run element is the lowest level text separation mechanism within a text body. A text run may contain text run properties associated with the run. If no properties are listed then properties specified in the defRPr element are used.

[*Example*: Consider the case where the user would like to describe a text body that will contain two runs of text and would like one to be bold and the other not. The following DrawingML would specify such a text body.

```
<p:txBody>
  …
  <a:r>
    <a:rPr b="1">
    </a:rPr>
    <a:t>Some text</a:t>
  </a:r>
  …
  <a:r>
    <a:rPr/>
    <a:t>Some text</a:t>
  </a:r>
</p:txBody>
```

The above text body will have the first run be formatted bold and the second normally. *end example*]

Parent Elements
p (§5.1.5.2.6)

Child Elements	Subclause
rPr (Text Run Properties)	§5.1.5.3.9
t (Text String)	§5.1.5.3.11

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RegularTextRun">
   <sequence>
      <element name="rPr" type="CT_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>
      <element name="t" type="xsd:string" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.5.3.9 rPr (Text Run Properties)

This element contains all run level text properties for the text runs within a containing paragraph.

[*Example*: Consider the DrawingML shown below.

```
<a:p>
  …
  <a:rPr u="sng"/>
  …
  <a:t>Some Text</a:t>
  …
</a:p>
```

The run of text described above will be formatting with a single underline of text matching color. *end example*]

Parent Elements
br (§5.1.5.2.1); fld (§5.1.5.2.4); r (§5.1.5.3.8)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
cs (Complex Script Font)	§5.1.5.3.1
ea (East Asian Font)	§5.1.5.3.3
effectDag (Effect Container)	§5.1.10.25
effectLst (Effect Container)	§5.1.10.26

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35
highlight (Highlight Color)	§5.1.5.3.4
hlinkClick (Click Hyperlink)	§5.1.5.3.5
hlinkMouseOver (Mouse-Over Hyperlink)	§5.1.5.3.6
latin (Latin Font)	§5.1.5.3.7
ln (Outline)	§5.1.2.1.24
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
solidFill (Solid Fill)	§5.1.10.54
sym (Symbol Font)	§5.1.5.3.10
uFill (Underline Fill)	§5.1.5.3.12
uFillTx (Underline Fill Properties Follow Text)	§5.1.5.3.13
uLn (Underline Stroke)	§5.1.5.3.14
uLnTx (Underline Follows Text)	§5.1.5.3.15

Attributes	Description
altLang (Alternative Language)	Specifies the alternate language to use when the generating application is displaying the user interface controls. If this attribute is omitted, than the lang attribute will be used here. The possible values for this attribute are defined by the ST_TextLanguageID simple type (§5.1.12.72).

b (Bold)	Specifies whether a run of text will be formatted as bold text. If this attribute is omitted, than a value of 0, or false is assumed.
	[*Example*: Consider the DrawingML shown below.
	`<a:p>`
	`…`
	`<a:rPr b="1"/>`
	`…`
	`<a:t>Some Text</a:t>`
	`…`
	`</a:p>`
	The above run of text will be formatted as bold text. *end example*]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

baseline (Baseline)	Specifies the baseline for both the superscript and subscript fonts. The size is specified using a percentage where 1000 is equal to 1 percent of the font size and 100000 is equal

Attributes	Description
	to 100 percent font of the font size. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
bmk (Bookmark Link Target)	Specifies the link target name that is used to reference to the proper link properties in a custom XML part within the document. The possible values for this attribute are defined by the XML Schema string datatype.
cap (Capitalization)	Specifies the capitalization that is to be applied to the text run. This is a render-only modification and does not affect the actual characters stored in the text run. This attribute is also distinct from the toggle function where the actual characters stored in the text run are changed. The possible values for this attribute are defined by the ST_TextCapsType simple type (§5.1.12.64).
dirty (Dirty)	Specifies that the content of a text run has changed since the proofing tools have last been run. Effectively this flags text that must be checked again by the generating application for mistakes such as spelling, grammar, etc. The possible values for this attribute are defined by the XML Schema boolean datatype.
err (Spelling Error)	Specifies that when this run of text was checked for spelling, grammar, etc. that a mistake was indeed found. This allows the generating application to effectively save the state of the mistakes within the document instead of having to perform a full pass check upon opening the document. The possible values for this attribute are defined by the XML Schema boolean datatype.

i (Italics)	Specifies whether a run of text will be formatted as italic text. If this attribute is omitted, than a value of 0, or false is assumed.
	
	[*Example*: Consider the DrawingML shown below.
	
	`<a:p>`
	`…`
	`<a:rPr i="1"/>`
	`…`
	`<a:t>Some Text</a:t>`
	`…`
	`</a:p>`
	
	The above run of text will be formatted as italic text. *end example*]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

kern (Kerning)	Specifies the minimum font size at which character kerning will occur for this text run. Whole points are specified in increments of 100 starting with 100 being a point size of 1.

Attributes	Description
	For instance a font point size of 12 would be 1200 and a font point size of 12.5 would be 1250. If this attribute is omitted, than kerning will occur for all font sizes down to a 0 point font. The possible values for this attribute are defined by the ST_TextNonNegativePoint simple type (§5.1.12.74).
kumimoji (Kumimoji)	Specifies whether the numbers contained within vertical text will continue vertically with the text or whether they are to be displayed horizontally while the surrounding characters continue in a vertical fashion. If this attribute is omitted, than a value of 0, or false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
lang (Language ID)	Specifies the language to be used when the generating application is displaying the user interface controls. If this attribute is omitted, than the generating application may select a language of its choice. The possible values for this attribute are defined by the ST_TextLanguageID simple type (§5.1.12.72).
noProof (No Proofing)	Specifies that a run of text has been selected by the user to not be checked for mistakes. Therefore if there are spelling, grammar, etc mistakes within this text the generating application should ignore them. The possible values for this attribute are defined by the XML Schema boolean datatype.
normalizeH (Normalize Heights)	Specifies the normalization of height that is to be applied to the text run. This is a render-only modification and does not affect the actual characters stored in the text run. This attribute is also distinct from the toggle function where the actual characters stored in the text run are changed. If this attribute is omitted, than a value of 0, or false will be assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
smtClean (SmartTag Clean)	Specifies whether or not a text run has been checked for smart tags. This attribute acts much like the dirty attribute dose for the checking of spelling, grammar, etc. A value of true here indicates to the generating application that this text run should be checked for smart tags. If this attribute is omitted, than a value of 0, or false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
smtId (SmartTag ID)	Specifies the reference id for the smart tag. This id corresponds to a link within this slides relationship file. The following of this link within the relationship file will result in the actual smart tag information for this piece of text. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
spc (Spacing)	Specifies the spacing between characters within a text run. This spacing is specified numerically and should be consistently applied across the entire run of text by the

Attributes	Description
	generating application. Whole points are specified in increments of 100 starting with 100 being a point size of 1. For instance a font point size of 12 would be 1200 and a font point size of 12.5 would be 1250. If this attribute is omitted than a value of 0 or no adjustment is assumed. The possible values for this attribute are defined by the ST_TextPoint simple type (§5.1.12.75).

strike (Strikethrough)	Specifies whether a run of text will be formatted as strikethrough text. If this attribute is omitted, than no strikethrough is assumed.
	
	[*Example*: Consider the DrawingML shown below.
	
	`<a:p>`
	`…`
	`<a:rPr strike="sngStrike"/>`
	`…`
	`<a:t>Some Text</a:t>`
	`…`
	`</a:p>`
	
	The above run of text will be formatted as single strikethrough text. *end example*]
	
	The possible values for this attribute are defined by the ST_TextStrikeType simple type (§5.1.12.79).

sz (Font Size)	Specifies the size of text within a text run. Whole points are specified in increments of 100 starting with 100 being a point size of 1. For instance a font point size of 12 would be 1200 and a font point size of 12.5 would be 1250. If this attribute is omitted, than the value in defRPr should be used.
	
	[*Example*: Consider the DrawingML shown below.
	
	`<a:p>`
	`…`
	`<a:rPr sz="1200"/>`
	`…`
	`<a:t>Some Text</a:t>`
	`…`
	`</a:p>`
	
	The above run of text will be formatted with a 12 point text size. *end example*]
	
	The possible values for this attribute are defined by the ST_TextFontSize simple type (§5.1.12.68).

u (Underline)	Specifies whether a run of text will be formatted as underlined text. If this attribute is omitted, than no underline is assumed.

Attributes	Description

	[*Example*: Consider the DrawingML shown below.
	
	`<a:p>`
	`…`
	`<a:rPr u="sng"/>`
	`…`
	`<a:t>Some Text</a:t>`
	`…`
	`</a:p>`
	
	The above run of text will be formatted as single underline text. *end example*]
	
	The possible values for this attribute are defined by the ST_TextUnderlineType simple type (§5.1.12.82).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextCharacterProperties">
   <sequence>
      <element name="ln" type="CT_LineProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_FillProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_EffectProperties" minOccurs="0" maxOccurs="1"/>
      <element name="highlight" type="CT_Color" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextUnderlineLine" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextUnderlineFill" minOccurs="0" maxOccurs="1"/>
      <element name="latin" type="CT_TextFont" minOccurs="0" maxOccurs="1"/>
      <element name="ea" type="CT_TextFont" minOccurs="0" maxOccurs="1"/>
      <element name="cs" type="CT_TextFont" minOccurs="0" maxOccurs="1"/>
      <element name="sym" type="CT_TextFont" minOccurs="0" maxOccurs="1"/>
      <element name="hlinkClick" type="CT_Hyperlink" minOccurs="0" maxOccurs="1"/>
      <element name="hlinkMouseOver" type="CT_Hyperlink" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="kumimoji" type="xsd:boolean" use="optional"/>
   <attribute name="lang" type="ST_TextLanguageID" use="optional"/>
   <attribute name="altLang" type="ST_TextLanguageID" use="optional"/>
   <attribute name="sz" type="ST_TextFontSize" use="optional"/>
   <attribute name="b" type="xsd:boolean" use="optional"/>
   <attribute name="i" type="xsd:boolean" use="optional"/>
   <attribute name="u" type="ST_TextUnderlineType" use="optional"/>
   <attribute name="strike" type="ST_TextStrikeType" use="optional"/>
   <attribute name="kern" type="ST_TextNonNegativePoint" use="optional"/>
   <attribute name="cap" type="ST_TextCapsType" use="optional"/>
   <attribute name="spc" type="ST_TextPoint" use="optional"/>
   <attribute name="normalizeH" type="xsd:boolean" use="optional"/>
   <attribute name="baseline" type="ST_Percentage" use="optional"/>
   <attribute name="noProof" type="xsd:boolean" use="optional"/>
   <attribute name="dirty" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="err" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="smtClean" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="smtId" type="xsd:unsignedInt" use="optional" default="0"/>
   <attribute name="bmk" type="xsd:string" use="optional"/>
</complexType>
```

5.1.5.3.10 sym (Symbol Font)

This element specifies that a symbol font be used for a specific run of text. This font will be specified with a typeface attribute much like the others but will specifically be classified as a symbol font.

[*Example*: Consider the DrawingML shown below.

```
<a:r>
  <a:rPr …>
    <a:sym typeface="Sample Font"/>
  </a:rPr>
  <a:t>Sample Text</a:t>
</a:r>
```

The above run of text will be rendered using the symbol font "Sample Font". *end example*]

Parent Elements
defRPr (§5.1.5.3.2); endParaRPr (§5.1.5.2.3); rPr (§5.1.5.3.9)

Attributes	Description
charset (Similar Character Set)	Specifies the most similar character set to the one being used. This is useful if the generating application cannot use the current font and must choose a similar one. The possible values for this attribute are defined by the XML Schema byte datatype.
panose (Panose Setting)	Specifies the panose standard setting that will be used to determine the closest matching font by any generating application that employs this method. The possible values for this attribute are defined by the ST_Panose simple type (§5.1.12.37).
pitchFamily (Similar Font Family)	Specifies the most similar font family to the one being used. This is useful if the generating application cannot use the current font and must choose a similar one. The possible values for this attribute are defined by the XML Schema byte datatype.
typeface (Text Typeface)	Specifies the typeface, or name of the font that is to be used for a bullet. The typeface used here should be selected from the font list of the generating application. The possible values for this attribute are defined by the ST_TextTypeface simple type (§5.1.12.81).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextFont">
   <attribute name="typeface" type="ST_TextTypeface"/>
   <attribute name="panose" type="ST_Panose" use="optional"/>
   <attribute name="pitchFamily" type="xsd:byte" use="optional" default="0"/>
   <attribute name="charset" type="xsd:byte" use="optional" default="1"/>
</complexType>
```

5.1.5.3.11 t (Text String)

This element specifies the actual text for this text run. This is the text that will be formatted using all specified body, paragraph and run properties. This element must be present within a run of text.

[*Example*: Consider the DrawingML shown below.

```
<p:txBody>
  …
  <a:p>
    …
    <a:r>
      …
      <a:t>Sample Text</a:t>
      …
    </a:r>
    …
  </a:p>
  …
</p:txBody>
```

The above DrawingML specifies a text body containing a single paragraph, containing a single run which contains the actual text specified with the <a:t> element.

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
fld (§5.1.5.2.4); r (§5.1.5.3.8)

5.1.5.3.12 uFill (Underline Fill)

This element specifies the fill color of an underline for a run of text.

[*Example*: Consider the DrawingML shown below.

```
<p:txBody>
  …
  <a:p>
    <a:r>
      <a:rPr …>
        <a:uFill>
          <a:solidFill>
            <a:srgbClr val="FFFF00"/>
          </a:solidFill>
        </a:uFill>
      </a:rPr>
```

```
        …
        <a:t>Sample Text</a:t>
        …
      </a:r>
    </a:p>
    …
  </p:txBody>
```

The underline color of the above text will be yellow specified by the srgbClr child element. *end example*]

Parent Elements
defRPr (§5.1.5.3.2); endParaRPr (§5.1.5.2.3); rPr (§5.1.5.3.9)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
solidFill (Solid Fill)	§5.1.10.54

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextUnderlineFillGroupWrapper">
   <group ref="EG_FillProperties" minOccurs="1" maxOccurs="1"/>
</complexType>
```

5.1.5.3.13 uFillTx (Underline Fill Properties Follow Text)

This element specifies that the fill color of an underline for a run of text should be of the same color as the text run within which it is contained.

[*Example*: Consider the DrawingML shown below.

```
  <p:txBody>
    …
    <a:p>
      <a:r>
        <a:rPr …>
          <a:uFillTx>
        </a:rPr>
        …
```

```
        <a:t>Sample Text</a:t>
        …
      </a:r>
    </a:p>
    …
  </p:txBody>
```

The underline color of the above text will follow the color of the text run within which it resides. *end example*]

Parent Elements
defRPr (§5.1.5.3.2); endParaRPr (§5.1.5.2.3); rPr (§5.1.5.3.9)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextUnderlineFillFollowText"/>
```

5.1.5.3.14 uLn (Underline Stroke)

This element specifies the properties for the stroke of the underline that will be present within a run of text.

[*Example*: Consider the DrawingML shown below.

```
  <p:txBody>
    …
    <a:p>
      <a:r>
        <a:rPr …>
          <a:uLn algn="r">
        </a:rPr>
        …
        <a:t>Sample Text</a:t>
        …
      </a:r>
    </a:p>
    …
  </p:txBody>
```

The underline alignment of the above text will be right aligned. *end example*]

Parent Elements
defRPr (§5.1.5.3.2); endParaRPr (§5.1.5.2.3); rPr (§5.1.5.3.9)

Child Elements	Subclause
bevel (Line Join Bevel)	§5.1.10.9
custDash (Custom Dash)	§5.1.10.21

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
headEnd (Line Head/End Style)	§5.1.10.38
miter (Miter Line Join)	§5.1.10.43
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
prstDash (Preset Dash)	§5.1.10.48
round (Round Line Join)	§5.1.10.52
solidFill (Solid Fill)	§5.1.10.54
tailEnd (Tail line end style)	§5.1.10.57

Attributes	Description
algn (Stroke Alignment)	Specifies the alignment to be used for the underline stroke. The possible values for this attribute are defined by the ST_PenAlignment simple type (§5.1.12.40).
cap (Line Ending Cap Type)	Specifies the ending caps that should be used for this line. Examples of cap types are rounded, flat, etc. If this attribute is omitted, than a value of square is assumed. The possible values for this attribute are defined by the ST_LineCap simple type (§5.1.12.31).
cmpd (Compound Line Type)	Specifies the compound line type to be used for the underline stroke. If this attribute is omitted, then a value of sng is assumed. The possible values for this attribute are defined by the ST_CompoundLine simple type (§5.1.12.15).
w (Line Width)	Specifies the width to be used for the underline stroke. If this attribute is omitted, then a value of 0 is assumed. The possible values for this attribute are defined by the ST_LineWidth simple type (§5.1.12.35).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LineProperties">
   <sequence>
      <group ref="EG_LineFillProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_LineDashProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_LineJoinProperties" minOccurs="0" maxOccurs="1"/>
      <element name="headEnd" type="CT_LineEndProperties" minOccurs="0" maxOccurs="1"/>
      <element name="tailEnd" type="CT_LineEndProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="w" type="ST_LineWidth" use="optional"/>
   <attribute name="cap" type="ST_LineCap" use="optional"/>
   <attribute name="cmpd" type="ST_CompoundLine" use="optional"/>
   <attribute name="algn" type="ST_PenAlignment" use="optional"/>
</complexType>
```

5.1.5.3.15 uLnTx (Underline Follows Text)

This element specifies that the stroke style of an underline for a run of text should be of the same as the text run within which it is contained.

[*Example*: Consider the DrawingML shown below.

```
<p:txBody>
  …
  <a:p>
    <a:r>
      <a:rPr …>
        <a:uLnTx>
      </a:rPr>
      …
      <a:t>Sample Text</a:t>
      …
    </a:r>
  </a:p>
  …
</p:txBody>
```

The underline stroke of the above text will follow the stroke of the run text within which it resides. *end example*]

Parent Elements
defRPr (§5.1.5.3.2); endParaRPr (§5.1.5.2.3); rPr (§5.1.5.3.9)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextUnderlineLineFollowText"/>
```

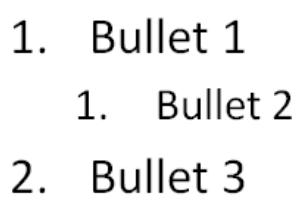
5.1.5.4 Bullets and Numbering

In addition to the above body, paragraph and text run properties there can also be a structure of bullets and numbering that can be defined by utilizing a few of these layers. Since Bullet and Numbering does span multiple formatting levels it will be described on it's own in the following section.

5.1.5.4.1 buAutoNum (Auto-Numbered Bullet)

This element specifies that automatic numbered bullet points should be applied to a paragraph. These are not just numbers used as bullet points but instead automatically assigned numbers that are based on both buAutoNum attributes and paragraph level.

[*Example*: Consider the DrawingML content shown below.




```
<p:txBody>
  …
  <a:p>
    <a:pPr …>
      <a:buAutoNum type="arabicPeriod"/>
    </a:pPr>
    …
    <a:t>Bullet 1</a:t>
    …
  </a:p>
  <a:p>
    <a:pPr lvl="1"…>
      <a:buAutoNum type="arabicPeriod"/>
    </a:pPr>
    …
    <a:t>Bullet 2</a:t>
    …
  </a:p>
```

```
    <a:p>
      <a:pPr …>
        <a:buAutoNum type="arabicPeriod"/>
      </a:pPr>
      …
      <a:t>Bullet 3</a:t>
      …
    </a:p>
    …
  </p:txBody>
```

For the above text there are a total of three bullet points. Two of which are at lvl="0" and one at lvl="1". Due to this breakdown of levels, the numbering sequence that should be automatically applied will be 1, 1, 2 as is shown in the picture above. *end example*]

Parent Elements
defPPr (§5.1.5.2.2); lvl1pPr (§5.1.5.4.13); lvl2pPr (§5.1.5.4.14); lvl3pPr (§5.1.5.4.15); lvl4pPr (§5.1.5.4.16); lvl5pPr (§5.1.5.4.17); lvl6pPr (§5.1.5.4.18); lvl7pPr (§5.1.5.4.19); lvl8pPr (§5.1.5.4.20); lvl9pPr (§5.1.5.4.21); pPr (§5.1.5.2.7)

Attributes	Description
startAt (Start Numbering At)	Specifies the number that will start number for a given sequence of automatically numbered bullets. When the numbering is alphabetical, the number should map to the appropriate letter. For instance 1 will map to 'a', 2 to 'b' and so on. If the numbers are larger than 26, then multiple letters should be used. For instance 27 should be represented as 'aa' and similarly 53 should be 'aaa'. The possible values for this attribute are defined by the ST_TextBulletStartAtNum simple type (§5.1.12.63).
type (Bullet Autonumbering Type)	Specifies the numbering scheme that is to be used. This allows for the describing of formats other than strictly numbers. For instance a set of bullets may need to be represented by a series of Roman numerals instead of the standard 1,2,3,etc. number set. The possible values for this attribute are defined by the ST_TextAutonumberScheme simple type (§5.1.12.61).

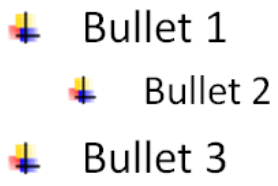
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextAutonumberBullet">
   <attribute name="type" type="ST_TextAutonumberScheme" use="required"/>
   <attribute name="startAt" type="ST_TextBulletStartAtNum" use="optional" default="1"/>
</complexType>
```

5.1.5.4.2 buBlip (Picture Bullet)

This element specifies that a picture be applied to a set of bullets. This element allows for any standard picture format graphic to be used instead of the typical bullet characters. This opens up the possibility for bullets to be anything the generating application would seek to apply.

[*Example*: Consider the DrawingML shown below.




```
<p:txBody>
  …
  <a:p>
    <a:pPr …>
      <a:buBlip>
        <a:blip r:embed="rId2"/>
      </a:buBlip>
    </a:pPr>
    …
    <a:t>Bullet 1</a:t>
    …
  </a:p>
  <a:p>
    <a:pPr lvl="1"…>
      <a:buBlip>
        <a:blip r:embed="rId2"/>
      </a:buBlip>
    </a:pPr>
    …
    <a:t>Bullet 2</a:t>
    …
  </a:p>
```

```
    <a:p>
      <a:pPr …>
        <a:buBlip>
          <a:blip r:embed="rId2"/>
        </a:buBlip>
      </a:pPr>
      …
      <a:t>Bullet 3</a:t>
      …
    </a:p>
    …
  </p:txBody>
```

For the above text there are a total of three bullet points. Two of which are at lvl="0" and one at lvl="1". Because the same picture is specified for each bullet the levels do not stand out here. The only difference is the indentation as shown in the picture above. *end example*]

Parent Elements
defPPr (§5.1.5.2.2); lvl1pPr (§5.1.5.4.13); lvl2pPr (§5.1.5.4.14); lvl3pPr (§5.1.5.4.15); lvl4pPr (§5.1.5.4.16); lvl5pPr (§5.1.5.4.17); lvl6pPr (§5.1.5.4.18); lvl7pPr (§5.1.5.4.19); lvl8pPr (§5.1.5.4.20); lvl9pPr (§5.1.5.4.21); pPr (§5.1.5.2.7)

Child Elements	Subclause
blip (Blip)	§5.1.10.13

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextBlipBullet">
   <sequence>
      <element name="blip" type="CT_Blip" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.5.4.3 buChar (Character Bullet)

This element specifies that a character be applied to a set of bullets. These bullets are allowed to be any character in any font that the system is able to support. If no bullet font is specified along with this element then the paragraph font will be used.

[*Example*: Consider the DrawingML shown below.

g Bullet 1

g Bullet 2

g Bullet 3

```
<p:txBody>
  …
  <a:p>
    <a:pPr …>
      <a:buFont typeface="Calibri"/>
      <a:buChar char="g"/>
    </a:pPr>
    …
    <a:t>Bullet 1</a:t>
    …
  </a:p>
  <a:p>
    <a:pPr lvl="1"…>
      <a:buFont typeface="Calibri"/>
      <a:buChar char="g"/>
    </a:pPr>
    …
    <a:t>Bullet 2</a:t>
    …
  </a:p>
  <a:p>
    <a:pPr …>
      <a:buFont typeface="Calibri"/>
      <a:buChar char="g"/>
    </a:pPr>
    …
    <a:t>Bullet 3</a:t>
    …
  </a:p>
  …
</p:txBody>
```

For the above text there are a total of three bullet points. Two of which are at lvl="0" and one at lvl="1". Because the same character is specified for each bullet the levels do not stand out here. The only difference is the indentation as shown in the picture above. *end example*]

Parent Elements

Parent Elements
defPPr (§5.1.5.2.2); lvl1pPr (§5.1.5.4.13); lvl2pPr (§5.1.5.4.14); lvl3pPr (§5.1.5.4.15); lvl4pPr (§5.1.5.4.16); lvl5pPr (§5.1.5.4.17); lvl6pPr (§5.1.5.4.18); lvl7pPr (§5.1.5.4.19); lvl8pPr (§5.1.5.4.20); lvl9pPr (§5.1.5.4.21); pPr (§5.1.5.2.7)

Attributes	Description
char (Bullet Character)	Specifies the character to be used in place of the standard bullet point. This character may be any character for the specified font that will be supported by the system upon which this document is being viewed. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextCharBullet">
   <attribute name="char" type="xsd:string" use="required"/>
</complexType>
```

5.1.5.4.4 buClr (Color Specified)

This element specifies the color to be used on bullet characters within a given paragraph. The color is specified using the numerical RGB color format.

[*Example*: Consider the DrawingML shown below.

```
<p:txBody>
  …
  <a:p>
    <a:pPr …>
      <a:buClr>
        <a:srgbClr val="FFFF00"/>
      </a:buClr>
    </a:pPr>
    …
    <a:t>Bullet 1</a:t>
    …
  </a:p>
  …
</p:txBody>
```

The color of the above bullet will not follow the text color but instead will have a yellow color specified by val="FFFF00". This color should only apply to the actual bullet character and not to the text within the bullet. *end example*]

Parent Elements

Parent Elements
defPPr (§5.1.5.2.2); lvl1pPr (§5.1.5.4.13); lvl2pPr (§5.1.5.4.14); lvl3pPr (§5.1.5.4.15); lvl4pPr (§5.1.5.4.16); lvl5pPr (§5.1.5.4.17); lvl6pPr (§5.1.5.4.18); lvl7pPr (§5.1.5.4.19); lvl8pPr (§5.1.5.4.20); lvl9pPr (§5.1.5.4.21); pPr (§5.1.5.2.7)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

5.1.5.4.5 buClrTx (Follow Text)

This element specifies that the color of the bullets for a paragraph should be of the same color as the text run within which each bullet is contained.

[*Example*: Consider the DrawingML shown below.

```
<p:txBody>
  …
  <a:p>
    <a:pPr …>
      <a:buClrTx>
    </a:pPr>
    …
    <a:t>Bullet 1</a:t>
    …
  </a:p>
  …
</p:txBody>
```

The color of the above bullet will follow the default text color of the text for the run of text shown above since no specific text color was specified. *end example*]

Parent Elements
defPPr (§5.1.5.2.2); lvl1pPr (§5.1.5.4.13); lvl2pPr (§5.1.5.4.14); lvl3pPr (§5.1.5.4.15); lvl4pPr (§5.1.5.4.16); lvl5pPr (§5.1.5.4.17); lvl6pPr (§5.1.5.4.18); lvl7pPr (§5.1.5.4.19); lvl8pPr (§5.1.5.4.20); lvl9pPr (§5.1.5.4.21); pPr (§5.1.5.2.7)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextBulletColorFollowText"/>
```

5.1.5.4.6 buFont (Specified)

This element specifies the font to be used on bullet characters within a given paragraph. The font is specified using the typeface that it is registered as within the generating application.

[*Example*: Consider the DrawingML shown below.

```
<p:txBody>
  …
  <a:p>
    <a:pPr …>
      <a:buFont typeface="Arial"/>
      <a:buChar char="g"/>
    </a:pPr>
    …
    <a:t>Bullet 1</a:t>
    …
  </a:p>
  …
</p:txBody>
```

The font of the above bullet will not follow the text font but instead will have Arial font specified by typeface="Arial". This font should only apply to the actual bullet character and not to the text within the bullet. *end example*]

Parent Elements
defPPr (§5.1.5.2.2); lvl1pPr (§5.1.5.4.13); lvl2pPr (§5.1.5.4.14); lvl3pPr (§5.1.5.4.15); lvl4pPr (§5.1.5.4.16); lvl5pPr (§5.1.5.4.17); lvl6pPr (§5.1.5.4.18); lvl7pPr (§5.1.5.4.19); lvl8pPr (§5.1.5.4.20); lvl9pPr (§5.1.5.4.21); pPr (§5.1.5.2.7)

Attributes	Description
charset (Similar Character Set)	Specifies the most similar character set to the one being used. This is useful if the generating application cannot use the current font and must choose a similar one. The possible values for this attribute are defined by the XML Schema byte datatype.
panose (Panose	Specifies the panose standard setting that will be used to determine the closest matching

Attributes	Description
Setting)	font by any generating application that employs this method. The possible values for this attribute are defined by the ST_Panose simple type (§5.1.12.37).
pitchFamily (Similar Font Family)	Specifies the most similar font family to the one being used. This is useful if the generating application cannot use the current font and must choose a similar one. The possible values for this attribute are defined by the XML Schema byte datatype.
typeface (Text Typeface)	Specifies the typeface, or name of the font that is to be used for a bullet. The typeface used here should be selected from the font list of the generating application. The possible values for this attribute are defined by the ST_TextTypeface simple type (§5.1.12.81).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextFont">
   <attribute name="typeface" type="ST_TextTypeface"/>
   <attribute name="panose" type="ST_Panose" use="optional"/>
   <attribute name="pitchFamily" type="xsd:byte" use="optional" default="0"/>
   <attribute name="charset" type="xsd:byte" use="optional" default="1"/>
</complexType>
```

5.1.5.4.7 buFontTx (Follow text)

This element specifies that the font of the bullets for a paragraph should be of the same font as the text run within which each bullet is contained.

[*Example*: Consider the DrawingML shown below.

```
<p:txBody>
  …
  <a:p>
    <a:pPr …>
      <a:buFontTx>
    </a:pPr>
    …
    <a:t>Bullet 1</a:t>
    …
  </a:p>
  …
</p:txBody>
```

The font of the above bullet will follow the default text font of the text for the run of text shown above since no specific text font was specified. *end example*]

Parent Elements
defPPr (§5.1.5.2.2); lvl1pPr (§5.1.5.4.13); lvl2pPr (§5.1.5.4.14); lvl3pPr (§5.1.5.4.15); lvl4pPr (§5.1.5.4.16); lvl5pPr (§5.1.5.4.17); lvl6pPr (§5.1.5.4.18); lvl7pPr (§5.1.5.4.19); lvl8pPr (§5.1.5.4.20); lvl9pPr (§5.1.5.4.21); pPr (§5.1.5.2.7)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextBulletTypefaceFollowText"/>
```

5.1.5.4.8 buNone (No Bullet)

This element specifies that the paragraph within which it is applied is to have no bullet formatting applied to it. That is to say that there should be no bulleting found within the paragraph where this element is specified.

[*Example*: Consider the DrawingML shown below.

```
<p:txBody>
  …
  <a:p>
    <a:pPr …>
      <a:buNone/>
    </a:pPr>
    …
    <a:t>Bullet 1</a:t>
    …
  </a:p>
  …
</p:txBody>
```

The above paragraph will be formatted with no bullets. *end example*]

Parent Elements
defPPr (§5.1.5.2.2); lvl1pPr (§5.1.5.4.13); lvl2pPr (§5.1.5.4.14); lvl3pPr (§5.1.5.4.15); lvl4pPr (§5.1.5.4.16); lvl5pPr (§5.1.5.4.17); lvl6pPr (§5.1.5.4.18); lvl7pPr (§5.1.5.4.19); lvl8pPr (§5.1.5.4.20); lvl9pPr (§5.1.5.4.21); pPr (§5.1.5.2.7)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextNoBullet"/>
```

5.1.5.4.9 buSzPct (Bullet Size Percentage)

This element specifies the size in percentage of the surrounding text to be used on bullet characters within a given paragraph. The size is specified using a percentage where 1000 is equal to 1 percent of the font size and 100000 is equal to 100 percent font of the font size.

[*Example*: Consider the DrawingML shown below.

```
<p:txBody>
  …
  <a:p>
    <a:pPr …>
      <a:buSzPct val="111000"/>
    </a:pPr>
    …
    <a:t>Bullet 1</a:t>
    …
  </a:p>
  …
</p:txBody>
```

The size of the above bullet will follow the text size in that it will always be rendered at 111% the size of the text within the given text run. This is specified by val="111000", with a restriction on the values not being less than 25% or more than 400%. A value of 100000 is equal to 100%, similarly a value of 1000 is equal to 1%. This percentage size should only apply to the actual bullet character and not to the text within the bullet. *end example*]

Parent Elements
defPPr (§5.1.5.2.2); lvl1pPr (§5.1.5.4.13); lvl2pPr (§5.1.5.4.14); lvl3pPr (§5.1.5.4.15); lvl4pPr (§5.1.5.4.16); lvl5pPr (§5.1.5.4.17); lvl6pPr (§5.1.5.4.18); lvl7pPr (§5.1.5.4.19); lvl8pPr (§5.1.5.4.20); lvl9pPr (§5.1.5.4.21); pPr (§5.1.5.2.7)

Attributes	Description
val (Value)	Specifies the percentage of the text size that this bullet should be. It is specified here in terms of 100% being equal to 100000 and 1% being specified in increments of 1000. This attribute should not be lower than 25%, or 25000 and not be higher than 400%, or 400000. The possible values for this attribute are defined by the ST_TextBulletSizePercent simple type (§5.1.12.62).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextBulletSizePercent">
   <attribute name="val" type="ST_TextBulletSizePercent"/>
</complexType>
```

5.1.5.4.10 buSzPts (Bullet Size Points)

This element specifies the size in points to be used on bullet characters within a given paragraph. The size is specified using the points where 100 is equal to 1 point font and 1200 is equal to 12 point font.

[*Example*: Consider the DrawingML shown below.

```
<p:txBody>
  …
  <a:p>
    <a:pPr …>
      <a:buSzPts val="1400"/>
    </a:pPr>
    …
    <a:t>Bullet 1</a:t>
    …
  </a:p>
  …
</p:txBody>
```

The size of the above bullet will not follow the text size of the text within the given text run. The bullets size is specified by val="1400", which corresponds to a point size of 14. This bullet size should only apply to the actual bullet character and not to the text within the bullet. *end example*]

Parent Elements
defPPr (§5.1.5.2.2); lvl1pPr (§5.1.5.4.13); lvl2pPr (§5.1.5.4.14); lvl3pPr (§5.1.5.4.15); lvl4pPr (§5.1.5.4.16); lvl5pPr (§5.1.5.4.17); lvl6pPr (§5.1.5.4.18); lvl7pPr (§5.1.5.4.19); lvl8pPr (§5.1.5.4.20); lvl9pPr (§5.1.5.4.21); pPr (§5.1.5.2.7)

Attributes	Description
val (Value)	Specifies the size of the bullets in point size. Whole points are specified in increments of 100 starting with 100 being a point size of 1. For instance a font point size of 12 would be 1200 and a font point size of 12.5 would be 1250. The possible values for this attribute are defined by the ST_TextFontSize simple type (§5.1.12.68).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextBulletSizePoint">
   <attribute name="val" type="ST_TextFontSize"/>
</complexType>
```

5.1.5.4.11 buSzTx (Bullet Size Follows Text)

This element specifies that the size of the bullets for a paragraph should be of the same point size as the text run within which each bullet is contained.

[*Example*: Consider the DrawingML shown below.

```
<p:txBody>
  …
  <a:p>
    <a:pPr …>
      <a:buSzTx>
    </a:pPr>
    …
    <a:t>Bullet 1</a:t>
    …
  </a:p>
  …
</p:txBody>
```

The size of the above bullet will follow the default text size of the text for the run of text shown above since no specific text size was specified. *end example*]

Parent Elements
defPPr (§5.1.5.2.2); lvl1pPr (§5.1.5.4.13); lvl2pPr (§5.1.5.4.14); lvl3pPr (§5.1.5.4.15); lvl4pPr (§5.1.5.4.16); lvl5pPr (§5.1.5.4.17); lvl6pPr (§5.1.5.4.18); lvl7pPr (§5.1.5.4.19); lvl8pPr (§5.1.5.4.20); lvl9pPr (§5.1.5.4.21); pPr (§5.1.5.2.7)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextBulletSizeFollowText"/>
```

5.1.5.4.12 lstStyle (Text List Styles)

This element specifies the list of styles associated with this body of text.

Parent Elements
lnDef (§5.1.4.1.20); rich (§5.7.2.157); spDef (§5.1.4.1.27); t (§5.9.3.8); txBody (§5.8.2.26); txBody (§5.1.2.1.40); txBody (§5.6.2.33); txBody (§4.4.1.47); txDef (§5.1.4.1.28); txPr (§5.7.2.217)

Child Elements	Subclause
defPPr (Default Paragraph Style)	§5.1.5.2.2
extLst (Extension List)	§5.1.2.1.15
lvl1pPr (List Level 1 Text Style)	§5.1.5.4.13
lvl2pPr (List Level 2 Text Style)	§5.1.5.4.14
lvl3pPr (List Level 3 Text Style)	§5.1.5.4.15
lvl4pPr (List Level 4 Text Style)	§5.1.5.4.16
lvl5pPr (List Level 5 Text Style)	§5.1.5.4.17
lvl6pPr (List Level 6 Text Style)	§5.1.5.4.18

Child Elements	Subclause
lvl7pPr (List Level 7 Text Style)	§5.1.5.4.19
lvl8pPr (List Level 8 Text Style)	§5.1.5.4.20
lvl9pPr (List Level 9 Text Style)	§5.1.5.4.21

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextListStyle">
   <sequence>
      <element name="defPPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl1pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl2pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl3pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl4pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl5pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl6pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl7pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl8pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lvl9pPr" type="CT_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.5.4.13 lvl1pPr (List Level 1 Text Style)

This element specifies all paragraph level text properties for all elements that have the attribute lvl="0". There are a total of 9 level text property elements allowed, levels 0-8. It is recommended that the order in which this and other level property elements are specified be in order of increasing level. That is lvl2pPr should come before lvl3pPr. This allows the lower level properties to take precedence over the higher level ones because they are parsed first.

[*Example*: Consider the following DrawingML code that would specify a paragraph to follow the level style defined in lvl1pPr and thus create a paragraph of text that has no bullets and is right aligned.

```
<p:txBody>
  …
  <a:lstStyle>
    <a:lvl1pPr algn="r">
      <a:buNone/>
    </a:lvl1pPr>
  </a:lstStyle>
  <a:p>
    <a:pPr lvl="0">
    </a:pPr>
    …
    <a:t>Some text</a:t>
    …
  </a:p>
</p:txBody>
```

end example]

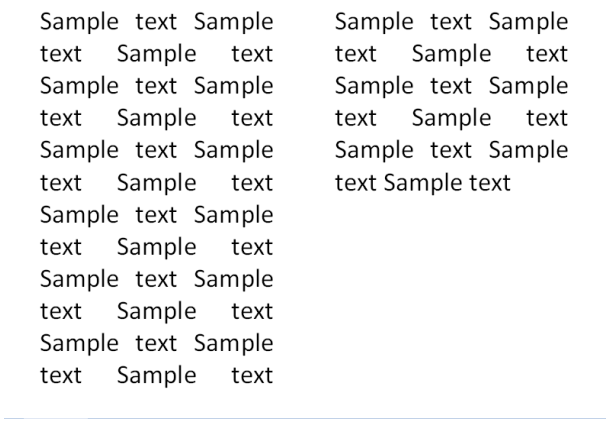
[*Note*: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The rule here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. *end note*]

Parent Elements
bodyStyle (§4.4.1.5); defaultTextStyle (§4.3.1.7); lstStyle (§5.1.5.4.12); notesStyle (§4.4.1.25); otherStyle (§4.4.1.32); titleStyle (§4.4.1.45)

Child Elements	Subclause
buAutoNum (Auto-Numbered Bullet)	§5.1.5.4.1
buBlip (Picture Bullet)	§5.1.5.4.2
buChar (Character Bullet)	§5.1.5.4.3
buClr (Color Specified)	§5.1.5.4.4
buClrTx (Follow Text)	§5.1.5.4.5
buFont (Specified)	§5.1.5.4.6
buFontTx (Follow text)	§5.1.5.4.7
buNone (No Bullet)	§5.1.5.4.8
buSzPct (Bullet Size Percentage)	§5.1.5.4.9
buSzPts (Bullet Size Points)	§5.1.5.4.10
buSzTx (Bullet Size Follows Text)	§5.1.5.4.11

Child Elements	Subclause
defRPr (Default Text Run Properties)	§5.1.5.3.2
extLst (Extension List)	§5.1.2.1.15
lnSpc (Line Spacing)	§5.1.5.2.5
spcAft (Space After)	§5.1.5.2.8
spcBef (Space Before)	§5.1.5.2.9
tabLst (Tab List)	§5.1.5.2.13

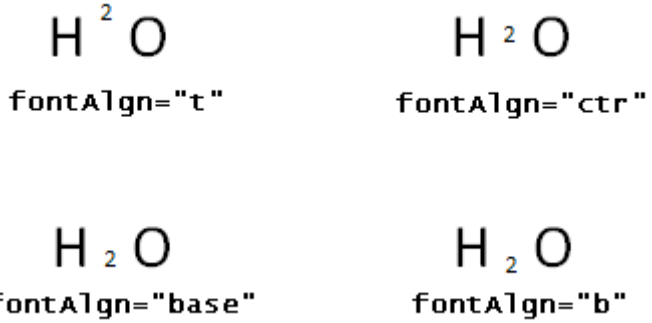
Attributes	Description

algn (Alignment)	Specifies the alignment that is to be applied to the paragraph. Possible values for this include left, right, centered, justified and distributed. If this attribute is omitted, then a value of left is implied.
	
	 
	
	[*Example*: Consider the case where the user wishes to have two columns of text that have a justified alignment, much like text within a book. The following DrawingML could describe this.
	
	`<p:txBody>`
	`<a:bodyPr numCol="2" spcCol="914400"…/>`
	`<a:normAutofit/>`
	`</a:bodyPr>`
	`…`
	`<a:p>`
	`<a:pPr marL="0" algn="just">`
	`<a:buNone/>`
	`</a:pPr>`
	`…`
	`<a:t>Sample Text …</a:t>`
	`…`
	`</a:p>`
	`</p:txBody>`

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_TextAlignType simple type (§5.1.12.59).

defTabSz (Default Tab Size)	Specifies the default size for a tab character within this paragraph. This attribute should be used to describe the spacing of tabs within the paragraph instead of a leading indentation tab. For indentation tabs there are the marL and indent attributes to assist with this.
	
	[*Example*: Consider the case where a paragraph contains numerous tabs that need to be of a specific size. The following DrawingML would describe this.
	
	`<p:txBody>`
	`  …`
	`  <a:p>`
	`  <a:pPr defTabSz="376300" …/>`
	`  …`
	`  <a:t>Sample Text …</a:t>`
	`  …`
	`  </a:p>`
	`</p:txBody>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_Coordinate32 simple type (§5.1.12.17).

eaLnBrk (East Asian Line Break)	Specifies whether an East Asian word can be broken in half and wrapped onto the next line without a hyphen being added. To determine whether an East Asian word can be broken the presentation application would use the kinsoku settings here. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable East Asian words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.
	
	[*Example*: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in whole on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.
	
	`<p:txBody>`
	`  …`
	`  <a:p>`
	`  <a:pPr eaLnBrk="0" …/>`

Attributes	Description
	… <a:t>Sample text (Long word)</a:t> … </a:p> </p:txBody> Sample text Sample text Sample text supercalifr agilisticexpialidocious Sample text Sample text Sample text supercalifragilisticexpialidocious *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
fontAlgn (Font Alignment)	Determines where vertically on a line of text the actual words are positioned. This deals with vertical placement of the characters with respect to the baselines. For instance having text anchored to the top baseline, anchored to the bottom baseline, centered in between, etc. To understand this attribute and it's use it is helpful to understand what baselines are. A diagram describing these different cases is shown below. If this attribute is omitted, then a value of base is implied. [*Example*: Consider the case where the user wishes to represent the chemical compound of a water molecule. For this they will need to make sure the H, the 2, and the O are all in the correct position and are of the correct size. The results below can be achieved through the DrawingML shown below.  fontAlgn="t" fontAlgn="ctr" fontAlgn="base" fontAlgn="b" <a:txtBody> … <a:pPr fontAlgn="b" …/> … <a:r> <a:rPr …/> <a:t>H </a:t>

Attributes	Description
	</a:r> <a:r> <a:rPr sz="1200" …/> <a:t>2</a:t> </a:r> <a:r> <a:rPr …/> <a:t>O</a:t> </a:r> … </p:txBody> *end example*] The possible values for this attribute are defined by the ST_TextFontAlignType simple type (§5.1.12.66).
hangingPunct (Hanging Punctuation)	Specifies whether punctuation is to be forcefully laid out on a line of text or put on a different line of text. That is, if there is punctuation at the end of a run of text that should be carried over to a separate line does it actually get carried over. A true value will allow for hanging punctuation forcing the punctuation to not be carried over and a value of false will allow the punctuation to be carried onto the next text line. If this attribute is omitted, then a value of 0, or false is implied. The possible values for this attribute are defined by the XML Schema boolean datatype.
indent (Indent)	Specifies the indent size that will be applied to the first line of text in the paragraph. An indentation of 0 will be considered to be at the same location as marL attribute. If this attribute is omitted, then a value of -342900 is implied. Here is some Sample text Sample text Sample text Sample text Sample text Sample text Sample text Sample text Sample text Sample text Sample text Sample text Sample text Sample text Sample text Sample text Sample text Sample text Sample text Sample text Sample text [*Example*: Consider the scenario where the user now wanted to add a paragraph indentation to the first line of text in their two column format book. <p:txBody>

Attributes	Description
	<a:bodyPr numCol="2" spcCol="914400"…/> <a:normAutofit/> </a:bodyPr> … <a:p> <a:pPr marL="0" indent="571500" algn="just"> <a:buNone/> </a:pPr> … <a:t>Here is some…</a:t> … </a:p> </p:txBody> By adding the indent attribute the user has effectively added a first line indent to this paragraph of text. *end example*] The possible values for this attribute are defined by the ST_TextIndent simple type (§5.1.12.70).
latinLnBrk (Latin Line Break)	Specifies whether a Latin word can be broken in half and wrapped onto the next line without a hyphen being added. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable Latin words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied. [*Example*: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in whole on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows. <p:txBody> … <a:p> <a:pPr latinLnBrk="0" …/> … <a:t>Sample text (Long word)</a:t> … </a:p> </p:txBody> Sample text Sample text Sample text supercalifr agilisticexpialidocious

Attributes	Description
	Sample text Sample text Sample text supercalifragilisticexpialidocious *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.

lvl (Level)	Specifies the particular level text properties that this paragraph will follow. The value for this attribute is numerical and formats the text according to the corresponding level paragraph properties that are listed within the lstStyle element. Since there are nine separate level properties defined, this tag will have an effective range of 0-8 = 9 available values.
	
	[*Example*: Consider the following DrawingML. This would specify that this paragraph should follow the lvl2pPr formatting style because once again lvl="1" is considered to be level 2.
	
	`<p:txBody>`
	`  …`
	`  <a:p>`
	`  <a:pPr lvl="1" …/>`
	`  …`
	`  <a:t>Sample text</a:t>`
	`  …`
	`  </a:p>`
	`</p:txBody>`
	
	end example]
	
	[*Note*: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The rule here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. *end note*]
	
	The possible values for this attribute are defined by the ST_TextIndentLevelType simple type (§5.1.12.71).

marL (Left Margin)	Specifies the left margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marL attributes are additive with respect to the text position. If this attribute is omitted, then a value of 347663 is implied. The possible values for this attribute are defined by the ST_TextMargin simple type (§5.1.12.73).
marR (Right Margin)	Specifies the right margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marR

Attributes	Description
	attributes are additive with respect to the text position. If this attribute is omitted, then a value of 0 is implied. The possible values for this attribute are defined by the ST_TextMargin simple type (§5.1.12.73).

rtl (Right To Left)	Specifies whether the text is right-to-left or left-to-right in its flow direction. If this attribute is omitted, then a value of 0, or left-to-right is implied.
	
	[*Example*: Consider the scenario where the user wanted text to flow from right to left. That is within the bounding text box the first word would be right aligned with each additional word being written to the left of the previous while continuing to flow top to bottom. The DrawingML to describe this might be as follows.
	
	`<p:txBody>`
	`…`
	`<a:p>`
	`<a:pPr rtl="1" …/>`
	`…`
	`<a:t>Sample text…</a:t>`
	`…`
	`</a:p>`
	`</p:txBody>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextParagraphProperties">
   <sequence>
      <element name="lnSpc" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <element name="spcBef" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <element name="spcAft" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletColor" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletSize" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletTypeface" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBullet" minOccurs="0" maxOccurs="1"/>
      <element name="tabLst" type="CT_TextTabStopList" minOccurs="0" maxOccurs="1"/>
      <element name="defRPr" type="CT_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="marL" type="ST_TextMargin" use="optional"/>
   <attribute name="marR" type="ST_TextMargin" use="optional"/>
   <attribute name="lvl" type="ST_TextIndentLevelType" use="optional"/>
   <attribute name="indent" type="ST_TextIndent" use="optional"/>
   <attribute name="algn" type="ST_TextAlignType" use="optional"/>
   <attribute name="defTabSz" type="ST_Coordinate32" use="optional"/>
   <attribute name="rtl" type="xsd:boolean" use="optional"/>
   <attribute name="eaLnBrk" type="xsd:boolean" use="optional"/>
   <attribute name="fontAlgn" type="ST_TextFontAlignType" use="optional"/>
   <attribute name="latinLnBrk" type="xsd:boolean" use="optional"/>
   <attribute name="hangingPunct" type="xsd:boolean" use="optional"/>
</complexType>
```

5.1.5.4.14 lvl2pPr (List Level 2 Text Style)

This element specifies all paragraph level text properties for all elements that have the attribute lvl="1". There are a total of 9 level text property elements allowed, levels 0-8. It is recommended that the order in which this and other level property elements are specified be in order of increasing level. That is lvl2pPr should come before lvl3pPr. This allows the lower level properties to take precedence over the higher level ones because they are parsed first.

[*Example*: Consider the following DrawingML code that would specify a paragraph to follow the level style defined in lvl2pPr and thus create a paragraph of text that has no bullets and is right aligned.

```
<p:txBody>
  …
  <a:lstStyle>
    <a:lvl2pPr algn="r">
      <a:buNone/>
    </a:lvl2pPr>
  </a:lstStyle>
```

```
    <a:p>
      <a:pPr lvl="1">
      </a:pPr>
      …
      <a:t>Some text</a:t>
      …
    </a:p>
  </p:txBody>
```

end example]

[*Note*: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The rule here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. *end note*]

Parent Elements
bodyStyle (§4.4.1.5); defaultTextStyle (§4.3.1.7); lstStyle (§5.1.5.4.12); notesStyle (§4.4.1.25); otherStyle (§4.4.1.32); titleStyle (§4.4.1.45)

Child Elements	Subclause
buAutoNum (Auto-Numbered Bullet)	§5.1.5.4.1
buBlip (Picture Bullet)	§5.1.5.4.2
buChar (Character Bullet)	§5.1.5.4.3
buClr (Color Specified)	§5.1.5.4.4
buClrTx (Follow Text)	§5.1.5.4.5
buFont (Specified)	§5.1.5.4.6
buFontTx (Follow text)	§5.1.5.4.7
buNone (No Bullet)	§5.1.5.4.8
buSzPct (Bullet Size Percentage)	§5.1.5.4.9
buSzPts (Bullet Size Points)	§5.1.5.4.10
buSzTx (Bullet Size Follows Text)	§5.1.5.4.11
defRPr (Default Text Run Properties)	§5.1.5.3.2
extLst (Extension List)	§5.1.2.1.15
lnSpc (Line Spacing)	§5.1.5.2.5
spcAft (Space After)	§5.1.5.2.8
spcBef (Space Before)	§5.1.5.2.9
tabLst (Tab List)	§5.1.5.2.13

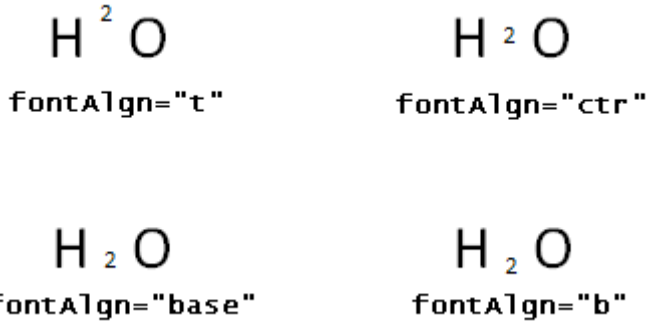
Attributes	Description
algn (Alignment)	Specifies the alignment that is to be applied to the paragraph. Possible values for this include left, right, centered, justified and distributed. If this attribute is omitted, then a value of left is implied. Sample text [*Example*: Consider the case where the user wishes to have two columns of text that have a justified alignment, much like text within a book. The following DrawingML could describe this. `<p:txBody>` `  <a:bodyPr numCol="2" spcCol="914400"…/>` `    <a:normAutofit/>` `    </a:bodyPr>` `  …` `  <a:p>` `    <a:pPr marL="0" algn="just">` `      <a:buNone/>` `    </a:pPr>` `    …` `    <a:t>Sample Text …</a:t>` `    …` `  </a:p>` `</p:txBody>` *end example*] The possible values for this attribute are defined by the ST_TextAlignType simple type (§5.1.12.59).
defTabSz (Default Tab Size)	Specifies the default size for a tab character within this paragraph. This attribute should be used to describe the spacing of tabs within the paragraph instead of a leading indentation tab. For indentation tabs there are the marL and indent attributes to assist with this.

Attributes	Description

	[*Example*: Consider the case where a paragraph contains numerous tabs that need to be of a specific size. The following DrawingML would describe this.
	
	`<p:txBody>`
	`  …`
	`  <a:p>`
	`  <a:pPr defTabSz="376300" …/>`
	`  …`
	`  <a:t>Sample Text …</a:t>`
	`  …`
	`  </a:p>`
	`</p:txBody>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_Coordinate32 simple type (§5.1.12.17).

eaLnBrk (East Asian Line Break)	Specifies whether an East Asian word can be broken in half and wrapped onto the next line without a hyphen being added. To determine whether an East Asian word can be broken the presentation application would use the kinsoku settings here. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable East Asian words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.
	
	[*Example*: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in whole on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.
	
	`<p:txBody>`
	`  …`
	`  <a:p>`
	`  <a:pPr eaLnBrk="0" …/>`
	`  …`
	`  <a:t>Sample text (Long word)</a:t>`
	`  …`
	`  </a:p>`
	`</p:txBody>`
	
	Sample text Sample text Sample text supercalifr
	agilisticexpialidocious

Attributes	Description
	Sample text Sample text Sample text supercalifragilisticexpialidocious *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.

fontAlgn (Font Alignment)	Determines where vertically on a line of text the actual words are positioned. This deals with vertical placement of the characters with respect to the baselines. For instance having text anchored to the top baseline, anchored to the bottom baseline, centered in between, etc. To understand this attribute and it's use it is helpful to understand what baselines are. A diagram describing these different cases is shown below. If this attribute is omitted, then a value of base is implied.
	
	[*Example*: Consider the case where the user wishes to represent the chemical compound of a water molecule. For this they will need to make sure the H, the 2, and the O are all in the correct position and are of the correct size. The results below can be achieved through the DrawingML shown below.
	
	 
	
	`<a:txtBody>`
	`…`
	`<a:pPr fontAlgn="b" …/>`
	`…`
	`<a:r>`
	`<a:rPr …/>`
	`<a:t>H </a:t>`
	`</a:r>`
	`<a:r>`
	`<a:rPr sz="1200" …/>`
	`<a:t>2</a:t>`
	`</a:r>`
	`<a:r>`
	`<a:rPr …/>`
	`<a:t>O</a:t>`
	`</a:r>`

Attributes	Description
	… </p:txBody> *end example*] The possible values for this attribute are defined by the ST_TextFontAlignType simple type (§5.1.12.66).
hangingPunct (Hanging Punctuation)	Specifies whether punctuation is to be forcefully laid out on a line of text or put on a different line of text. That is, if there is punctuation at the end of a run of text that should be carried over to a separate line does it actually get carried over. A true value will allow for hanging punctuation forcing the punctuation to not be carried over and a value of false will allow the punctuation to be carried onto the next text line. If this attribute is omitted, then a value of 0, or false is implied. The possible values for this attribute are defined by the XML Schema boolean datatype.

indent (Indent)	Specifies the indent size that will be applied to the first line of text in the paragraph. An indentation of 0 will be considered to be at the same location as marL attribute. If this attribute is omitted, then a value of -342900 is implied.
	
	Here is some Sample text
	
	[*Example*: Consider the scenario where the user now wanted to add a paragraph indentation to the first line of text in their two column format book.
	
	<p:txBody>
	  <a:bodyPr numCol="2" spcCol="914400"…/>
	    <a:normAutofit/>
	    </a:bodyPr>
	  …
	  <a:p>
	    <a:pPr marL="0" indent="571500" algn="just">
	      <a:buNone/>
	    </a:pPr>
	    …

Attributes	Description
	<a:t>Here is some…</a:t> … </a:p> </p:txBody> By adding the indent attribute the user has effectively added a first line indent to this paragraph of text. *end example*] The possible values for this attribute are defined by the ST_TextIndent simple type (§5.1.12.70).
latinLnBrk (Latin Line Break)	Specifies whether a Latin word can be broken in half and wrapped onto the next line without a hyphen being added. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable Latin words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied. [*Example*: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in whole on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows. <p:txBody> … <a:p> <a:pPr latinLnBrk="0" …/> … <a:t>Sample text (Long word)</a:t> … </a:p> </p:txBody> Sample text Sample text Sample text supercalifr agilisticexpialidocious Sample text Sample text Sample text supercalifragilisticexpialidocious *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
lvl (Level)	Specifies the particular level text properties that this paragraph will follow. The value for this attribute is numerical and formats the text according to the corresponding level paragraph properties that are listed within the lstStyle element. Since there are nine

Attributes	Description

	separate level properties defined, this tag will have an effective range of 0-8 = 9 available values.
	
	[*Example*: Consider the following DrawingML. This would specify that this paragraph should follow the lvl2pPr formatting style because once again lvl="1" is considered to be level 2.
	
	`<p:txBody>`
	`…`
	`<a:p>`
	`<a:pPr lvl="1" …/>`
	`…`
	`<a:t>Sample text</a:t>`
	`…`
	`</a:p>`
	`</p:txBody>`
	
	end example]
	
	[*Note*: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The rule here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. *end note*]
	
	The possible values for this attribute are defined by the ST_TextIndentLevelType simple type (§5.1.12.71).

marL (Left Margin)	Specifies the left margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marL attributes are additive with respect to the text position. If this attribute is omitted, then a value of 347663 is implied. The possible values for this attribute are defined by the ST_TextMargin simple type (§5.1.12.73).
marR (Right Margin)	Specifies the right margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marR attributes are additive with respect to the text position. If this attribute is omitted, then a value of 0 is implied. The possible values for this attribute are defined by the ST_TextMargin simple type (§5.1.12.73).
rtl (Right To Left)	Specifies whether the text is right-to-left or left-to-right in its flow direction. If this attribute is omitted, then a value of 0, or left-to-right is implied.

Attributes	Description

	[*Example*: Consider the scenario where the user wanted text to flow from right to left. That is within the bounding text box the first word would be right aligned with each additional word being written to the left of the previous while continuing to flow top to bottom. The DrawingML to describe this might be as follows.
	
	`<p:txBody>`
	`…`
	`<a:p>`
	`<a:pPr rtl="1" …/>`
	`…`
	`<a:t>Sample text…</a:t>`
	`…`
	`</a:p>`
	`</p:txBody>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextParagraphProperties">
   <sequence>
      <element name="lnSpc" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <element name="spcBef" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <element name="spcAft" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletColor" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletSize" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletTypeface" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBullet" minOccurs="0" maxOccurs="1"/>
      <element name="tabLst" type="CT_TextTabStopList" minOccurs="0" maxOccurs="1"/>
      <element name="defRPr" type="CT_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="marL" type="ST_TextMargin" use="optional"/>
   <attribute name="marR" type="ST_TextMargin" use="optional"/>
   <attribute name="lvl" type="ST_TextIndentLevelType" use="optional"/>
   <attribute name="indent" type="ST_TextIndent" use="optional"/>
   <attribute name="algn" type="ST_TextAlignType" use="optional"/>
   <attribute name="defTabSz" type="ST_Coordinate32" use="optional"/>
   <attribute name="rtl" type="xsd:boolean" use="optional"/>
   <attribute name="eaLnBrk" type="xsd:boolean" use="optional"/>
   <attribute name="fontAlgn" type="ST_TextFontAlignType" use="optional"/>
   <attribute name="latinLnBrk" type="xsd:boolean" use="optional"/>
   <attribute name="hangingPunct" type="xsd:boolean" use="optional"/>
</complexType>
```

5.1.5.4.15 lvl3pPr (List Level 3 Text Style)

This element specifies all paragraph level text properties for all elements that have the attribute lvl="2". There are a total of 9 level text property elements allowed, levels 0-8. It is recommended that the order in which this and other level property elements are specified be in order of increasing level. That is lvl2pPr should come before lvl3pPr. This allows the lower level properties to take precedence over the higher level ones because they are parsed first.

[*Example*: Consider the following DrawingML code that would specify a paragraph to follow the level style defined in lvl3pPr and thus create a paragraph of text that has no bullets and is right aligned.

```
<p:txBody>
  …
  <a:lstStyle>
    <a:lvl3pPr algn="r">
      <a:buNone/>
    </a:lvl3pPr>
  </a:lstStyle>
  <a:p>
    <a:pPr lvl="2">
    </a:pPr>
    …
    <a:t>Some text</a:t>
    …
  </a:p>
</p:txBody>
```

end example]

[*Note*: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The rule here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. *end note*]

Parent Elements
bodyStyle (§4.4.1.5); defaultTextStyle (§4.3.1.7); lstStyle (§5.1.5.4.12); notesStyle (§4.4.1.25); otherStyle (§4.4.1.32); titleStyle (§4.4.1.45)

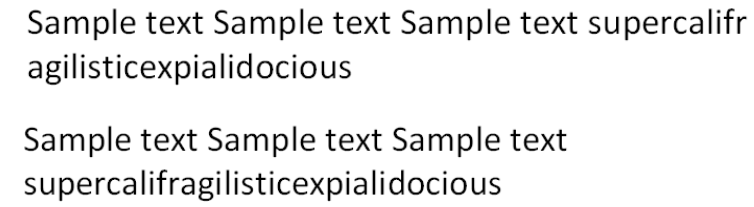
Child Elements	Subclause
buAutoNum (Auto-Numbered Bullet)	§5.1.5.4.1
buBlip (Picture Bullet)	§5.1.5.4.2
buChar (Character Bullet)	§5.1.5.4.3

Child Elements	Subclause
buClr (Color Specified)	§5.1.5.4.4
buClrTx (Follow Text)	§5.1.5.4.5
buFont (Specified)	§5.1.5.4.6
buFontTx (Follow text)	§5.1.5.4.7
buNone (No Bullet)	§5.1.5.4.8
buSzPct (Bullet Size Percentage)	§5.1.5.4.9
buSzPts (Bullet Size Points)	§5.1.5.4.10
buSzTx (Bullet Size Follows Text)	§5.1.5.4.11
defRPr (Default Text Run Properties)	§5.1.5.3.2
extLst (Extension List)	§5.1.2.1.15
lnSpc (Line Spacing)	§5.1.5.2.5
spcAft (Space After)	§5.1.5.2.8
spcBef (Space Before)	§5.1.5.2.9
tabLst (Tab List)	§5.1.5.2.13

Attributes	Description
algn (Alignment)	Specifies the alignment that is to be applied to the paragraph. Possible values for this include left, right, centered, justified and distributed. If this attribute is omitted, then a value of left is implied. Sample text [*Example*: Consider the case where the user wishes to have two columns of text that have a justified alignment, much like text within a book. The following DrawingML could describe this. `<p:txBody>` `  <a:bodyPr numCol="2" spcCol="914400"…/>`

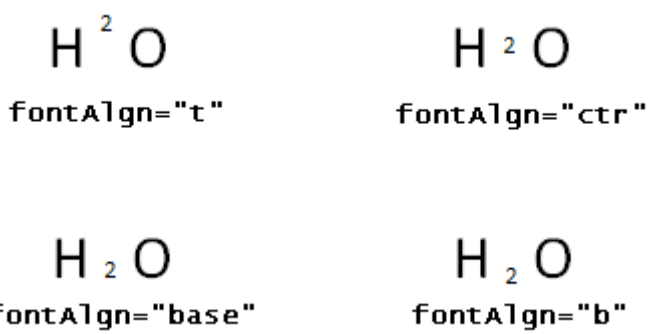
Attributes	Description
	<a:normAutofit/> </a:bodyPr> … <a:p> <a:pPr marL="0" algn="just"> <a:buNone/> </a:pPr> … <a:t>Sample Text …</a:t> … </a:p> </p:txBody> *end example*] The possible values for this attribute are defined by the ST_TextAlignType simple type (§5.1.12.59).
defTabSz (Default Tab Size)	Specifies the default size for a tab character within this paragraph. This attribute should be used to describe the spacing of tabs within the paragraph instead of a leading indentation tab. For indentation tabs there are the marL and indent attributes to assist with this. [*Example*: Consider the case where a paragraph contains numerous tabs that need to be of a specific size. The following DrawingML would describe this. <p:txBody> … <a:p> <a:pPr defTabSz="376300" …/> … <a:t>Sample Text …</a:t> … </a:p> </p:txBody> *end example*] The possible values for this attribute are defined by the ST_Coordinate32 simple type (§5.1.12.17).
eaLnBrk (East Asian Line Break)	Specifies whether an East Asian word can be broken in half and wrapped onto the next line without a hyphen being added. To determine whether an East Asian word can be broken the presentation application would use the kinsoku settings here. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable East Asian words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.

Attributes	Description

	[*Example*: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in whole on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.
	
	`<p:txBody>`
	`  …`
	`  <a:p>`
	`  <a:pPr eaLnBrk="0" …/>`
	`  …`
	`  <a:t>Sample text (Long word)</a:t>`
	`  …`
	`  </a:p>`
	`</p:txBody>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

fontAlgn (Font Alignment)	Determines where vertically on a line of text the actual words are positioned. This deals with vertical placement of the characters with respect to the baselines. For instance having text anchored to the top baseline, anchored to the bottom baseline, centered in between, etc. To understand this attribute and it's use it is helpful to understand what baselines are. A diagram describing these different cases is shown below. If this attribute is omitted, then a value of base is implied. [*Example*: Consider the case where the user wishes to represent the chemical compound of a water molecule. For this they will need to make sure the H, the 2, and the O are all in the correct position and are of the correct size. The results below can be achieved through the DrawingML shown below.

Attributes	Description

	
	
	``
	<a:txtBody>
	  …
	  <a:pPr fontAlgn="b" …/>
	  …
	  <a:r>
	  <a:rPr …/>
	  <a:t>H </a:t>
	  </a:r>
	  <a:r>
	  <a:rPr sz="1200" …/>
	  <a:t>2</a:t>
	  </a:r>
	  <a:r>
	  <a:rPr …/>
	  <a:t>O</a:t>
	  </a:r>
	  …
	</p:txBody>
	``
	
	end example]
	
	The possible values for this attribute are defined by the ST_TextFontAlignType simple type (§5.1.12.66).

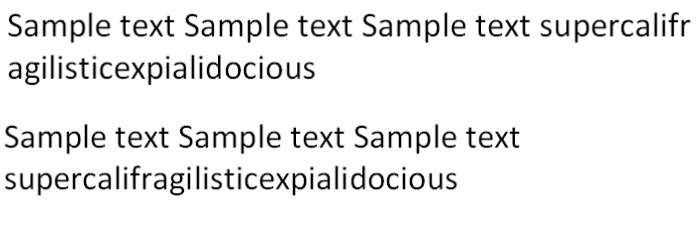
hangingPunct (Hanging Punctuation)	Specifies whether punctuation is to be forcefully laid out on a line of text or put on a different line of text. That is, if there is punctuation at the end of a run of text that should be carried over to a separate line does it actually get carried over. A true value will allow for hanging punctuation forcing the punctuation to not be carried over and a value of false will allow the punctuation to be carried onto the next text line. If this attribute is omitted, then a value of 0, or false is implied. The possible values for this attribute are defined by the XML Schema boolean datatype.
indent (Indent)	Specifies the indent size that will be applied to the first line of text in the paragraph. An indentation of 0 will be considered to be at the same location as marL attribute. If this attribute is omitted, then a value of -342900 is implied.

Attributes	Description

	Here is some Sample text
	
	[*Example*: Consider the scenario where the user now wanted to add a paragraph indentation to the first line of text in their two column format book.
	
	`<p:txBody>`
	`  <a:bodyPr numCol="2" spcCol="914400"…/>`
	`    <a:normAutofit/>`
	`    </a:bodyPr>`
	`  …`
	`  <a:p>`
	`    <a:pPr marL="0" indent="571500" algn="just">`
	`      <a:buNone/>`
	`    </a:pPr>`
	`    …`
	`    <a:t>Here is some…</a:t>`
	`    …`
	`  </a:p>`
	`</p:txBody>`
	
	By adding the indent attribute the user has effectively added a first line indent to this paragraph of text. *end example*]
	
	The possible values for this attribute are defined by the ST_TextIndent simple type (§5.1.12.70).

latinLnBrk (Latin Line Break)	Specifies whether a Latin word can be broken in half and wrapped onto the next line without a hyphen being added. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable Latin words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied. [*Example*: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in whole on a new line so that it

Attributes	Description

	may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.
	
	`<p:txBody>`
	`  …`
	`  <a:p>`
	`  <a:pPr latinLnBrk="0" …/>`
	`  …`
	`  <a:t>Sample text (Long word)</a:t>`
	`  …`
	`  </a:p>`
	`</p:txBody>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

lvl (Level)	Specifies the particular level text properties that this paragraph will follow. The value for this attribute is numerical and formats the text according to the corresponding level paragraph properties that are listed within the lstStyle element. Since there are nine separate level properties defined, this tag will have an effective range of 0-8 = 9 available values.
	
	[*Example*: Consider the following DrawingML. This would specify that this paragraph should follow the lvl2pPr formatting style because once again lvl="1" is considered to be level 2.
	
	`<p:txBody>`
	`  …`
	`  <a:p>`
	`  <a:pPr lvl="1" …/>`
	`  …`
	`  <a:t>Sample text</a:t>`
	`  …`
	`  </a:p>`
	`</p:txBody>`
	
	end example]
	
	[*Note*: To resolve conflicting paragraph properties the linear hierarchy of paragraph

Attributes	Description
	properties should be examined starting first with the pPr element. The rule here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. *end note*] The possible values for this attribute are defined by the ST_TextIndentLevelType simple type (§5.1.12.71).
marL (Left Margin)	Specifies the left margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marL attributes are additive with respect to the text position. If this attribute is omitted, then a value of 347663 is implied. The possible values for this attribute are defined by the ST_TextMargin simple type (§5.1.12.73).
marR (Right Margin)	Specifies the right margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marR attributes are additive with respect to the text position. If this attribute is omitted, then a value of 0 is implied. The possible values for this attribute are defined by the ST_TextMargin simple type (§5.1.12.73).

rtl (Right To Left)	Specifies whether the text is right-to-left or left-to-right in its flow direction. If this attribute is omitted, then a value of 0, or left-to-right is implied.
	
	[*Example*: Consider the scenario where the user wanted text to flow from right to left. That is within the bounding text box the first word would be right aligned with each additional word being written to the left of the previous while continuing to flow top to bottom. The DrawingML to describe this might be as follows.
	
	`<p:txBody>`
	`…`
	`<a:p>`
	`<a:pPr rtl="1" …/>`
	`…`
	`<a:t>Sample text…</a:t>`
	`…`
	`</a:p>`
	`</p:txBody>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextParagraphProperties">
   <sequence>
      <element name="lnSpc" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <element name="spcBef" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <element name="spcAft" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletColor" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletSize" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletTypeface" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBullet" minOccurs="0" maxOccurs="1"/>
      <element name="tabLst" type="CT_TextTabStopList" minOccurs="0" maxOccurs="1"/>
      <element name="defRPr" type="CT_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="marL" type="ST_TextMargin" use="optional"/>
   <attribute name="marR" type="ST_TextMargin" use="optional"/>
   <attribute name="lvl" type="ST_TextIndentLevelType" use="optional"/>
   <attribute name="indent" type="ST_TextIndent" use="optional"/>
   <attribute name="algn" type="ST_TextAlignType" use="optional"/>
   <attribute name="defTabSz" type="ST_Coordinate32" use="optional"/>
   <attribute name="rtl" type="xsd:boolean" use="optional"/>
   <attribute name="eaLnBrk" type="xsd:boolean" use="optional"/>
   <attribute name="fontAlgn" type="ST_TextFontAlignType" use="optional"/>
   <attribute name="latinLnBrk" type="xsd:boolean" use="optional"/>
   <attribute name="hangingPunct" type="xsd:boolean" use="optional"/>
</complexType>
```

5.1.5.4.16 lvl4pPr (List Level 4 Text Style)

This element specifies all paragraph level text properties for all elements that have the attribute lvl="3". There are a total of 9 level text property elements allowed, levels 0-8. It is recommended that the order in which this and other level property elements are specified be in order of increasing level. That is lvl2pPr should come before lvl3pPr. This allows the lower level properties to take precedence over the higher level ones because they are parsed first.

[*Example*: Consider the following DrawingML code that would specify a paragraph to follow the level style defined in lvl4pPr and thus create a paragraph of text that has no bullets and is right aligned.

```
<p:txBody>
  …
  <a:lstStyle>
    <a:lvl4pPr algn="r">
      <a:buNone/>
    </a:lvl4pPr>
  </a:lstStyle>
```

```
    <a:p>
      <a:pPr lvl="3">
      </a:pPr>
      …
      <a:t>Some text</a:t>
      …
    </a:p>
  </p:txBody>
```

end example]

[*Note*: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The rule here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. *end note*]

Parent Elements
bodyStyle (§4.4.1.5); defaultTextStyle (§4.3.1.7); lstStyle (§5.1.5.4.12); notesStyle (§4.4.1.25); otherStyle (§4.4.1.32); titleStyle (§4.4.1.45)

Child Elements	Subclause
buAutoNum (Auto-Numbered Bullet)	§5.1.5.4.1
buBlip (Picture Bullet)	§5.1.5.4.2
buChar (Character Bullet)	§5.1.5.4.3
buClr (Color Specified)	§5.1.5.4.4
buClrTx (Follow Text)	§5.1.5.4.5
buFont (Specified)	§5.1.5.4.6
buFontTx (Follow text)	§5.1.5.4.7
buNone (No Bullet)	§5.1.5.4.8
buSzPct (Bullet Size Percentage)	§5.1.5.4.9
buSzPts (Bullet Size Points)	§5.1.5.4.10
buSzTx (Bullet Size Follows Text)	§5.1.5.4.11
defRPr (Default Text Run Properties)	§5.1.5.3.2
extLst (Extension List)	§5.1.2.1.15
lnSpc (Line Spacing)	§5.1.5.2.5
spcAft (Space After)	§5.1.5.2.8
spcBef (Space Before)	§5.1.5.2.9
tabLst (Tab List)	§5.1.5.2.13

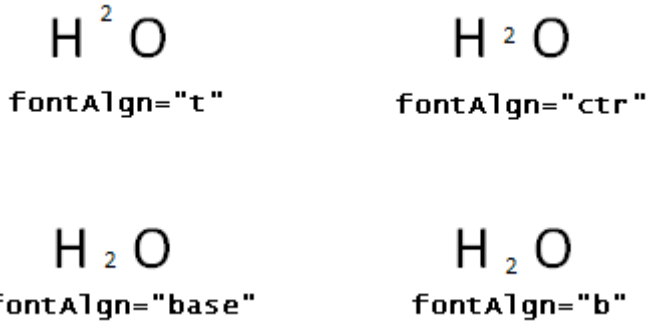
Attributes	Description
algn (Alignment)	Specifies the alignment that is to be applied to the paragraph. Possible values for this include left, right, centered, justified and distributed. If this attribute is omitted, then a value of left is implied. Sample text [*Example*: Consider the case where the user wishes to have two columns of text that have a justified alignment, much like text within a book. The following DrawingML could describe this. `<p:txBody>` `<a:bodyPr numCol="2" spcCol="914400"…/>` `<a:normAutofit/>` `</a:bodyPr>` `…` `<a:p>` `<a:pPr marL="0" algn="just">` `<a:buNone/>` `</a:pPr>` `…` `<a:t>Sample Text …</a:t>` `…` `</a:p>` `</p:txBody>` *end example*] The possible values for this attribute are defined by the ST_TextAlignType simple type (§5.1.12.59).
defTabSz (Default Tab Size)	Specifies the default size for a tab character within this paragraph. This attribute should be used to describe the spacing of tabs within the paragraph instead of a leading indentation tab. For indentation tabs there are the marL and indent attributes to assist with this.

Attributes	Description

	[*Example*: Consider the case where a paragraph contains numerous tabs that need to be of a specific size. The following DrawingML would describe this.
	
	`<p:txBody>`
	`  …`
	`  <a:p>`
	`  <a:pPr defTabSz="376300" …/>`
	`  …`
	`  <a:t>Sample Text …</a:t>`
	`  …`
	`  </a:p>`
	`</p:txBody>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_Coordinate32 simple type (§5.1.12.17).

eaLnBrk (East Asian Line Break)	Specifies whether an East Asian word can be broken in half and wrapped onto the next line without a hyphen being added. To determine whether an East Asian word can be broken the presentation application would use the kinsoku settings here. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable East Asian words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.
	
	[*Example*: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in whole on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.
	
	`<p:txBody>`
	`  …`
	`  <a:p>`
	`  <a:pPr eaLnBrk="0" …/>`
	`  …`
	`  <a:t>Sample text (Long word)</a:t>`
	`  …`
	`  </a:p>`
	`</p:txBody>`
	
	Sample text Sample text Sample text supercalifr
	agilisticexpialidocious

Attributes	Description
	Sample text Sample text Sample text supercalifragilisticexpialidocious *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.

fontAlgn (Font Alignment)	Determines where vertically on a line of text the actual words are positioned. This deals with vertical placement of the characters with respect to the baselines. For instance having text anchored to the top baseline, anchored to the bottom baseline, centered in between, etc. To understand this attribute and it's use it is helpful to understand what baselines are. A diagram describing these different cases is shown below. If this attribute is omitted, then a value of base is implied.
	
	[*Example*: Consider the case where the user wishes to represent the chemical compound of a water molecule. For this they will need to make sure the H, the 2, and the O are all in the correct position and are of the correct size. The results below can be achieved through the DrawingML shown below.
	
	 
	
	`<a:txtBody>`
	`…`
	`<a:pPr fontAlgn="b" …/>`
	`…`
	`<a:r>`
	`<a:rPr …/>`
	`<a:t>H </a:t>`
	`</a:r>`
	`<a:r>`
	`<a:rPr sz="1200" …/>`
	`<a:t>2</a:t>`
	`</a:r>`
	`<a:r>`
	`<a:rPr …/>`
	`<a:t>O</a:t>`
	`</a:r>`

Attributes	Description
	… </p:txBody> *end example*] The possible values for this attribute are defined by the ST_TextFontAlignType simple type (§5.1.12.66).
hangingPunct (Hanging Punctuation)	Specifies whether punctuation is to be forcefully laid out on a line of text or put on a different line of text. That is, if there is punctuation at the end of a run of text that should be carried over to a separate line does it actually get carried over. A true value will allow for hanging punctuation forcing the punctuation to not be carried over and a value of false will allow the punctuation to be carried onto the next text line. If this attribute is omitted, then a value of 0, or false is implied. The possible values for this attribute are defined by the XML Schema boolean datatype.

indent (Indent)	Specifies the indent size that will be applied to the first line of text in the paragraph. An indentation of 0 will be considered to be at the same location as marL attribute. If this attribute is omitted, then a value of -342900 is implied.
	
	Here is some Sample text
	
	[*Example*: Consider the scenario where the user now wanted to add a paragraph indentation to the first line of text in their two column format book.
	
	<p:txBody>
	  <a:bodyPr numCol="2" spcCol="914400"…/>
	    <a:normAutofit/>
	    </a:bodyPr>
	  …
	  <a:p>
	    <a:pPr marL="0" indent="571500" algn="just">
	      <a:buNone/>
	    </a:pPr>
	    …

Attributes	Description
	<a:t>Here is some…</a:t> … </a:p> </p:txBody> By adding the indent attribute the user has effectively added a first line indent to this paragraph of text. *end example*] The possible values for this attribute are defined by the ST_TextIndent simple type (§5.1.12.70).
latinLnBrk (Latin Line Break)	Specifies whether a Latin word can be broken in half and wrapped onto the next line without a hyphen being added. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable Latin words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied. [*Example*: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in whole on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows. <p:txBody> … <a:p> <a:pPr latinLnBrk="0" …/> … <a:t>Sample text (Long word)</a:t> … </a:p> </p:txBody> Sample text Sample text Sample text supercalifr agilisticexpialidocious Sample text Sample text Sample text supercalifragilisticexpialidocious *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
lvl (Level)	Specifies the particular level text properties that this paragraph will follow. The value for this attribute is numerical and formats the text according to the corresponding level paragraph properties that are listed within the lstStyle element. Since there are nine

Attributes	Description

	separate level properties defined, this tag will have an effective range of 0-8 = 9 available values.
	
	[*Example*: Consider the following DrawingML. This would specify that this paragraph should follow the lvl2pPr formatting style because once again lvl="1" is considered to be level 2.
	
	`<p:txBody>`
	`…`
	`<a:p>`
	`<a:pPr lvl="1" …/>`
	`…`
	`<a:t>Sample text</a:t>`
	`…`
	`</a:p>`
	`</p:txBody>`
	
	end example]
	
	[*Note*: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The rule here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. *end note*]
	
	The possible values for this attribute are defined by the ST_TextIndentLevelType simple type (§5.1.12.71).

marL (Left Margin)	Specifies the left margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marL attributes are additive with respect to the text position. If this attribute is omitted, then a value of 347663 is implied. The possible values for this attribute are defined by the ST_TextMargin simple type (§5.1.12.73).
marR (Right Margin)	Specifies the right margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marR attributes are additive with respect to the text position. If this attribute is omitted, then a value of 0 is implied. The possible values for this attribute are defined by the ST_TextMargin simple type (§5.1.12.73).
rtl (Right To Left)	Specifies whether the text is right-to-left or left-to-right in its flow direction. If this attribute is omitted, then a value of 0, or left-to-right is implied.

Attributes	Description

	[*Example*: Consider the scenario where the user wanted text to flow from right to left. That is within the bounding text box the first word would be right aligned with each additional word being written to the left of the previous while continuing to flow top to bottom. The DrawingML to describe this might be as follows.
	
	`<p:txBody>`
	`…`
	`<a:p>`
	`<a:pPr rtl="1" …/>`
	`…`
	`<a:t>Sample text…</a:t>`
	`…`
	`</a:p>`
	`</p:txBody>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextParagraphProperties">
   <sequence>
      <element name="lnSpc" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <element name="spcBef" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <element name="spcAft" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletColor" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletSize" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletTypeface" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBullet" minOccurs="0" maxOccurs="1"/>
      <element name="tabLst" type="CT_TextTabStopList" minOccurs="0" maxOccurs="1"/>
      <element name="defRPr" type="CT_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="marL" type="ST_TextMargin" use="optional"/>
   <attribute name="marR" type="ST_TextMargin" use="optional"/>
   <attribute name="lvl" type="ST_TextIndentLevelType" use="optional"/>
   <attribute name="indent" type="ST_TextIndent" use="optional"/>
   <attribute name="algn" type="ST_TextAlignType" use="optional"/>
   <attribute name="defTabSz" type="ST_Coordinate32" use="optional"/>
   <attribute name="rtl" type="xsd:boolean" use="optional"/>
   <attribute name="eaLnBrk" type="xsd:boolean" use="optional"/>
   <attribute name="fontAlgn" type="ST_TextFontAlignType" use="optional"/>
   <attribute name="latinLnBrk" type="xsd:boolean" use="optional"/>
   <attribute name="hangingPunct" type="xsd:boolean" use="optional"/>
</complexType>
```

5.1.5.4.17 lvl5pPr (List Level 5 Text Style)

This element specifies all paragraph level text properties for all elements that have the attribute lvl="4". There are a total of 9 level text property elements allowed, levels 0-8. It is recommended that the order in which this and other level property elements are specified be in order of increasing level. That is lvl2pPr should come before lvl3pPr. This allows the lower level properties to take precedence over the higher level ones because they are parsed first.

[*Example*: Consider the following DrawingML code that would specify a paragraph to follow the level style defined in lvl5pPr and thus create a paragraph of text that has no bullets and is right aligned.

```
<p:txBody>
  …
  <a:lstStyle>
    <a:lvl5pPr algn="r">
      <a:buNone/>
    </a:lvl5pPr>
  </a:lstStyle>
  <a:p>
    <a:pPr lvl="4">
    </a:pPr>
    …
    <a:t>Some text</a:t>
    …
  </a:p>
</p:txBody>
```

end example]

[*Note*: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The rule here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. *end note*]

Parent Elements
bodyStyle (§4.4.1.5); defaultTextStyle (§4.3.1.7); lstStyle (§5.1.5.4.12); notesStyle (§4.4.1.25); otherStyle (§4.4.1.32); titleStyle (§4.4.1.45)

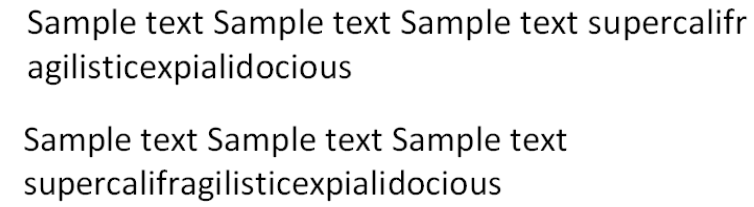
Child Elements	Subclause
buAutoNum (Auto-Numbered Bullet)	§5.1.5.4.1
buBlip (Picture Bullet)	§5.1.5.4.2
buChar (Character Bullet)	§5.1.5.4.3

Child Elements	Subclause
buClr (Color Specified)	§5.1.5.4.4
buClrTx (Follow Text)	§5.1.5.4.5
buFont (Specified)	§5.1.5.4.6
buFontTx (Follow text)	§5.1.5.4.7
buNone (No Bullet)	§5.1.5.4.8
buSzPct (Bullet Size Percentage)	§5.1.5.4.9
buSzPts (Bullet Size Points)	§5.1.5.4.10
buSzTx (Bullet Size Follows Text)	§5.1.5.4.11
defRPr (Default Text Run Properties)	§5.1.5.3.2
extLst (Extension List)	§5.1.2.1.15
lnSpc (Line Spacing)	§5.1.5.2.5
spcAft (Space After)	§5.1.5.2.8
spcBef (Space Before)	§5.1.5.2.9
tabLst (Tab List)	§5.1.5.2.13

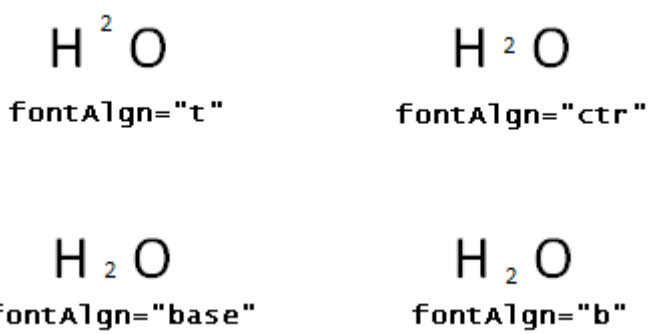
Attributes	Description
algn (Alignment)	Specifies the alignment that is to be applied to the paragraph. Possible values for this include left, right, centered, justified and distributed. If this attribute is omitted, then a value of left is implied. Sample text [*Example*: Consider the case where the user wishes to have two columns of text that have a justified alignment, much like text within a book. The following DrawingML could describe this. `<p:txBody>` `<a:bodyPr numCol="2" spcCol="914400"…/>`

Attributes	Description
	<a:normAutofit/> </a:bodyPr> … <a:p> <a:pPr marL="0" algn="just"> <a:buNone/> </a:pPr> … <a:t>Sample Text …</a:t> … </a:p> </p:txBody> *end example*] The possible values for this attribute are defined by the ST_TextAlignType simple type (§5.1.12.59).
defTabSz (Default Tab Size)	Specifies the default size for a tab character within this paragraph. This attribute should be used to describe the spacing of tabs within the paragraph instead of a leading indentation tab. For indentation tabs there are the marL and indent attributes to assist with this. [*Example*: Consider the case where a paragraph contains numerous tabs that need to be of a specific size. The following DrawingML would describe this. <p:txBody> … <a:p> <a:pPr defTabSz="376300" …/> … <a:t>Sample Text …</a:t> … </a:p> </p:txBody> *end example*] The possible values for this attribute are defined by the ST_Coordinate32 simple type (§5.1.12.17).
eaLnBrk (East Asian Line Break)	Specifies whether an East Asian word can be broken in half and wrapped onto the next line without a hyphen being added. To determine whether an East Asian word can be broken the presentation application would use the kinsoku settings here. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable East Asian words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.

Attributes	Description

	[*Example*: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in whole on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.
	
	`<p:txBody>`
	`  …`
	`  <a:p>`
	`  <a:pPr eaLnBrk="0" …/>`
	`  …`
	`  <a:t>Sample text (Long word)</a:t>`
	`  …`
	`  </a:p>`
	`</p:txBody>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

fontAlgn (Font Alignment)	Determines where vertically on a line of text the actual words are positioned. This deals with vertical placement of the characters with respect to the baselines. For instance having text anchored to the top baseline, anchored to the bottom baseline, centered in between, etc. To understand this attribute and it's use it is helpful to understand what baselines are. A diagram describing these different cases is shown below. If this attribute is omitted, then a value of base is implied. [*Example*: Consider the case where the user wishes to represent the chemical compound of a water molecule. For this they will need to make sure the H, the 2, and the O are all in the correct position and are of the correct size. The results below can be achieved through the DrawingML shown below.

Attributes	Description
	 ``` <a:txtBody> … <a:pPr fontAlgn="b" …/> … <a:r> <a:rPr …/> <a:t>H </a:t> </a:r> <a:r> <a:rPr sz="1200" …/> <a:t>2</a:t> </a:r> <a:r> <a:rPr …/> <a:t>O</a:t> </a:r> … </p:txBody> ``` *end example*] The possible values for this attribute are defined by the ST_TextFontAlignType simple type (§5.1.12.66).
hangingPunct (Hanging Punctuation)	Specifies whether punctuation is to be forcefully laid out on a line of text or put on a different line of text. That is, if there is punctuation at the end of a run of text that should be carried over to a separate line does it actually get carried over. A true value will allow for hanging punctuation forcing the punctuation to not be carried over and a value of false will allow the punctuation to be carried onto the next text line. If this attribute is omitted, then a value of 0, or false is implied. The possible values for this attribute are defined by the XML Schema boolean datatype.
indent (Indent)	Specifies the indent size that will be applied to the first line of text in the paragraph. An indentation of 0 will be considered to be at the same location as marL attribute. If this attribute is omitted, then a value of -342900 is implied.

Attributes	Description
	Here is some Sample text [*Example*: Consider the scenario where the user now wanted to add a paragraph indentation to the first line of text in their two column format book. `<p:txBody>` `  <a:bodyPr numCol="2" spcCol="914400"…/>` `    <a:normAutofit/>` `    </a:bodyPr>` `  …` `  <a:p>` `    <a:pPr marL="0" indent="571500" algn="just">` `      <a:buNone/>` `    </a:pPr>` `    …` `    <a:t>Here is some…</a:t>` `    …` `  </a:p>` `</p:txBody>` By adding the indent attribute the user has effectively added a first line indent to this paragraph of text. *end example*] The possible values for this attribute are defined by the ST_TextIndent simple type (§5.1.12.70).
latinLnBrk (Latin Line Break)	Specifies whether a Latin word can be broken in half and wrapped onto the next line without a hyphen being added. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable Latin words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied. [*Example*: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in whole on a new line so that it

Attributes	Description

	may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.
	
	`<p:txBody>`
	`  …`
	`  <a:p>`
	`  <a:pPr latinLnBrk="0" …/>`
	`  …`
	`  <a:t>Sample text (Long word)</a:t>`
	`  …`
	`  </a:p>`
	`</p:txBody>`
	
	Sample text Sample text Sample text supercalifr agilisticexpialidocious
	
	Sample text Sample text Sample text supercalifragilisticexpialidocious
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

lvl (Level)	Specifies the particular level text properties that this paragraph will follow. The value for this attribute is numerical and formats the text according to the corresponding level paragraph properties that are listed within the lstStyle element. Since there are nine separate level properties defined, this tag will have an effective range of 0-8 = 9 available values.
	
	[*Example*: Consider the following DrawingML. This would specify that this paragraph should follow the lvl2pPr formatting style because once again lvl="1" is considered to be level 2.
	
	`<p:txBody>`
	`  …`
	`  <a:p>`
	`  <a:pPr lvl="1" …/>`
	`  …`
	`  <a:t>Sample text</a:t>`
	`  …`
	`  </a:p>`
	`</p:txBody>`
	
	end example]
	
	[*Note*: To resolve conflicting paragraph properties the linear hierarchy of paragraph

Attributes	Description
	properties should be examined starting first with the pPr element. The rule here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. *end note*] The possible values for this attribute are defined by the ST_TextIndentLevelType simple type (§5.1.12.71).
marL (Left Margin)	Specifies the left margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marL attributes are additive with respect to the text position. If this attribute is omitted, then a value of 347663 is implied. The possible values for this attribute are defined by the ST_TextMargin simple type (§5.1.12.73).
marR (Right Margin)	Specifies the right margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marR attributes are additive with respect to the text position. If this attribute is omitted, then a value of 0 is implied. The possible values for this attribute are defined by the ST_TextMargin simple type (§5.1.12.73).

rtl (Right To Left)	Specifies whether the text is right-to-left or left-to-right in its flow direction. If this attribute is omitted, then a value of 0, or left-to-right is implied.
	
	[*Example*: Consider the scenario where the user wanted text to flow from right to left. That is within the bounding text box the first word would be right aligned with each additional word being written to the left of the previous while continuing to flow top to bottom. The DrawingML to describe this might be as follows.
	
	`<p:txBody>`
	`…`
	`<a:p>`
	`<a:pPr rtl="1" …/>`
	`…`
	`<a:t>Sample text…</a:t>`
	`…`
	`</a:p>`
	`</p:txBody>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextParagraphProperties">
   <sequence>
      <element name="lnSpc" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <element name="spcBef" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <element name="spcAft" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletColor" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletSize" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletTypeface" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBullet" minOccurs="0" maxOccurs="1"/>
      <element name="tabLst" type="CT_TextTabStopList" minOccurs="0" maxOccurs="1"/>
      <element name="defRPr" type="CT_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="marL" type="ST_TextMargin" use="optional"/>
   <attribute name="marR" type="ST_TextMargin" use="optional"/>
   <attribute name="lvl" type="ST_TextIndentLevelType" use="optional"/>
   <attribute name="indent" type="ST_TextIndent" use="optional"/>
   <attribute name="algn" type="ST_TextAlignType" use="optional"/>
   <attribute name="defTabSz" type="ST_Coordinate32" use="optional"/>
   <attribute name="rtl" type="xsd:boolean" use="optional"/>
   <attribute name="eaLnBrk" type="xsd:boolean" use="optional"/>
   <attribute name="fontAlgn" type="ST_TextFontAlignType" use="optional"/>
   <attribute name="latinLnBrk" type="xsd:boolean" use="optional"/>
   <attribute name="hangingPunct" type="xsd:boolean" use="optional"/>
</complexType>
```

5.1.5.4.18 lvl6pPr (List Level 6 Text Style)

This element specifies all paragraph level text properties for all elements that have the attribute lvl="5". There are a total of 9 level text property elements allowed, levels 0-8. It is recommended that the order in which this and other level property elements are specified be in order of increasing level. That is lvl2pPr should come before lvl3pPr. This allows the lower level properties to take precedence over the higher level ones because they are parsed first.

[*Example*: Consider the following DrawingML code that would specify a paragraph to follow the level style defined in lvl6pPr and thus create a paragraph of text that has no bullets and is right aligned.

```
<p:txBody>
  …
  <a:lstStyle>
    <a:lvl6pPr algn="r">
      <a:buNone/>
    </a:lvl6pPr>
  </a:lstStyle>
```

```
    <a:p>
      <a:pPr lvl="5">
      </a:pPr>
      …
      <a:t>Some text</a:t>
      …
    </a:p>
  </p:txBody>
```

end example]

[*Note*: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The rule here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. *end note*]

Parent Elements
bodyStyle (§4.4.1.5); defaultTextStyle (§4.3.1.7); lstStyle (§5.1.5.4.12); notesStyle (§4.4.1.25); otherStyle (§4.4.1.32); titleStyle (§4.4.1.45)

Child Elements	Subclause
buAutoNum (Auto-Numbered Bullet)	§5.1.5.4.1
buBlip (Picture Bullet)	§5.1.5.4.2
buChar (Character Bullet)	§5.1.5.4.3
buClr (Color Specified)	§5.1.5.4.4
buClrTx (Follow Text)	§5.1.5.4.5
buFont (Specified)	§5.1.5.4.6
buFontTx (Follow text)	§5.1.5.4.7
buNone (No Bullet)	§5.1.5.4.8
buSzPct (Bullet Size Percentage)	§5.1.5.4.9
buSzPts (Bullet Size Points)	§5.1.5.4.10
buSzTx (Bullet Size Follows Text)	§5.1.5.4.11
defRPr (Default Text Run Properties)	§5.1.5.3.2
extLst (Extension List)	§5.1.2.1.15
lnSpc (Line Spacing)	§5.1.5.2.5
spcAft (Space After)	§5.1.5.2.8
spcBef (Space Before)	§5.1.5.2.9
tabLst (Tab List)	§5.1.5.2.13

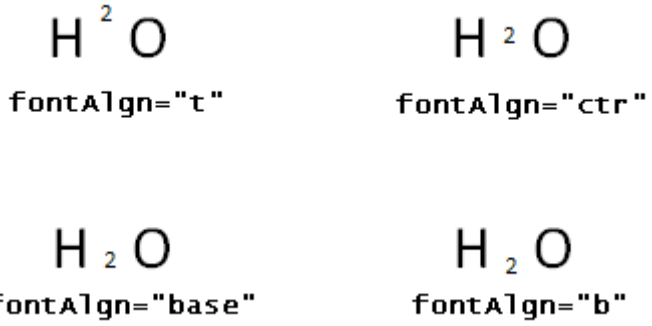
Attributes	Description
algn (Alignment)	Specifies the alignment that is to be applied to the paragraph. Possible values for this include left, right, centered, justified and distributed. If this attribute is omitted, then a value of left is implied. Sample text [*Example*: Consider the case where the user wishes to have two columns of text that have a justified alignment, much like text within a book. The following DrawingML could describe this. `<p:txBody>` `<a:bodyPr numCol="2" spcCol="914400"…/>` `<a:normAutofit/>` `</a:bodyPr>` … `<a:p>` `<a:pPr marL="0" algn="just">` `<a:buNone/>` `</a:pPr>` … `<a:t>Sample Text …</a:t>` … `</a:p>` `</p:txBody>` *end example*] The possible values for this attribute are defined by the ST_TextAlignType simple type (§5.1.12.59).
defTabSz (Default Tab Size)	Specifies the default size for a tab character within this paragraph. This attribute should be used to describe the spacing of tabs within the paragraph instead of a leading indentation tab. For indentation tabs there are the marL and indent attributes to assist with this.

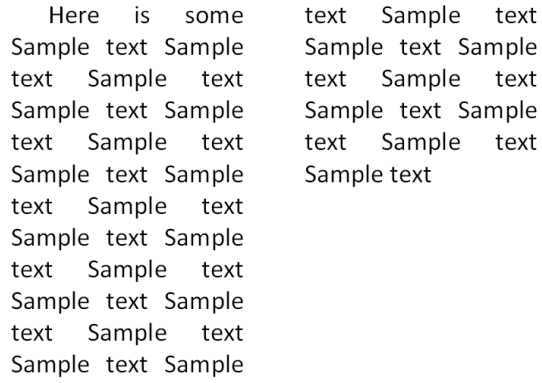
Attributes	Description

	[*Example*: Consider the case where a paragraph contains numerous tabs that need to be of a specific size. The following DrawingML would describe this.
	
	`<p:txBody>`
	`  …`
	`  <a:p>`
	`  <a:pPr defTabSz="376300" …/>`
	`  …`
	`  <a:t>Sample Text …</a:t>`
	`  …`
	`  </a:p>`
	`</p:txBody>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_Coordinate32 simple type (§5.1.12.17).

eaLnBrk (East Asian Line Break)	Specifies whether an East Asian word can be broken in half and wrapped onto the next line without a hyphen being added. To determine whether an East Asian word can be broken the presentation application would use the kinsoku settings here. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable East Asian words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.
	
	[*Example*: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in whole on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.
	
	`<p:txBody>`
	`  …`
	`  <a:p>`
	`  <a:pPr eaLnBrk="0" …/>`
	`  …`
	`  <a:t>Sample text (Long word)</a:t>`
	`  …`
	`  </a:p>`
	`</p:txBody>`
	
	Sample text Sample text Sample text supercalifr
	agilisticexpialidocious

Attributes	Description
	Sample text Sample text Sample text supercalifragilisticexpialidocious *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.

fontAlgn (Font Alignment)	Determines where vertically on a line of text the actual words are positioned. This deals with vertical placement of the characters with respect to the baselines. For instance having text anchored to the top baseline, anchored to the bottom baseline, centered in between, etc. To understand this attribute and it's use it is helpful to understand what baselines are. A diagram describing these different cases is shown below. If this attribute is omitted, then a value of base is implied.
	
	[*Example*: Consider the case where the user wishes to represent the chemical compound of a water molecule. For this they will need to make sure the H, the 2, and the O are all in the correct position and are of the correct size. The results below can be achieved through the DrawingML shown below.
	
	 
	fontAlgn="t" fontAlgn="ctr" fontAlgn="base" fontAlgn="b"
	
	`<a:txtBody>`
	`…`
	`<a:pPr fontAlgn="b" …/>`
	`…`
	`<a:r>`
	`<a:rPr …/>`
	`<a:t>H </a:t>`
	`</a:r>`
	`<a:r>`
	`<a:rPr sz="1200" …/>`
	`<a:t>2</a:t>`
	`</a:r>`
	`<a:r>`
	`<a:rPr …/>`
	`<a:t>O</a:t>`
	`</a:r>`

Attributes	Description
	… </p:txBody> *end example*] The possible values for this attribute are defined by the ST_TextFontAlignType simple type (§5.1.12.66).
hangingPunct (Hanging Punctuation)	Specifies whether punctuation is to be forcefully laid out on a line of text or put on a different line of text. That is, if there is punctuation at the end of a run of text that should be carried over to a separate line does it actually get carried over. A true value will allow for hanging punctuation forcing the punctuation to not be carried over and a value of false will allow the punctuation to be carried onto the next text line. If this attribute is omitted, then a value of 0, or false is implied. The possible values for this attribute are defined by the XML Schema boolean datatype.
indent (Indent)	Specifies the indent size that will be applied to the first line of text in the paragraph. An indentation of 0 will be considered to be at the same location as marL attribute. If this attribute is omitted, then a value of -342900 is implied.  [*Example*: Consider the scenario where the user now wanted to add a paragraph indentation to the first line of text in their two column format book. <p:txBody> <a:bodyPr numCol="2" spcCol="914400"…/> <a:normAutofit/> </a:bodyPr> … <a:p> <a:pPr marL="0" indent="571500" algn="just"> <a:buNone/> </a:pPr> …

Attributes	Description
	<a:t>Here is some…</a:t> … </a:p> </p:txBody> By adding the indent attribute the user has effectively added a first line indent to this paragraph of text. *end example*] The possible values for this attribute are defined by the ST_TextIndent simple type (§5.1.12.70).
latinLnBrk (Latin Line Break)	Specifies whether a Latin word can be broken in half and wrapped onto the next line without a hyphen being added. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable Latin words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied. [*Example*: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in whole on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows. <p:txBody> … <a:p> <a:pPr latinLnBrk="0" …/> … <a:t>Sample text (Long word)</a:t> … </a:p> </p:txBody> Sample text Sample text Sample text supercalifr agilisticexpialidocious Sample text Sample text Sample text supercalifragilisticexpialidocious *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
lvl (Level)	Specifies the particular level text properties that this paragraph will follow. The value for this attribute is numerical and formats the text according to the corresponding level paragraph properties that are listed within the lstStyle element. Since there are nine

Attributes	Description

	separate level properties defined, this tag will have an effective range of 0-8 = 9 available values.
	
	[*Example*: Consider the following DrawingML. This would specify that this paragraph should follow the lvl2pPr formatting style because once again lvl="1" is considered to be level 2.
	
	`<p:txBody>`
	`…`
	`<a:p>`
	`<a:pPr lvl="1" …/>`
	`…`
	`<a:t>Sample text</a:t>`
	`…`
	`</a:p>`
	`</p:txBody>`
	
	end example]
	
	[*Note*: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The rule here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. *end note*]
	
	The possible values for this attribute are defined by the ST_TextIndentLevelType simple type (§5.1.12.71).

marL (Left Margin)	Specifies the left margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marL attributes are additive with respect to the text position. If this attribute is omitted, then a value of 347663 is implied. The possible values for this attribute are defined by the ST_TextMargin simple type (§5.1.12.73).
marR (Right Margin)	Specifies the right margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marR attributes are additive with respect to the text position. If this attribute is omitted, then a value of 0 is implied. The possible values for this attribute are defined by the ST_TextMargin simple type (§5.1.12.73).
rtl (Right To Left)	Specifies whether the text is right-to-left or left-to-right in its flow direction. If this attribute is omitted, then a value of 0, or left-to-right is implied.

Attributes	Description

	[*Example*: Consider the scenario where the user wanted text to flow from right to left. That is within the bounding text box the first word would be right aligned with each additional word being written to the left of the previous while continuing to flow top to bottom. The DrawingML to describe this might be as follows.
	
	`<p:txBody>`
	`…`
	`<a:p>`
	`<a:pPr rtl="1" …/>`
	`…`
	`<a:t>Sample text…</a:t>`
	`…`
	`</a:p>`
	`</p:txBody>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextParagraphProperties">
   <sequence>
      <element name="lnSpc" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <element name="spcBef" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <element name="spcAft" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletColor" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletSize" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletTypeface" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBullet" minOccurs="0" maxOccurs="1"/>
      <element name="tabLst" type="CT_TextTabStopList" minOccurs="0" maxOccurs="1"/>
      <element name="defRPr" type="CT_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="marL" type="ST_TextMargin" use="optional"/>
   <attribute name="marR" type="ST_TextMargin" use="optional"/>
   <attribute name="lvl" type="ST_TextIndentLevelType" use="optional"/>
   <attribute name="indent" type="ST_TextIndent" use="optional"/>
   <attribute name="algn" type="ST_TextAlignType" use="optional"/>
   <attribute name="defTabSz" type="ST_Coordinate32" use="optional"/>
   <attribute name="rtl" type="xsd:boolean" use="optional"/>
   <attribute name="eaLnBrk" type="xsd:boolean" use="optional"/>
   <attribute name="fontAlgn" type="ST_TextFontAlignType" use="optional"/>
   <attribute name="latinLnBrk" type="xsd:boolean" use="optional"/>
   <attribute name="hangingPunct" type="xsd:boolean" use="optional"/>
</complexType>
```

5.1.5.4.19 lvl7pPr (List Level 7 Text Style)

This element specifies all paragraph level text properties for all elements that have the attribute lvl="6". There are a total of 9 level text property elements allowed, levels 0-8. It is recommended that the order in which this and other level property elements are specified be in order of increasing level. That is lvl2pPr should come before lvl3pPr. This allows the lower level properties to take precedence over the higher level ones because they are parsed first.

[*Example*: Consider the following DrawingML code that would specify a paragraph to follow the level style defined in lvl7pPr and thus create a paragraph of text that has no bullets and is right aligned.

```
<p:txBody>
  …
  <a:lstStyle>
    <a:lvl7pPr algn="r">
      <a:buNone/>
    </a:lvl7pPr>
  </a:lstStyle>
  <a:p>
    <a:pPr lvl="6">
    </a:pPr>
    …
    <a:t>Some text</a:t>
    …
  </a:p>
</p:txBody>
```

end example]

[*Note*: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The rule here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. *end note*]

Parent Elements
bodyStyle (§4.4.1.5); defaultTextStyle (§4.3.1.7); lstStyle (§5.1.5.4.12); notesStyle (§4.4.1.25); otherStyle (§4.4.1.32); titleStyle (§4.4.1.45)

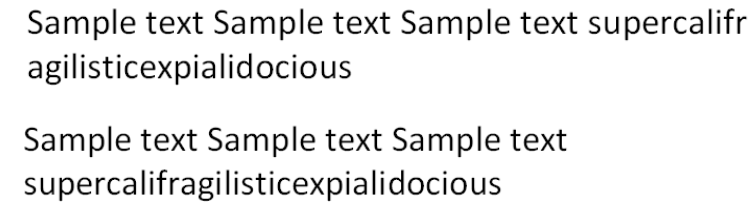
Child Elements	Subclause
buAutoNum (Auto-Numbered Bullet)	§5.1.5.4.1
buBlip (Picture Bullet)	§5.1.5.4.2
buChar (Character Bullet)	§5.1.5.4.3

Child Elements	Subclause
buClr (Color Specified)	§5.1.5.4.4
buClrTx (Follow Text)	§5.1.5.4.5
buFont (Specified)	§5.1.5.4.6
buFontTx (Follow text)	§5.1.5.4.7
buNone (No Bullet)	§5.1.5.4.8
buSzPct (Bullet Size Percentage)	§5.1.5.4.9
buSzPts (Bullet Size Points)	§5.1.5.4.10
buSzTx (Bullet Size Follows Text)	§5.1.5.4.11
defRPr (Default Text Run Properties)	§5.1.5.3.2
extLst (Extension List)	§5.1.2.1.15
lnSpc (Line Spacing)	§5.1.5.2.5
spcAft (Space After)	§5.1.5.2.8
spcBef (Space Before)	§5.1.5.2.9
tabLst (Tab List)	§5.1.5.2.13

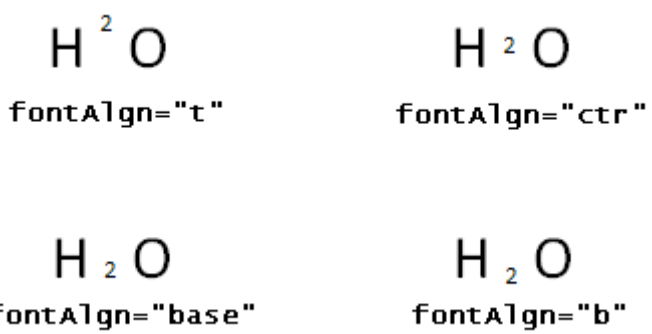
Attributes	Description
algn (Alignment)	Specifies the alignment that is to be applied to the paragraph. Possible values for this include left, right, centered, justified and distributed. If this attribute is omitted, then a value of left is implied. Sample text [*Example*: Consider the case where the user wishes to have two columns of text that have a justified alignment, much like text within a book. The following DrawingML could describe this. `<p:txBody>` `  <a:bodyPr numCol="2" spcCol="914400"…/>`

Attributes	Description
	<a:normAutofit/> </a:bodyPr> … <a:p> <a:pPr marL="0" algn="just"> <a:buNone/> </a:pPr> … <a:t>Sample Text …</a:t> … </a:p> </p:txBody> *end example*] The possible values for this attribute are defined by the ST_TextAlignType simple type (§5.1.12.59).
defTabSz (Default Tab Size)	Specifies the default size for a tab character within this paragraph. This attribute should be used to describe the spacing of tabs within the paragraph instead of a leading indentation tab. For indentation tabs there are the marL and indent attributes to assist with this. [*Example*: Consider the case where a paragraph contains numerous tabs that need to be of a specific size. The following DrawingML would describe this. <p:txBody> … <a:p> <a:pPr defTabSz="376300" …/> … <a:t>Sample Text …</a:t> … </a:p> </p:txBody> *end example*] The possible values for this attribute are defined by the ST_Coordinate32 simple type (§5.1.12.17).
eaLnBrk (East Asian Line Break)	Specifies whether an East Asian word can be broken in half and wrapped onto the next line without a hyphen being added. To determine whether an East Asian word can be broken the presentation application would use the kinsoku settings here. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable East Asian words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.

Attributes	Description

	[*Example*: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in whole on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.
	
	`<p:txBody>`
	`  …`
	`  <a:p>`
	`  <a:pPr eaLnBrk="0" …/>`
	`  …`
	`  <a:t>Sample text (Long word)</a:t>`
	`  …`
	`  </a:p>`
	`</p:txBody>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

fontAlgn (Font Alignment)	Determines where vertically on a line of text the actual words are positioned. This deals with vertical placement of the characters with respect to the baselines. For instance having text anchored to the top baseline, anchored to the bottom baseline, centered in between, etc. To understand this attribute and it's use it is helpful to understand what baselines are. A diagram describing these different cases is shown below. If this attribute is omitted, then a value of base is implied. [*Example*: Consider the case where the user wishes to represent the chemical compound of a water molecule. For this they will need to make sure the H, the 2, and the O are all in the correct position and are of the correct size. The results below can be achieved through the DrawingML shown below.

Attributes	Description
	 ``` <a:txtBody> … <a:pPr fontAlgn="b" …/> … <a:r> <a:rPr …/> <a:t>H </a:t> </a:r> <a:r> <a:rPr sz="1200" …/> <a:t>2</a:t> </a:r> <a:r> <a:rPr …/> <a:t>O</a:t> </a:r> … </p:txBody> ``` *end example*] The possible values for this attribute are defined by the ST_TextFontAlignType simple type (§5.1.12.66).
hangingPunct (Hanging Punctuation)	Specifies whether punctuation is to be forcefully laid out on a line of text or put on a different line of text. That is, if there is punctuation at the end of a run of text that should be carried over to a separate line does it actually get carried over. A true value will allow for hanging punctuation forcing the punctuation to not be carried over and a value of false will allow the punctuation to be carried onto the next text line. If this attribute is omitted, then a value of 0, or false is implied. The possible values for this attribute are defined by the XML Schema boolean datatype.
indent (Indent)	Specifies the indent size that will be applied to the first line of text in the paragraph. An indentation of 0 will be considered to be at the same location as marL attribute. If this attribute is omitted, then a value of -342900 is implied.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>

Here is some Sample text

[*Example*: Consider the scenario where the user now wanted to add a paragraph indentation to the first line of text in their two column format book.

```
<p:txBody>
  <a:bodyPr numCol="2" spcCol="914400"…/>
    <a:normAutofit/>
    </a:bodyPr>
  …
  <a:p>
    <a:pPr marL="0" indent="571500" algn="just">
      <a:buNone/>
    </a:pPr>
    …
    <a:t>Here is some…</a:t>
    …
  </a:p>
</p:txBody>
```

By adding the indent attribute the user has effectively added a first line indent to this paragraph of text. *end example*]

The possible values for this attribute are defined by the ST_TextIndent simple type (§5.1.12.70).

</td></tr>
<tr><td>latinLnBrk (Latin Line Break)</td><td>

Specifies whether a Latin word can be broken in half and wrapped onto the next line without a hyphen being added. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable Latin words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.

[*Example*: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in whole on a new line so that it

</td></tr>
</table>

Attributes	Description

	may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.
	
	`<p:txBody>`
	`  …`
	`  <a:p>`
	`  <a:pPr latinLnBrk="0" …/>`
	`  …`
	`  <a:t>Sample text (Long word)</a:t>`
	`  …`
	`  </a:p>`
	`</p:txBody>`
	
	Sample text Sample text Sample text supercalifr agilisticexpialidocious
	
	Sample text Sample text Sample text supercalifragilisticexpialidocious
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

lvl (Level)	Specifies the particular level text properties that this paragraph will follow. The value for this attribute is numerical and formats the text according to the corresponding level paragraph properties that are listed within the lstStyle element. Since there are nine separate level properties defined, this tag will have an effective range of 0-8 = 9 available values.
	
	[*Example*: Consider the following DrawingML. This would specify that this paragraph should follow the lvl2pPr formatting style because once again lvl="1" is considered to be level 2.
	
	`<p:txBody>`
	`  …`
	`  <a:p>`
	`  <a:pPr lvl="1" …/>`
	`  …`
	`  <a:t>Sample text</a:t>`
	`  …`
	`  </a:p>`
	`</p:txBody>`
	
	end example]
	
	[*Note*: To resolve conflicting paragraph properties the linear hierarchy of paragraph

Attributes	Description
	properties should be examined starting first with the pPr element. The rule here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. *end note*] The possible values for this attribute are defined by the ST_TextIndentLevelType simple type (§5.1.12.71).
marL (Left Margin)	Specifies the left margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marL attributes are additive with respect to the text position. If this attribute is omitted, then a value of 347663 is implied. The possible values for this attribute are defined by the ST_TextMargin simple type (§5.1.12.73).
marR (Right Margin)	Specifies the right margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marR attributes are additive with respect to the text position. If this attribute is omitted, then a value of 0 is implied. The possible values for this attribute are defined by the ST_TextMargin simple type (§5.1.12.73).

rtl (Right To Left)	Specifies whether the text is right-to-left or left-to-right in its flow direction. If this attribute is omitted, then a value of 0, or left-to-right is implied.
	
	[*Example*: Consider the scenario where the user wanted text to flow from right to left. That is within the bounding text box the first word would be right aligned with each additional word being written to the left of the previous while continuing to flow top to bottom. The DrawingML to describe this might be as follows.
	
	`<p:txBody>`
	`…`
	`<a:p>`
	`<a:pPr rtl="1" …/>`
	`…`
	`<a:t>Sample text…</a:t>`
	`…`
	`</a:p>`
	`</p:txBody>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextParagraphProperties">
   <sequence>
      <element name="lnSpc" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <element name="spcBef" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <element name="spcAft" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletColor" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletSize" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletTypeface" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBullet" minOccurs="0" maxOccurs="1"/>
      <element name="tabLst" type="CT_TextTabStopList" minOccurs="0" maxOccurs="1"/>
      <element name="defRPr" type="CT_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="marL" type="ST_TextMargin" use="optional"/>
   <attribute name="marR" type="ST_TextMargin" use="optional"/>
   <attribute name="lvl" type="ST_TextIndentLevelType" use="optional"/>
   <attribute name="indent" type="ST_TextIndent" use="optional"/>
   <attribute name="algn" type="ST_TextAlignType" use="optional"/>
   <attribute name="defTabSz" type="ST_Coordinate32" use="optional"/>
   <attribute name="rtl" type="xsd:boolean" use="optional"/>
   <attribute name="eaLnBrk" type="xsd:boolean" use="optional"/>
   <attribute name="fontAlgn" type="ST_TextFontAlignType" use="optional"/>
   <attribute name="latinLnBrk" type="xsd:boolean" use="optional"/>
   <attribute name="hangingPunct" type="xsd:boolean" use="optional"/>
</complexType>
```

5.1.5.4.20 lvl8pPr (List Level 8 Text Style)

This element specifies all paragraph level text properties for all elements that have the attribute lvl="7". There are a total of 9 level text property elements allowed, levels 0-8. It is recommended that the order in which this and other level property elements are specified be in order of increasing level. That is lvl2pPr should come before lvl3pPr. This allows the lower level properties to take precedence over the higher level ones because they are parsed first.

[*Example*: Consider the following DrawingML code that would specify a paragraph to follow the level style defined in lvl8pPr and thus create a paragraph of text that has no bullets and is right aligned.

```
<p:txBody>
  …
  <a:lstStyle>
    <a:lvl8pPr algn="r">
      <a:buNone/>
    </a:lvl8pPr>
  </a:lstStyle>
  <a:p>
    <a:pPr lvl="7">
    </a:pPr>
    …
```

```
      <a:t>Some text</a:t>
      …
    </a:p>
  </p:txBody>
```

end example]

[*Note*: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The rule here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. *end note*]

Parent Elements
bodyStyle (§4.4.1.5); defaultTextStyle (§4.3.1.7); lstStyle (§5.1.5.4.12); notesStyle (§4.4.1.25); otherStyle (§4.4.1.32); titleStyle (§4.4.1.45)

Child Elements	Subclause
buAutoNum (Auto-Numbered Bullet)	§5.1.5.4.1
buBlip (Picture Bullet)	§5.1.5.4.2
buChar (Character Bullet)	§5.1.5.4.3
buClr (Color Specified)	§5.1.5.4.4
buClrTx (Follow Text)	§5.1.5.4.5
buFont (Specified)	§5.1.5.4.6
buFontTx (Follow text)	§5.1.5.4.7
buNone (No Bullet)	§5.1.5.4.8
buSzPct (Bullet Size Percentage)	§5.1.5.4.9
buSzPts (Bullet Size Points)	§5.1.5.4.10
buSzTx (Bullet Size Follows Text)	§5.1.5.4.11
defRPr (Default Text Run Properties)	§5.1.5.3.2
extLst (Extension List)	§5.1.2.1.15
lnSpc (Line Spacing)	§5.1.5.2.5
spcAft (Space After)	§5.1.5.2.8
spcBef (Space Before)	§5.1.5.2.9
tabLst (Tab List)	§5.1.5.2.13

Attributes	Description

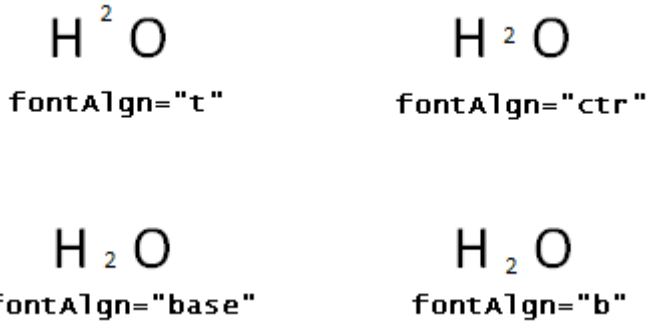
Attributes	Description
algn (Alignment)	Specifies the alignment that is to be applied to the paragraph. Possible values for this include left, right, centered, justified and distributed. If this attribute is omitted, then a value of left is implied. Sample text [*Example*: Consider the case where the user wishes to have two columns of text that have a justified alignment, much like text within a book. The following DrawingML could describe this. `<p:txBody>` `  <a:bodyPr numCol="2" spcCol="914400"…/>` `    <a:normAutofit/>` `    </a:bodyPr>` `  …` `  <a:p>` `    <a:pPr marL="0" algn="just">` `      <a:buNone/>` `    </a:pPr>` `    …` `    <a:t>Sample Text …</a:t>` `    …` `  </a:p>` `</p:txBody>` *end example*] The possible values for this attribute are defined by the ST_TextAlignType simple type (§5.1.12.59).
defTabSz (Default Tab Size)	Specifies the default size for a tab character within this paragraph. This attribute should be used to describe the spacing of tabs within the paragraph instead of a leading indentation tab. For indentation tabs there are the marL and indent attributes to assist with this. [*Example*: Consider the case where a paragraph contains numerous tabs that need to be

Attributes	Description

	of a specific size. The following DrawingML would describe this.
	
	`<p:txBody>`
	`…`
	`<a:p>`
	`<a:pPr defTabSz="376300" …/>`
	`…`
	`<a:t>Sample Text …</a:t>`
	`…`
	`</a:p>`
	`</p:txBody>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_Coordinate32 simple type (§5.1.12.17).

eaLnBrk (East Asian Line Break)	Specifies whether an East Asian word can be broken in half and wrapped onto the next line without a hyphen being added. To determine whether an East Asian word can be broken the presentation application would use the kinsoku settings here. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable East Asian words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.
	
	[*Example*: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in whole on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.
	
	`<p:txBody>`
	`…`
	`<a:p>`
	`<a:pPr eaLnBrk="0" …/>`
	`…`
	`<a:t>Sample text (Long word)</a:t>`
	`…`
	`</a:p>`
	`</p:txBody>`
	
	Sample text Sample text Sample text supercalifr
	agilisticexpialidocious

Attributes	Description
	Sample text Sample text Sample text supercalifragilisticexpialidocious *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.

fontAlgn (Font Alignment)	Determines where vertically on a line of text the actual words are positioned. This deals with vertical placement of the characters with respect to the baselines. For instance having text anchored to the top baseline, anchored to the bottom baseline, centered in between, etc. To understand this attribute and it's use it is helpful to understand what baselines are. A diagram describing these different cases is shown below. If this attribute is omitted, then a value of base is implied.
	
	[*Example*: Consider the case where the user wishes to represent the chemical compound of a water molecule. For this they will need to make sure the H, the 2, and the O are all in the correct position and are of the correct size. The results below can be achieved through the DrawingML shown below.
	
	 
	
	`<a:txtBody>`
	`…`
	`<a:pPr fontAlgn="b" …/>`
	`…`
	`<a:r>`
	`<a:rPr …/>`
	`<a:t>H </a:t>`
	`</a:r>`
	`<a:r>`
	`<a:rPr sz="1200" …/>`
	`<a:t>2</a:t>`
	`</a:r>`
	`<a:r>`
	`<a:rPr …/>`
	`<a:t>O</a:t>`
	`</a:r>`

Attributes	Description

	`…`
	`</p:txBody>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TextFontAlignType simple type (§5.1.12.66).

hangingPunct (Hanging Punctuation)	Specifies whether punctuation is to be forcefully laid out on a line of text or put on a different line of text. That is, if there is punctuation at the end of a run of text that should be carried over to a separate line does it actually get carried over. A true value will allow for hanging punctuation forcing the punctuation to not be carried over and a value of false will allow the punctuation to be carried onto the next text line. If this attribute is omitted, then a value of 0, or false is implied. The possible values for this attribute are defined by the XML Schema boolean datatype.
indent (Indent)	Specifies the indent size that will be applied to the first line of text in the paragraph. An indentation of 0 will be considered to be at the same location as marL attribute. If this attribute is omitted, then a value of -342900 is implied. Here is some Sample text [*Example*: Consider the scenario where the user now wanted to add a paragraph indentation to the first line of text in their two column format book. `<p:txBody>` `  <a:bodyPr numCol="2" spcCol="914400"…/>` `    <a:normAutofit/>` `    </a:bodyPr>` `  …` `  <a:p>` `    <a:pPr marL="0" indent="571500" algn="just">` `      <a:buNone/>` `    </a:pPr>` `    …`

Attributes	Description
	<a:t>Here is some…</a:t> … </a:p> </p:txBody> By adding the indent attribute the user has effectively added a first line indent to this paragraph of text. *end example*] The possible values for this attribute are defined by the ST_TextIndent simple type (§5.1.12.70).
latinLnBrk (Latin Line Break)	Specifies whether a Latin word can be broken in half and wrapped onto the next line without a hyphen being added. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable Latin words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied. [*Example*: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in whole on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows. <p:txBody> … <a:p> <a:pPr latinLnBrk="0" …/> … <a:t>Sample text (Long word)</a:t> … </a:p> </p:txBody> Sample text Sample text Sample text supercalifr agilisticexpialidocious Sample text Sample text Sample text supercalifragilisticexpialidocious *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
lvl (Level)	Specifies the particular level text properties that this paragraph will follow. The value for this attribute is numerical and formats the text according to the corresponding level paragraph properties that are listed within the lstStyle element. Since there are nine

Attributes	Description

	separate level properties defined, this tag will have an effective range of 0-8 = 9 available values.
	
	[*Example*: Consider the following DrawingML. This would specify that this paragraph should follow the lvl2pPr formatting style because once again lvl="1" is considered to be level 2.
	
	`<p:txBody>`
	`…`
	`<a:p>`
	`<a:pPr lvl="1" …/>`
	`…`
	`<a:t>Sample text</a:t>`
	`…`
	`</a:p>`
	`</p:txBody>`
	
	end example]
	
	[*Note*: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The rule here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. *end note*]
	
	The possible values for this attribute are defined by the ST_TextIndentLevelType simple type (§5.1.12.71).

marL (Left Margin)	Specifies the left margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marL attributes are additive with respect to the text position. If this attribute is omitted, then a value of 347663 is implied. The possible values for this attribute are defined by the ST_TextMargin simple type (§5.1.12.73).
marR (Right Margin)	Specifies the right margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marR attributes are additive with respect to the text position. If this attribute is omitted, then a value of 0 is implied. The possible values for this attribute are defined by the ST_TextMargin simple type (§5.1.12.73).
rtl (Right To Left)	Specifies whether the text is right-to-left or left-to-right in its flow direction. If this attribute is omitted, then a value of 0, or left-to-right is implied.

Attributes	Description

	[*Example*: Consider the scenario where the user wanted text to flow from right to left. That is within the bounding text box the first word would be right aligned with each additional word being written to the left of the previous while continuing to flow top to bottom. The DrawingML to describe this might be as follows.
	
	`<p:txBody>`
	`…`
	`<a:p>`
	`<a:pPr rtl="1" …/>`
	`…`
	`<a:t>Sample text…</a:t>`
	`…`
	`</a:p>`
	`</p:txBody>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextParagraphProperties">
   <sequence>
      <element name="lnSpc" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <element name="spcBef" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <element name="spcAft" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletColor" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletSize" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletTypeface" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBullet" minOccurs="0" maxOccurs="1"/>
      <element name="tabLst" type="CT_TextTabStopList" minOccurs="0" maxOccurs="1"/>
      <element name="defRPr" type="CT_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="marL" type="ST_TextMargin" use="optional"/>
   <attribute name="marR" type="ST_TextMargin" use="optional"/>
   <attribute name="lvl" type="ST_TextIndentLevelType" use="optional"/>
   <attribute name="indent" type="ST_TextIndent" use="optional"/>
   <attribute name="algn" type="ST_TextAlignType" use="optional"/>
   <attribute name="defTabSz" type="ST_Coordinate32" use="optional"/>
   <attribute name="rtl" type="xsd:boolean" use="optional"/>
   <attribute name="eaLnBrk" type="xsd:boolean" use="optional"/>
   <attribute name="fontAlgn" type="ST_TextFontAlignType" use="optional"/>
   <attribute name="latinLnBrk" type="xsd:boolean" use="optional"/>
   <attribute name="hangingPunct" type="xsd:boolean" use="optional"/>
</complexType>
```

5.1.5.4.21 lvl9pPr (List Level 9 Text Style)

This element specifies all paragraph level text properties for all elements that have the attribute lvl="8". There are a total of 9 level text property elements allowed, levels 0-8. It is recommended that the order in which this and other level property elements are specified be in order of increasing level. That is lvl2pPr should come before lvl3pPr. This allows the lower level properties to take precedence over the higher level ones because they are parsed first.

[*Example*: Consider the following DrawingML code that would specify a paragraph to follow the level style defined in lvl9pPr and thus create a paragraph of text that has no bullets and is right aligned.

```
<p:txBody>
  …
  <a:lstStyle>
    <a:lvl9pPr algn="r">
      <a:buNone/>
    </a:lvl9pPr>
  </a:lstStyle>
  <a:p>
    <a:pPr lvl="8">
    </a:pPr>
    …
    <a:t>Some text</a:t>
    …
  </a:p>
</p:txBody>
```

end example]

[*Note*: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The rule here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. *end note*]

Parent Elements
bodyStyle (§4.4.1.5); defaultTextStyle (§4.3.1.7); lstStyle (§5.1.5.4.12); notesStyle (§4.4.1.25); otherStyle (§4.4.1.32); titleStyle (§4.4.1.45)

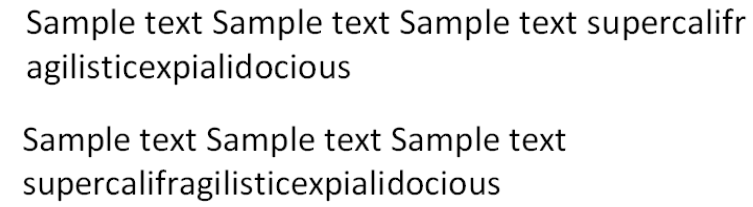
Child Elements	Subclause
buAutoNum (Auto-Numbered Bullet)	§5.1.5.4.1
buBlip (Picture Bullet)	§5.1.5.4.2
buChar (Character Bullet)	§5.1.5.4.3

Child Elements	Subclause
buClr (Color Specified)	§5.1.5.4.4
buClrTx (Follow Text)	§5.1.5.4.5
buFont (Specified)	§5.1.5.4.6
buFontTx (Follow text)	§5.1.5.4.7
buNone (No Bullet)	§5.1.5.4.8
buSzPct (Bullet Size Percentage)	§5.1.5.4.9
buSzPts (Bullet Size Points)	§5.1.5.4.10
buSzTx (Bullet Size Follows Text)	§5.1.5.4.11
defRPr (Default Text Run Properties)	§5.1.5.3.2
extLst (Extension List)	§5.1.2.1.15
lnSpc (Line Spacing)	§5.1.5.2.5
spcAft (Space After)	§5.1.5.2.8
spcBef (Space Before)	§5.1.5.2.9
tabLst (Tab List)	§5.1.5.2.13

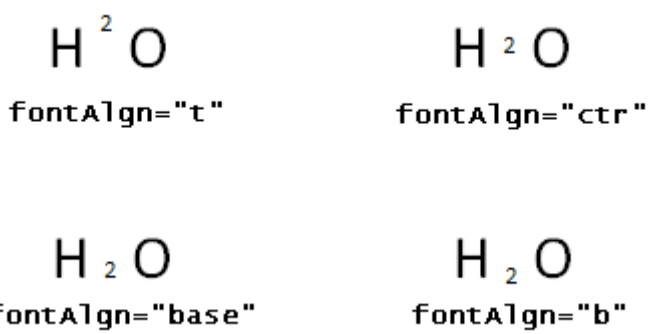
Attributes	Description
algn (Alignment)	Specifies the alignment that is to be applied to the paragraph. Possible values for this include left, right, centered, justified and distributed. If this attribute is omitted, then a value of left is implied. Sample text [*Example*: Consider the case where the user wishes to have two columns of text that have a justified alignment, much like text within a book. The following DrawingML could describe this. `<p:txBody>` `<a:bodyPr numCol="2" spcCol="914400"…/>`

Attributes	Description
	<a:normAutofit/> </a:bodyPr> … <a:p> <a:pPr marL="0" algn="just"> <a:buNone/> </a:pPr> … <a:t>Sample Text …</a:t> … </a:p> </p:txBody> *end example*] The possible values for this attribute are defined by the ST_TextAlignType simple type (§5.1.12.59).
defTabSz (Default Tab Size)	Specifies the default size for a tab character within this paragraph. This attribute should be used to describe the spacing of tabs within the paragraph instead of a leading indentation tab. For indentation tabs there are the marL and indent attributes to assist with this. [*Example*: Consider the case where a paragraph contains numerous tabs that need to be of a specific size. The following DrawingML would describe this. <p:txBody> … <a:p> <a:pPr defTabSz="376300" …/> … <a:t>Sample Text …</a:t> … </a:p> </p:txBody> *end example*] The possible values for this attribute are defined by the ST_Coordinate32 simple type (§5.1.12.17).
eaLnBrk (East Asian Line Break)	Specifies whether an East Asian word can be broken in half and wrapped onto the next line without a hyphen being added. To determine whether an East Asian word can be broken the presentation application would use the kinsoku settings here. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable East Asian words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.

Attributes	Description

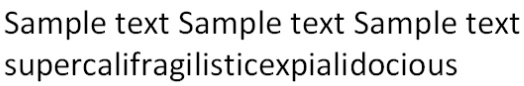
	[*Example*: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in whole on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.
	
	`<p:txBody>`
	`  …`
	`  <a:p>`
	`  <a:pPr eaLnBrk="0" …/>`
	`  …`
	`  <a:t>Sample text (Long word)</a:t>`
	`  …`
	`  </a:p>`
	`</p:txBody>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

fontAlgn (Font Alignment)	Determines where vertically on a line of text the actual words are positioned. This deals with vertical placement of the characters with respect to the baselines. For instance having text anchored to the top baseline, anchored to the bottom baseline, centered in between, etc. To understand this attribute and it's use it is helpful to understand what baselines are. A diagram describing these different cases is shown below. If this attribute is omitted, then a value of base is implied. [*Example*: Consider the case where the user wishes to represent the chemical compound of a water molecule. For this they will need to make sure the H, the 2, and the O are all in the correct position and are of the correct size. The results below can be achieved through the DrawingML shown below.

Attributes	Description
	 ``` <a:txtBody> … <a:pPr fontAlgn="b" …/> … <a:r> <a:rPr …/> <a:t>H </a:t> </a:r> <a:r> <a:rPr sz="1200" …/> <a:t>2</a:t> </a:r> <a:r> <a:rPr …/> <a:t>O</a:t> </a:r> … </p:txBody> ``` *end example*] The possible values for this attribute are defined by the ST_TextFontAlignType simple type (§5.1.12.66).
hangingPunct (Hanging Punctuation)	Specifies whether punctuation is to be forcefully laid out on a line of text or put on a different line of text. That is, if there is punctuation at the end of a run of text that should be carried over to a separate line does it actually get carried over. A true value will allow for hanging punctuation forcing the punctuation to not be carried over and a value of false will allow the punctuation to be carried onto the next text line. If this attribute is omitted, then a value of 0, or false is implied. The possible values for this attribute are defined by the XML Schema boolean datatype.
indent (Indent)	Specifies the indent size that will be applied to the first line of text in the paragraph. An indentation of 0 will be considered to be at the same location as marL attribute. If this attribute is omitted, then a value of -342900 is implied.

Attributes	Description
	Here is some Sample text [*Example*: Consider the scenario where the user now wanted to add a paragraph indentation to the first line of text in their two column format book. `<p:txBody>` `  <a:bodyPr numCol="2" spcCol="914400"…/>` `    <a:normAutofit/>` `    </a:bodyPr>` `  …` `  <a:p>` `    <a:pPr marL="0" indent="571500" algn="just">` `      <a:buNone/>` `    </a:pPr>` `    …` `    <a:t>Here is some…</a:t>` `    …` `  </a:p>` `</p:txBody>` By adding the indent attribute the user has effectively added a first line indent to this paragraph of text. *end example*] The possible values for this attribute are defined by the ST_TextIndent simple type (§5.1.12.70).
latinLnBrk (Latin Line Break)	Specifies whether a Latin word can be broken in half and wrapped onto the next line without a hyphen being added. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable Latin words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied. [*Example*: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in whole on a new line so that it

Attributes	Description

	may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.
	
	`<p:txBody>`
	`  …`
	`  <a:p>`
	`  <a:pPr latinLnBrk="0" …/>`
	`  …`
	`  <a:t>Sample text (Long word)</a:t>`
	`  …`
	`  </a:p>`
	`</p:txBody>`
	
	 
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

lvl (Level)	Specifies the particular level text properties that this paragraph will follow. The value for this attribute is numerical and formats the text according to the corresponding level paragraph properties that are listed within the lstStyle element. Since there are nine separate level properties defined, this tag will have an effective range of 0-8 = 9 available values.
	
	[*Example*: Consider the following DrawingML. This would specify that this paragraph should follow the lvl2pPr formatting style because once again lvl="1" is considered to be level 2.
	
	`<p:txBody>`
	`  …`
	`  <a:p>`
	`  <a:pPr lvl="1" …/>`
	`  …`
	`  <a:t>Sample text</a:t>`
	`  …`
	`  </a:p>`
	`</p:txBody>`
	
	end example]
	
	[*Note*: To resolve conflicting paragraph properties the linear hierarchy of paragraph

Attributes	Description
	properties should be examined starting first with the pPr element. The rule here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. *end note*] The possible values for this attribute are defined by the ST_TextIndentLevelType simple type (§5.1.12.71).
marL (Left Margin)	Specifies the left margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marL attributes are additive with respect to the text position. If this attribute is omitted, then a value of 347663 is implied. The possible values for this attribute are defined by the ST_TextMargin simple type (§5.1.12.73).
marR (Right Margin)	Specifies the right margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marR attributes are additive with respect to the text position. If this attribute is omitted, then a value of 0 is implied. The possible values for this attribute are defined by the ST_TextMargin simple type (§5.1.12.73).

rtl (Right To Left)	Specifies whether the text is right-to-left or left-to-right in its flow direction. If this attribute is omitted, then a value of 0, or left-to-right is implied.
	
	[*Example*: Consider the scenario where the user wanted text to flow from right to left. That is within the bounding text box the first word would be right aligned with each additional word being written to the left of the previous while continuing to flow top to bottom. The DrawingML to describe this might be as follows.
	
	`<p:txBody>`
	`…`
	`<a:p>`
	`<a:pPr rtl="1" …/>`
	`…`
	`<a:t>Sample text…</a:t>`
	`…`
	`</a:p>`
	`</p:txBody>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextParagraphProperties">
   <sequence>
      <element name="lnSpc" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <element name="spcBef" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <element name="spcAft" type="CT_TextSpacing" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletColor" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletSize" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBulletTypeface" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_TextBullet" minOccurs="0" maxOccurs="1"/>
      <element name="tabLst" type="CT_TextTabStopList" minOccurs="0" maxOccurs="1"/>
      <element name="defRPr" type="CT_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="marL" type="ST_TextMargin" use="optional"/>
   <attribute name="marR" type="ST_TextMargin" use="optional"/>
   <attribute name="lvl" type="ST_TextIndentLevelType" use="optional"/>
   <attribute name="indent" type="ST_TextIndent" use="optional"/>
   <attribute name="algn" type="ST_TextAlignType" use="optional"/>
   <attribute name="defTabSz" type="ST_Coordinate32" use="optional"/>
   <attribute name="rtl" type="xsd:boolean" use="optional"/>
   <attribute name="eaLnBrk" type="xsd:boolean" use="optional"/>
   <attribute name="fontAlgn" type="ST_TextFontAlignType" use="optional"/>
   <attribute name="latinLnBrk" type="xsd:boolean" use="optional"/>
   <attribute name="hangingPunct" type="xsd:boolean" use="optional"/>
</complexType>
```

5.1.6 Tables

This section contains information regarding the definition of a table within DrawingML. The following image is an example table within DrawingML.

text	text	text	text	text
text	text	text	text	text
text	text	text	text	text
text	text	text	text	text
text	text	text	text	text

5.1.6.1 cell3D (Cell 3-D)

This element specifies a set of properties which dictate the 3-D appearance of a given cell in a table. Collectively, these properties are referred to as a cell 3-D. The application of these properties occurs on a per-cell basis in the table.

Parent Elements
tcPr (§5.1.6.15); tcStyle (§5.1.4.2.29)

Child Elements	Subclause
bevel (Bevel)	§5.1.4.2.5
extLst (Extension List)	§5.1.2.1.15
lightRig (Light Rig)	§5.1.7.9

Attributes	Description
prstMaterial (Preset Material)	Specifies a material type which will be used to define the material characteristics of the cell. The material properties, combined with the lighting characteristics of the scene in define the final look and feel of the 3-D appearance of the cell. The possible values for this attribute are defined by the ST_PresetMaterialType simple type (§5.1.12.50).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Cell3D">
   <sequence>
      <element name="bevel" type="CT_Bevel" minOccurs="1" maxOccurs="1"/>
      <element name="lightRig" type="CT_LightRig" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="prstMaterial" type="ST_PresetMaterialType" use="optional" default="plastic"/>
</complexType>
```

5.1.6.2 gridCol (Table Grid Column)

This element specifies the width of a given column within a table. For each column in a table, there will be an associated table grid column defining the width of the column.

[*Example*: Consider the following example of a table grid containing widths defined for three table grid columns:

```
<a:tblGrid>
  <a:gridCol w="1117600"/>
  <a:gridCol w="1117600"/>
```

```
  <a:gridCol w="1117600"/>
</a:tblGrid>
```

end example]

Parent Elements
tblGrid (§5.1.6.12)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15

Attributes	Description
w (Width)	The width of the column in EMUs. The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TableCol">
   <sequence>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="w" type="ST_Coordinate" use="required"/>
</complexType>
```

5.1.6.3 lnB (Bottom Border Line Properties)

This element defines the line properties associated with the bottom border of a given cell.

[*Example*: Consider the following example of a bottom border line properties element within DrawingML:

```
<a:lnB w="38100" cap="flat" cmpd="sng" algn="ctr">
  <a:solidFill>
    <a:schemeClr val="accent2"/>
  </a:solidFill>
  <a:prstDash val="solid"/>
  <a:round/>
  <a:headEnd type="none" w="med" len="med"/>
  <a:tailEnd type="none" w="med" len="med"/>
</a:lnB>
```

In this example, one can see that the bottom border line style defined with certain properties, such as a flat end line cap, a given width, head and tail end, color, etc. *end example*]

Parent Elements
tcPr (§5.1.6.15)

Child Elements	Subclause
bevel (Line Join Bevel)	§5.1.10.9
custDash (Custom Dash)	§5.1.10.21
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
headEnd (Line Head/End Style)	§5.1.10.38
miter (Miter Line Join)	§5.1.10.43
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
prstDash (Preset Dash)	§5.1.10.48
round (Round Line Join)	§5.1.10.52
solidFill (Solid Fill)	§5.1.10.54
tailEnd (Tail line end style)	§5.1.10.57

Attributes	Description
algn (Stroke Alignment)	Specifies the alignment to be used for the underline stroke. The possible values for this attribute are defined by the ST_PenAlignment simple type (§5.1.12.40).
cap (Line Ending Cap Type)	Specifies the ending caps that should be used for this line. Examples of cap types are rounded, flat, etc. If this attribute is omitted, than a value of square is assumed. The possible values for this attribute are defined by the ST_LineCap simple type (§5.1.12.31).
cmpd (Compound Line Type)	Specifies the compound line type to be used for the underline stroke. If this attribute is omitted, then a value of sng is assumed. The possible values for this attribute are defined by the ST_CompoundLine simple type (§5.1.12.15).
w (Line Width)	Specifies the width to be used for the underline stroke. If this attribute is omitted, then a value of 0 is assumed. The possible values for this attribute are defined by the ST_LineWidth simple type

Attributes	Description
	(§5.1.12.35).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LineProperties">
   <sequence>
      <group ref="EG_LineFillProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_LineDashProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_LineJoinProperties" minOccurs="0" maxOccurs="1"/>
      <element name="headEnd" type="CT_LineEndProperties" minOccurs="0" maxOccurs="1"/>
      <element name="tailEnd" type="CT_LineEndProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="w" type="ST_LineWidth" use="optional"/>
   <attribute name="cap" type="ST_LineCap" use="optional"/>
   <attribute name="cmpd" type="ST_CompoundLine" use="optional"/>
   <attribute name="algn" type="ST_PenAlignment" use="optional"/>
</complexType>
```

5.1.6.4 lnBlToTr (Bottom-Left to Top-Right Border Line Properties)

This element defines the line properties associated with the diagonal line from the bottom left corner of the cell to the top right corner.

[*Example*: Consider the following example of a lnBlToTr within DrawingML:

```
<a:lnBlToTr w="38100" cap="flat" cmpd="sng" algn="ctr">
  <a:solidFill>
    <a:schemeClr val="accent2"/>
  </a:solidFill>
  <a:prstDash val="solid"/>
  <a:round/>
  <a:headEnd type="none" w="med" len="med"/>
  <a:tailEnd type="none" w="med" len="med"/>
</a:lnBlToTr >
```

In this example, one can see that the border line style defined with certain properties, such as a flat end line cap, a given width, head and tail end, color, etc. *end example*]

Parent Elements
tcPr (§5.1.6.15)

Child Elements	Subclause
bevel (Line Join Bevel)	§5.1.10.9
custDash (Custom Dash)	§5.1.10.21

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
headEnd (Line Head/End Style)	§5.1.10.38
miter (Miter Line Join)	§5.1.10.43
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
prstDash (Preset Dash)	§5.1.10.48
round (Round Line Join)	§5.1.10.52
solidFill (Solid Fill)	§5.1.10.54
tailEnd (Tail line end style)	§5.1.10.57

Attributes	Description
algn (Stroke Alignment)	Specifies the alignment to be used for the underline stroke. The possible values for this attribute are defined by the ST_PenAlignment simple type (§5.1.12.40).
cap (Line Ending Cap Type)	Specifies the ending caps that should be used for this line. Examples of cap types are rounded, flat, etc. If this attribute is omitted, than a value of square is assumed. The possible values for this attribute are defined by the ST_LineCap simple type (§5.1.12.31).
cmpd (Compound Line Type)	Specifies the compound line type to be used for the underline stroke. If this attribute is omitted, then a value of sng is assumed. The possible values for this attribute are defined by the ST_CompoundLine simple type (§5.1.12.15).
w (Line Width)	Specifies the width to be used for the underline stroke. If this attribute is omitted, then a value of 0 is assumed. The possible values for this attribute are defined by the ST_LineWidth simple type (§5.1.12.35).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LineProperties">
   <sequence>
      <group ref="EG_LineFillProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_LineDashProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_LineJoinProperties" minOccurs="0" maxOccurs="1"/>
      <element name="headEnd" type="CT_LineEndProperties" minOccurs="0" maxOccurs="1"/>
      <element name="tailEnd" type="CT_LineEndProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="w" type="ST_LineWidth" use="optional"/>
   <attribute name="cap" type="ST_LineCap" use="optional"/>
   <attribute name="cmpd" type="ST_CompoundLine" use="optional"/>
   <attribute name="algn" type="ST_PenAlignment" use="optional"/>
</complexType>
```

5.1.6.5 lnL (Left Border Line Properties)

This element defines the line properties associated with the left border of a cell

[*Example*: Consider the following example of a lnL within DrawingML:

```
<a:lnL w="38100" cap="flat" cmpd="sng" algn="ctr">
  <a:solidFill>
    <a:schemeClr val="accent2"/>
  </a:solidFill>
  <a:prstDash val="solid"/>
  <a:round/>
  <a:headEnd type="none" w="med" len="med"/>
  <a:tailEnd type="none" w="med" len="med"/>
</a:lnL >
```

In this example, one can see that the border line style defined with certain properties, such as a flat end line cap, a given width, head and tail end, color, etc. *end example*]

Parent Elements
tcPr (§5.1.6.15)

Child Elements	Subclause
bevel (Line Join Bevel)	§5.1.10.9
custDash (Custom Dash)	§5.1.10.21
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
headEnd (Line Head/End Style)	§5.1.10.38
miter (Miter Line Join)	§5.1.10.43

Child Elements	Subclause
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
prstDash (Preset Dash)	§5.1.10.48
round (Round Line Join)	§5.1.10.52
solidFill (Solid Fill)	§5.1.10.54
tailEnd (Tail line end style)	§5.1.10.57

Attributes	Description
algn (Stroke Alignment)	Specifies the alignment to be used for the underline stroke. The possible values for this attribute are defined by the ST_PenAlignment simple type (§5.1.12.40).
cap (Line Ending Cap Type)	Specifies the ending caps that should be used for this line. Examples of cap types are rounded, flat, etc. If this attribute is omitted, than a value of square is assumed. The possible values for this attribute are defined by the ST_LineCap simple type (§5.1.12.31).
cmpd (Compound Line Type)	Specifies the compound line type to be used for the underline stroke. If this attribute is omitted, then a value of sng is assumed. The possible values for this attribute are defined by the ST_CompoundLine simple type (§5.1.12.15).
w (Line Width)	Specifies the width to be used for the underline stroke. If this attribute is omitted, then a value of 0 is assumed. The possible values for this attribute are defined by the ST_LineWidth simple type (§5.1.12.35).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LineProperties">
   <sequence>
      <group ref="EG_LineFillProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_LineDashProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_LineJoinProperties" minOccurs="0" maxOccurs="1"/>
      <element name="headEnd" type="CT_LineEndProperties" minOccurs="0" maxOccurs="1"/>
      <element name="tailEnd" type="CT_LineEndProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="w" type="ST_LineWidth" use="optional"/>
   <attribute name="cap" type="ST_LineCap" use="optional"/>
   <attribute name="cmpd" type="ST_CompoundLine" use="optional"/>
   <attribute name="algn" type="ST_PenAlignment" use="optional"/>
</complexType>
```

5.1.6.6 lnR (Right Border Line Properties)

This element defines the line properties associated with right border of a cell.

[*Example*: Consider the following example of a lnR within DrawingML:

```
<a:lnR w="38100" cap="flat" cmpd="sng" algn="ctr">
  <a:solidFill>
    <a:schemeClr val="accent2"/>
  </a:solidFill>
  <a:prstDash val="solid"/>
  <a:round/>
  <a:headEnd type="none" w="med" len="med"/>
  <a:tailEnd type="none" w="med" len="med"/>
</a:lnR >
```

In this example, one can see that the border line style defined with certain properties, such as a flat end line cap, a given width, head and tail end, color, etc. *end example*]

Parent Elements
tcPr (§5.1.6.15)

Child Elements	Subclause
bevel (Line Join Bevel)	§5.1.10.9
custDash (Custom Dash)	§5.1.10.21
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
headEnd (Line Head/End Style)	§5.1.10.38
miter (Miter Line Join)	§5.1.10.43
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
prstDash (Preset Dash)	§5.1.10.48
round (Round Line Join)	§5.1.10.52
solidFill (Solid Fill)	§5.1.10.54
tailEnd (Tail line end style)	§5.1.10.57

Attributes	Description
algn (Stroke Alignment)	Specifies the alignment to be used for the underline stroke. The possible values for this attribute are defined by the ST_PenAlignment simple type

Attributes	Description
	(§5.1.12.40).
cap (Line Ending Cap Type)	Specifies the ending caps that should be used for this line. Examples of cap types are rounded, flat, etc. If this attribute is omitted, than a value of square is assumed. The possible values for this attribute are defined by the ST_LineCap simple type (§5.1.12.31).
cmpd (Compound Line Type)	Specifies the compound line type to be used for the underline stroke. If this attribute is omitted, then a value of sng is assumed. The possible values for this attribute are defined by the ST_CompoundLine simple type (§5.1.12.15).
w (Line Width)	Specifies the width to be used for the underline stroke. If this attribute is omitted, then a value of 0 is assumed. The possible values for this attribute are defined by the ST_LineWidth simple type (§5.1.12.35).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LineProperties">
   <sequence>
      <group ref="EG_LineFillProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_LineDashProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_LineJoinProperties" minOccurs="0" maxOccurs="1"/>
      <element name="headEnd" type="CT_LineEndProperties" minOccurs="0" maxOccurs="1"/>
      <element name="tailEnd" type="CT_LineEndProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="w" type="ST_LineWidth" use="optional"/>
   <attribute name="cap" type="ST_LineCap" use="optional"/>
   <attribute name="cmpd" type="ST_CompoundLine" use="optional"/>
   <attribute name="algn" type="ST_PenAlignment" use="optional"/>
</complexType>
```

5.1.6.7 lnT (Top Border Line Properties)

This element defines the line properties associated with the top border of a cell.

[*Example*: Consider the following example of a lnT within DrawingML:

```
<a:lnT w="38100" cap="flat" cmpd="sng" algn="ctr">
  <a:solidFill>
    <a:schemeClr val="accent2"/>
  </a:solidFill>
  <a:prstDash val="solid"/>
  <a:round/>
  <a:headEnd type="none" w="med" len="med"/>
```

```
  <a:tailEnd type="none" w="med" len="med"/>
</a:lnT >
```

In this example, one can see that the border line style defined with certain properties, such as a flat end line cap, a given width, head and tail end, color, etc. *end example*]

Parent Elements
tcPr (§5.1.6.15)

Child Elements	Subclause
bevel (Line Join Bevel)	§5.1.10.9
custDash (Custom Dash)	§5.1.10.21
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
headEnd (Line Head/End Style)	§5.1.10.38
miter (Miter Line Join)	§5.1.10.43
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
prstDash (Preset Dash)	§5.1.10.48
round (Round Line Join)	§5.1.10.52
solidFill (Solid Fill)	§5.1.10.54
tailEnd (Tail line end style)	§5.1.10.57

Attributes	Description
algn (Stroke Alignment)	Specifies the alignment to be used for the underline stroke. The possible values for this attribute are defined by the ST_PenAlignment simple type (§5.1.12.40).
cap (Line Ending Cap Type)	Specifies the ending caps that should be used for this line. Examples of cap types are rounded, flat, etc. If this attribute is omitted, than a value of square is assumed. The possible values for this attribute are defined by the ST_LineCap simple type (§5.1.12.31).
cmpd (Compound Line Type)	Specifies the compound line type to be used for the underline stroke. If this attribute is omitted, then a value of sng is assumed. The possible values for this attribute are defined by the ST_CompoundLine simple type (§5.1.12.15).
w (Line Width)	Specifies the width to be used for the underline stroke. If this attribute is omitted, then a

Attributes	Description
	value of 0 is assumed. The possible values for this attribute are defined by the ST_LineWidth simple type (§5.1.12.35).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LineProperties">
   <sequence>
      <group ref="EG_LineFillProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_LineDashProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_LineJoinProperties" minOccurs="0" maxOccurs="1"/>
      <element name="headEnd" type="CT_LineEndProperties" minOccurs="0" maxOccurs="1"/>
      <element name="tailEnd" type="CT_LineEndProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="w" type="ST_LineWidth" use="optional"/>
   <attribute name="cap" type="ST_LineCap" use="optional"/>
   <attribute name="cmpd" type="ST_CompoundLine" use="optional"/>
   <attribute name="algn" type="ST_PenAlignment" use="optional"/>
</complexType>
```

5.1.6.8 lnTlToBr (Top-Left to Bottom-Right Border Line Properties)

This element defines the line properties associated with the diagonal line from the top left corner of the cell to the bottom right corner.

[*Example*: Consider the following example of a lnTlToBr within DrawingML:

```
<a:lnTlToBr w="38100" cap="flat" cmpd="sng" algn="ctr">
  <a:solidFill>
    <a:schemeClr val="accent2"/>
  </a:solidFill>
  <a:prstDash val="solid"/>
  <a:round/>
  <a:headEnd type="none" w="med" len="med"/>
  <a:tailEnd type="none" w="med" len="med"/>
</a:lnTlToBr>
```

In this example, one can see that the border line style defined with certain properties, such as a flat end line cap, a given width, head and tail end, color, etc. *end example*]

Parent Elements
tcPr (§5.1.6.15)

Child Elements	Subclause

Child Elements	Subclause
bevel (Line Join Bevel)	§5.1.10.9
custDash (Custom Dash)	§5.1.10.21
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
headEnd (Line Head/End Style)	§5.1.10.38
miter (Miter Line Join)	§5.1.10.43
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
prstDash (Preset Dash)	§5.1.10.48
round (Round Line Join)	§5.1.10.52
solidFill (Solid Fill)	§5.1.10.54
tailEnd (Tail line end style)	§5.1.10.57

Attributes	Description
algn (Stroke Alignment)	Specifies the alignment to be used for the underline stroke. The possible values for this attribute are defined by the ST_PenAlignment simple type (§5.1.12.40).
cap (Line Ending Cap Type)	Specifies the ending caps that should be used for this line. Examples of cap types are rounded, flat, etc. If this attribute is omitted, than a value of square is assumed. The possible values for this attribute are defined by the ST_LineCap simple type (§5.1.12.31).
cmpd (Compound Line Type)	Specifies the compound line type to be used for the underline stroke. If this attribute is omitted, then a value of sng is assumed. The possible values for this attribute are defined by the ST_CompoundLine simple type (§5.1.12.15).
w (Line Width)	Specifies the width to be used for the underline stroke. If this attribute is omitted, then a value of 0 is assumed. The possible values for this attribute are defined by the ST_LineWidth simple type (§5.1.12.35).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LineProperties">
   <sequence>
      <group ref="EG_LineFillProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_LineDashProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_LineJoinProperties" minOccurs="0" maxOccurs="1"/>
      <element name="headEnd" type="CT_LineEndProperties" minOccurs="0" maxOccurs="1"/>
      <element name="tailEnd" type="CT_LineEndProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="w" type="ST_LineWidth" use="optional"/>
   <attribute name="cap" type="ST_LineCap" use="optional"/>
   <attribute name="cmpd" type="ST_CompoundLine" use="optional"/>
   <attribute name="algn" type="ST_PenAlignment" use="optional"/>
</complexType>
```

5.1.6.9 tableStyle (Table Style)

This element specifies a particular table style. Fourteen elements make up the styling information of a given table style. These fourteen elements work together to provide visual formatting options for on/off states of the following toggles:

- First row on/off - Associated element: firstRow
- Last row on/off - Associated element: lastRow
- First column on/off - Associated element: firstCol
- Last column on/off - Associated element: lastCol
- Row banding on/off - Associated elements: band1H, band2H
- Column banding on/off - Associated elements: band1V, band2V

The formatting associated with the wholeTbl element defines the table formatting when all options are off. When an option is turned on, the formatting for that particular option is applied to the table. The four cell specific formatting options are enabled when overlapping options are toggled on. For example, when the first row, and first column formatting options are enabled, any formatting within the northwest cell will also be applied since that is the overlapping table cell when both first column and first row formatting options are on.

[*Example*: Consider the following partial example of a tblStyle within DrawingML:

```
<a:tblStyle styleId="{5940675A-B579-460E-94D1-54222C63F5DA}"
  styleName="No Style, Table Grid">
  <a:wholeTbl>
    <a:tcTxStyle>
      <a:fontRef idx="minor">
        <a:scrgbClr r="0" g="0" b="0"/>
      </a:fontRef>
```

```
        <a:schemeClr val="tx1"/>
      </a:tcTxStyle>
      <a:tcStyle>
        <a:tcBdr>
          <a:left>
            <a:ln w="12700" cmpd="sng">
              <a:solidFill>
                <a:schemeClr val="tx1"/>
              </a:solidFill>
            </a:ln>
          </a:left>
```

…right, top, bottom, insideH, insideV border information is defined just as the 'left' tag…

```
        </a:tcBdr>
        <a:fill>
          <a:noFill/>
        </a:fill>
      </a:tcStyle>
    </a:wholeTbl>
    <a:band1H>
      <a:tcStyle>
        <a:tcBdr/>
      </a:tcStyle>
    </a:band1H>
```

…band2H, band1V, band2V, firstCol, firstRow, lastCol, lastRow, neCell, nwCell, seCell, swCell tags are all defined just as the 'band1H' tag

```
  </a:tblStyle>
```

In this example, one can get an idea for the definition of a table style in its entirety. The above defined table style will create a style with only 1pt line formatting applied to all of the cells in a table. Notice that the on/off toggle formatting (band1H, band2H, firstCol, etc) do not define any formatting and therefore have no effect to the table when toggled. *end example*]

Parent Elements
tblPr (§5.1.6.13)

Child Elements	Subclause
band1H (Band 1 Horizontal)	§5.1.4.2.1
band1V (Band 1 Vertical)	§5.1.4.2.2
band2H (Band 2 Horizontal)	§5.1.4.2.3

Child Elements	Subclause
band2V (Band 2 Vertical)	§5.1.4.2.4
extLst (Extension List)	§5.1.2.1.15
firstCol (First Column)	§5.1.4.2.11
firstRow (First Row)	§5.1.4.2.12
lastCol (Last Column)	§5.1.4.2.16
lastRow (Last Row)	§5.1.4.2.17
neCell (Northeast Cell)	§5.1.4.2.20
nwCell (Northwest Cell)	§5.1.4.2.21
seCell (Southeast Cell)	§5.1.4.2.23
swCell (Southwest Cell)	§5.1.4.2.24
tblBg (Table Background)	§5.1.4.2.25
wholeTbl (Whole Table)	§5.1.4.2.34

Attributes	Description
styleId (Style ID)	Specifies a GUID identifying the table style in a unique manner. The possible values for this attribute are defined by the ST_Guid simple type (§5.1.12.27).
styleName (Name)	Specifies the name of the table style which can show up in the user interface identifying the style to a user. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TableStyle">
   <sequence>
      <element name="tblBg" type="CT_TableBackgroundStyle" minOccurs="0" maxOccurs="1"/>
      <element name="wholeTbl" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="band1H" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="band2H" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="band1V" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="band2V" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="lastCol" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="firstCol" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="lastRow" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="seCell" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="swCell" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="firstRow" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="neCell" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="nwCell" type="CT_TablePartStyle" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="styleId" type="ST_Guid" use="required"/>
   <attribute name="styleName" type="xsd:string" use="required"/>
</complexType>
```

5.1.6.10 tableStyleId (Table Style ID)

This element defines the table style which is currently applied to the table by referencing the styleId attribute of the tableStyle element.

[*Example*: Consider the following example of a tableStyleId within DrawingML:

```
<a:tblPr firstRow="1" bandRow="1">
  <a:tableStyleId>{5940675A-B579-460E-94D1-54222C63F5DA}</a:tableStyleId>
</a:tblPr>
```

In this example, we see a reference to a table style being specified in the tableStyleId element. *end example*]

The possible values for this element are defined by the ST_Guid simple type (§5.1.12.27).

Parent Elements
tblPr (§5.1.6.13)

5.1.6.11 tbl (Table)

This element is the root element for a table. Within this element is contained everything that one would need to define a table within DrawingML.

[*Example*: Consider the following example of a tbl within DrawingML:

```
<a:tbl>
  <a:tblPr firstRow="1" bandRow="1">
  …
  <a:tblPr>
  <a:tblGrid>
  …
  </a:tblGrid>
  <a:tr h="419100">
  …
  </a:tr>
</a:tbl>
```

In this example, we see can see the definition of a table within DrawingML. *end example*]

Child Elements	Subclause
tblGrid (Table Grid)	§5.1.6.12
tblPr (Table Properties)	§5.1.6.13
tr (Table Row)	§5.1.6.16

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Table">
   <sequence>
      <element name="tblPr" type="CT_TableProperties" minOccurs="0" maxOccurs="1"/>
      <element name="tblGrid" type="CT_TableGrid" minOccurs="1" maxOccurs="1"/>
      <element name="tr" type="CT_TableRow" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.1.6.12 tblGrid (Table Grid)

This element defines a list of table column (§5.1.6.2) elements. There should be a table column (§5.1.6.2) element for every column held within the table.

[*Example*: Consider the following example of a tblGrid within DrawingML:

```
<a:tblGrid>
  <a:gridCol w="1117600"/>
  <a:gridCol w="1117600"/>
  <a:gridCol w="1117600"/>
</a:tblGrid>
```

In this example, we have a tblGrid defined that holds three columns, therefore the table will have three columns. *end example*]

Parent Elements
tbl (§5.1.6.11)

Child Elements	Subclause
gridCol (Table Grid Column)	§5.1.6.2

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TableGrid">
   <sequence>
      <element name="gridCol" type="CT_TableCol" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.1.6.13 tblPr (Table Properties)

This element defines the properties of a table on the whole. Within this element are many visual modifications that can be applied to the table.

[*Example*: Consider the following example of a tblPr within DrawingML:

```
<a:tblPr firstRow="1" bandRow="1">
  <a:tableStyleId>{5940675A-B579-460E-94D1-54222C63F5DA}</a:tableStyleId>
</a:tblPr>
```

In this example, we see that there is a link to a table style id (§5.1.6.10) which is defined elsewhere and that the first column formatting and banded row formatting has been enabled. The table style defines the formatting applied with the two formatting options enabled. *end example*]

Parent Elements
tbl (§5.1.6.11)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
effectDag (Effect Container)	§5.1.10.25
effectLst (Effect Container)	§5.1.10.26
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
solidFill (Solid Fill)	§5.1.10.54
tableStyle (Table Style)	§5.1.6.9
tableStyleId (Table Style ID)	§5.1.6.10

Attributes	Description

bandCol (Banded Columns)	Enables or disables the banded column formatting for a table style. A value of on, 1 or true will enable the banded column formatting defined in the table style. The attribute will default to off if it is not specified.
	
	[*Example*: Consider the following run:
	
	`<a:tblPr bandCol="1">`
	`  <a:tableStyleId>{5940675A-B579-460E-94D1-`
	`  54222C63F5DA}</a:tableStyleId>`
	`</a:tblPr>`
	
	In this example, we can see the banded column formatting is enabled for the table. When applied, the linked table style defines the formatting for banded columns. *end example*]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

bandRow (Banded Rows)	Enables or disables the banded row formatting for a table style. A value of on, 1 or true will enable the banded row formatting defined in the table style. The attribute will default to off if it is not specified.
	
	[*Example*: Consider the following run:
	
	`<a:tblPr bandRow="1">`
	`  <a:tableStyleId>{5940675A-B579-460E-94D1-`
	`  54222C63F5DA}</a:tableStyleId>`
	`</a:tblPr>`
	
	In this example, we can see the banded row formatting is enabled for the table. When applied, the linked table style defines the formatting for banded rows. *end example*]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

firstCol (First Column)	Enables or disables the first column formatting for a table style. A value of on, 1 or true will enable the first column formatting defined in the table style. The attribute will default to off if it is not specified.
	
	[*Example*: Consider the following run:
	
	`<a:tblPr firstCol="1">`
	`  <a:tableStyleId>{5940675A-B579-460E-94D1-`
	`  54222C63F5DA}</a:tableStyleId>`
	`</a:tblPr>`
	
	In this example, we can see the first column formatting is enabled for the table. When

Attributes	Description
	applied, the linked table style defines the formatting for the first column. *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.

firstRow (First Row)	Enables or disables the first row formatting for a table style. A value of on, 1 or true will enable the first row formatting defined in the table style. The attribute will default to off if it is not specified.
	
	[*Example*: Consider the following run:
	
	`<a:tblPr firstRow="1">`
	`  <a:tableStyleId>{5940675A-B579-460E-94D1-`
	`  54222C63F5DA}</a:tableStyleId>`
	`</a:tblPr>`
	
	In this example, we can see the first row formatting is enabled for the table. When applied, the linked table style defines the formatting for the first row. *end example*]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

lastCol (Last Column)	Enables or disables the last column formatting for a table style. A value of on, 1 or true will enable the last column formatting defined in the table style. The attribute will default to off if it is not specified.
	
	[*Example*: Consider the following run:
	
	`<a:tblPr lastCol="1">`
	`  <a:tableStyleId>{5940675A-B579-460E-94D1-`
	`  54222C63F5DA}</a:tableStyleId>`
	`</a:tblPr>`
	
	In this example, we can see the last column formatting is enabled for the table. When applied, the linked table style defines the formatting for the last column. *end example*]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

lastRow (Last Row)	Enables or disables the last row formatting for a table style. A value of on, 1 or true will enable the last row formatting defined in the table style. The attribute will default to off if it is not specified.
	
	[*Example*: Consider the following run:
	
	`<a:tblPr lastRow="1">`
	`  <a:tableStyleId>{5940675A-B579-460E-94D1-`
	`  54222C63F5DA}</a:tableStyleId>`

Attributes	Description
	</a:tblPr> In this example, we can see the last row formatting is enabled for the table. When applied, the linked table style defines the formatting for the last row. *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
rtl (Right-to-Left)	Defines enables the right-to-left settings of a table. If the value of rtl is on, 1 or true, then the table will be laid out from the right-to-left rather than the default left-to-right. [*Example*: Consider the following run: <a:tblPr rtl="1"> <a:tableStyleId>{5940675A-B579-460E-94D1- 54222C63F5DA}</a:tableStyleId> </a:tblPr> In this example, we can see that the table is to be created in a right-to-left direction. *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TableProperties">
   <sequence>
      <group ref="EG_FillProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_EffectProperties" minOccurs="0" maxOccurs="1"/>
      <choice minOccurs="0" maxOccurs="1">
         <element name="tableStyle" type="CT_TableStyle"/>
         <element name="tableStyleId" type="ST_Guid"/>
      </choice>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="rtl" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="firstRow" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="firstCol" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="lastRow" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="lastCol" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="bandRow" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="bandCol" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.1.6.14 tc (Table Cell)

This element defines a cell within the table. The table cell holds a text body that actually contains the data held within the cell along with the properties of the table cell which hold formatting options associated with the cell.

[*Example*: Consider the following example of a tc within DrawingML:

```
<a:tc>
  <a:txBody>
    <a:bodyPr/>
    <a:lstStyle/>
    <a:p>
      <a:pPr marL="0" algn="ctr" rtl="0"/>
      <a:r>
        <a:rPr lang="en-US" dirty="0" smtClean="0"/>
        <a:t>data</a:t>
      </a:r>
      <a:endParaRPr lang="en-US" dirty="0"/>
    </a:p>
  </a:txBody>
  <a:tcPr/>
</a:tc>
```

In this example, we see a single cell in a table being defined with the default cell properties and a text body which contains the word "data". The text "data" will be the only text in the cell. *end example*]

Parent Elements
tr (§5.1.6.16)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
tcPr (Table Cell Properties)	§5.1.6.15
txBody (Shape Text Body)	§5.1.2.1.40

Attributes	Description

gridSpan (Grid Span)	Specifies the number of columns that a merged cell spans. This is used in combination with the hMerge attribute on other cells in order to specify the beginning cell of a horizontal merge.
	
	[*Example*: Consider the following example:
	
	`<a:tc gridSpan="3">`
	…
	`/a:tc>`
	`<a:tc hMerge="1">`
	…
	`/a:tc>`

Attributes	Description
	<a:tc hMerge="1"> … /a:tc> In this example, we can define what looks like a single cell in the table as a group of three cells merged together. The merged cell spans three columns of the table. *end example*] The possible values for this attribute are defined by the XML Schema int datatype.
hMerge (Horizontal Merge)	When this attribute is set to on, 1 or true, then this table cell is to be merged with the previous horizontal table cell when the table is created. [*Example*: Consider the following example: <a:tc hMerge="1"> … </a:tc> In this example, we see the hMerge attribute set to on which signifies that this cell is to be merged with the previous horizontal cell in the table. *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
rowSpan (Row Span)	Specifies the number of rows that a merged cell spans. This is used in combination with the vMerge attribute on other cells in order to specify the beginning cell of a horizontal merge. [*Example*: Consider the following example: <a:tc rowSpan="3"> … /a:tc> <a:tc vMerge="1"> … /a:tc> <a:tc vMerge="1"> … /a:tc> In this example, we can define what looks like a single cell in the table as a group of three cells merged together. The merged cell spans three rows of the table. *end example*] The possible values for this attribute are defined by the XML Schema int datatype.
vMerge (Vertical Merge)	When this attribute is set to on, 1 or true, then this table cell is to be merged with the previous vertical table cell when the table is created. [*Example*: Consider the following example:

Attributes	Description
	<a:tc vMerge="1"> … /a:tc> In this example, we see the vMerge attribute set to on which signifies that this cell is to be merged with the previous vertical cell in the table. *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TableCell">
   <sequence>
      <element name="txBody" type="CT_TextBody" minOccurs="0" maxOccurs="1"/>
      <element name="tcPr" type="CT_TableCellProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="rowSpan" type="xsd:int" use="optional" default="1"/>
   <attribute name="gridSpan" type="xsd:int" use="optional" default="1"/>
   <attribute name="hMerge" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="vMerge" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.1.6.15 tcPr (Table Cell Properties)

This element defines the formatting properties associated with a cell. The formatting options which are available to be adjusted range from the line types used for the borders to the cell fill to the margins associated with the layout of the text in the cell.

[*Example*: Consider the following example of a tcPr within DrawingML:

```
<a:tcPr marL="45720" marR="45720">
  <a:lnL w="38100" cap="flat" cmpd="sng" algn="ctr">
    <a:solidFill>
      <a:schemeClr val="accent2"/>
    </a:solidFill>
    <a:prstDash val="solid"/>
    <a:round/>
    <a:headEnd type="none" w="med" len="med"/>
    <a:tailEnd type="none" w="med" len="med"/>
  </a:lnL>
</a:tcPr>
```

In this example, we have a solid line defined as the left border of the cell along with left and right margin adjustments being made from the default margins. *end example*]

Parent Elements
tc (§5.1.6.14)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
cell3D (Cell 3-D)	§5.1.6.1
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35
lnB (Bottom Border Line Properties)	§5.1.6.3
lnBlToTr (Bottom-Left to Top-Right Border Line Properties)	§5.1.6.4
lnL (Left Border Line Properties)	§5.1.6.5
lnR (Right Border Line Properties)	§5.1.6.6
lnT (Top Border Line Properties)	§5.1.6.7
lnTlToBr (Top-Left to Bottom-Right Border Line Properties)	§5.1.6.8
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
solidFill (Solid Fill)	§5.1.10.54

Attributes	Description
anchor (Anchor)	Defines the alignment of the text vertically within the cell. [*Example*: Consider the following example: `<a:tcPr marL="45720" anchor="ctr"/>` In this example, the text in the cell will be anchored to the center of the cell vertically. *end example*] The possible values for this attribute are defined by the ST_TextAnchoringType simple type (§5.1.12.60).
anchorCtr (Anchor Center)	When this attribute is on, 1 or `true`, it modifies the anchor attribute. This attribute will center align the text box itself which will allow for text to be left aligned along the center of the cell for example. The possible values for this attribute are defined by the XML Schema boolean datatype.
horzOverflow (Horizontal	Specifies the clipping behavior of the cell. The two options here allow for the text to be clipped and out of view when outside of the bounds of the cell, or for the text to remain

Attributes	Description

Overflow)	visible and overflow outside of the cell.
	
	[*Example*: Consider the following example:
	
	`<a:tcPr horzOverflow="overflow">`
	`…`
	`</a:tcPr>`
	
	In this example, the text in the cell will freely overflow outside of the cell boundaries and will always remain visible. *end example*]
	
	The possible values for this attribute are defined by the ST_TextHorzOverflowType simple type (§5.1.12.69).

marB (Bottom Margin)	Specifies the bottom margin of the cell. The value specified in this attribute is the distance to offset from the bottom of the cell.
	
	[*Example*: Consider the following example:
	
	`<a:tcPr marB="45720" anchor="ctr">`
	`…`
	`</a:tcPr>`
	
	In this example, we have specified a value for the margin on the bottom of the cell. *end example*]
	
	The possible values for this attribute are defined by the ST_Coordinate32 simple type (§5.1.12.17).

marL (Left Margin)	This attribute specifies the left margin of the cell. The value specified in this attribute is the distance to offset from the left of the cell in EMU's.
	
	[*Example*: Consider the following example:
	
	`<a:tcPr marL="45720" anchor="ctr">`
	`…`
	`</a:tcPr>`
	
	In this example, we have specified a value for the margin on the left of the cell. *end example*]
	
	The possible values for this attribute are defined by the ST_Coordinate32 simple type (§5.1.12.17).

marR (Right Margin)	This attribute specifies the right margin of the cell. The value specified in this attribute is the distance to offset from the right of the cell in EMU's.

Attributes	Description

	[*Example*: Consider the following example:
	
	`<a:tcPr marR="45720" anchor="ctr">`
	`…`
	`</a:tcPr>`
	
	In this example, we have specified a value for the margin on the right of the cell. *end example*]
	
	The possible values for this attribute are defined by the ST_Coordinate32 simple type (§5.1.12.17).

marT (Top Margin)	This attribute specifies the top margin of the cell. The value specified in this attribute is the distance to offset from the top of the cell in EMU's.
	
	[*Example*: Consider the following example:
	
	`<a:tcPr marT="45720" anchor="ctr">`
	`…`
	`</a:tcPr>`
	
	In this example, we have specified a value for the margin on the top of the cell. *end example*]
	
	The possible values for this attribute are defined by the ST_Coordinate32 simple type (§5.1.12.17).

vert (Text Direction)	Defines the text direction within the cell.
	
	[*Example*: Consider the following example:
	`<a:tcPr vert="vert270">`
	`…`
	`</a:tcPr>`
	
	In this example, we have rotated the layout of the text 270 degrees so that it starts at the bottom of the cell and goes upward toward the top of the cell. *end example*]
	
	The possible values for this attribute are defined by the ST_TextVerticalType simple type (§5.1.12.83).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TableCellProperties">
   <sequence>
      <element name="lnL" type="CT_LineProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lnR" type="CT_LineProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lnT" type="CT_LineProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lnB" type="CT_LineProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lnTlToBr" type="CT_LineProperties" minOccurs="0" maxOccurs="1"/>
      <element name="lnBlToTr" type="CT_LineProperties" minOccurs="0" maxOccurs="1"/>
      <element name="cell3D" type="CT_Cell3D" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_FillProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="marL" type="ST_Coordinate32" use="optional" default="91440"/>
   <attribute name="marR" type="ST_Coordinate32" use="optional" default="91440"/>
   <attribute name="marT" type="ST_Coordinate32" use="optional" default="45720"/>
   <attribute name="marB" type="ST_Coordinate32" use="optional" default="45720"/>
   <attribute name="vert" type="ST_TextVerticalType" use="optional" default="horz"/>
   <attribute name="anchor" type="ST_TextAnchoringType" use="optional" default="t"/>
   <attribute name="anchorCtr" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="horzOverflow" type="ST_TextHorzOverflowType" use="optional" default="clip"/>
</complexType>
```

5.1.6.16 tr (Table Row)

This element defines a row in a table. A row as defined in a table is simply a listing of table cells (§5.1.6.14). There will be a table row element defined for every row in the table.

[*Example*: Consider the following example of a tr within DrawingML:

```
<a:tr h="774700">
  <a:tc>
    <a:txBody>
      <a:bodyPr/>
      <a:lstStyle/>
      <a:p>
        <a:endParaRPr lang="en-US" dirty="0"/>
      </a:p>
    </a:txBody>
    <a:tcPr/>
  </a:tc>
…
</a:tr>
```

In this example, we see a table row defined with an example table cell (§5.1.6.14) defined within it. The height of the row has been specified and in real use, there will be a table cell defined in this row for each grid column (§5.1.6.2) defined in the table. *end example*]

Parent Elements

Parent Elements
tbl (§5.1.6.11)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
tc (Table Cell)	§5.1.6.14

Attributes	Description
h (Height)	Defines the height of the row in the table. The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TableRow">
   <sequence>
      <element name="tc" type="CT_TableCell" minOccurs="0" maxOccurs="unbounded"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="h" type="ST_Coordinate" use="required"/>
</complexType>
```

5.1.7 3D

The 3D portion of the DrawingML framework allows for the describing of a 3D scene to be placed within a document. This 3D scene can be described using text and shape objects along with various lighting, material and camera settings.

5.1.7.1 anchor (Anchor Point)

This element specifies a point in 3D space. This point is the point in space that anchors the backdrop plane. Please see the example in the backdrop (§5.1.7.2) definition for an in depth explanation of this element.

Parent Elements
backdrop (§5.1.7.2)

Attributes	Description
x (X-Coordinate in 3D)	X-Coordinate in 3D space. The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).
y (Y-Coordinate in	Y-Coordinate in 3D space.

Attributes	Description
3D)	The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).
z (Z-Coordinate in 3D)	Z-Coordinate in 3D space. The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).

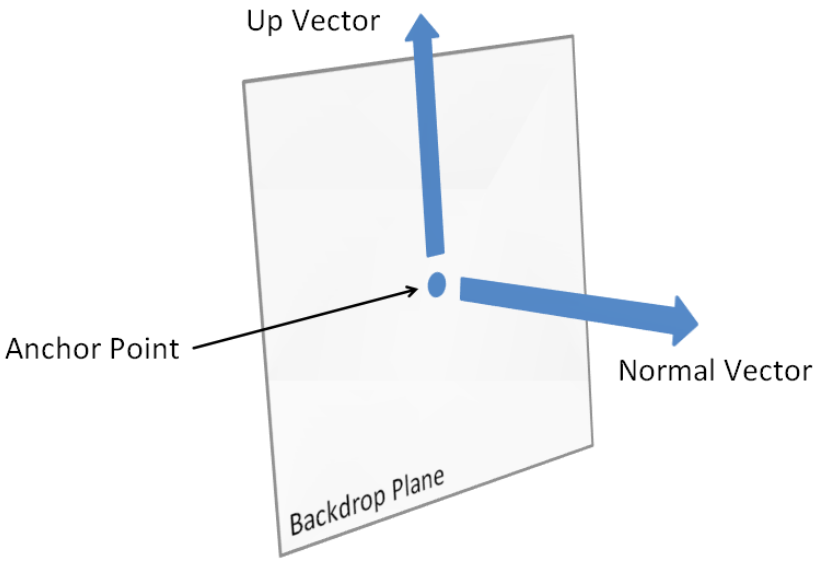
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Point3D">
   <attribute name="x" type="ST_Coordinate" use="required"/>
   <attribute name="y" type="ST_Coordinate" use="required"/>
   <attribute name="z" type="ST_Coordinate" use="required"/>
</complexType>
```

5.1.7.2 backdrop (Backdrop Plane)

This element defines a plane in which effects, such as glow and shadow, are applied in relation to the shape they are being applied to. The points and vectors contained within the backdrop define a plane in 3D space.

[*Example*: Consider the following image as an explanation of the backdrop plane definition:



In this image we see a plane being defined by an anchor point, the vector normal to the face of the plane and a vector pointing up in relation to the plane. *end example*]

Parent Elements
scene3d (§5.1.4.1.26); scene3d (§5.9.5.5)

Child Elements	Subclause
anchor (Anchor Point)	§5.1.7.1
extLst (Extension List)	§5.1.2.1.15
norm (Normal)	§5.1.7.10
up (Up Vector)	§5.1.7.13

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Backdrop">
   <sequence>
      <element name="anchor" type="CT_Point3D" minOccurs="1" maxOccurs="1"/>
      <element name="norm" type="CT_Vector3D" minOccurs="1" maxOccurs="1"/>
      <element name="up" type="CT_Vector3D" minOccurs="1" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.7.3 bevelB (Bottom Bevel)

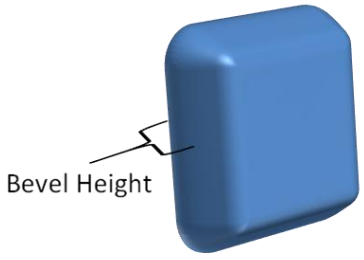
This element holds the properties associated with defining a bevel on the bottom or back face of a shape.

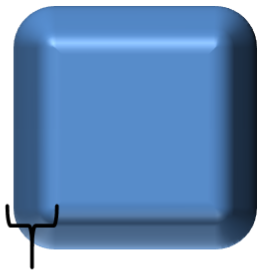
[*Example*: Consider the following example of an sp3d containing a bottom bevel.

```
<a:sp3d>
  <a:bevelB w="139700" h="127000" prst="coolSlant"/>
</a:sp3d>
```

In this example, we see a bottom bevel being defined with a preset bevel type along with a custom width and height. *end example*]

Parent Elements
sp3d (§5.1.7.12); sp3d (§5.9.5.6)

Attributes	Description
h (Height)	Specifies the height of the bevel, or how far above the shape it is applied. [*Example*: Consider the following example bevel 

Attributes	Description
	In this example, we see the height of an example bevel on a shape. *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).
prst (Preset Bevel)	Specifies the preset bevel type which defines the look of the bevel. The possible values for this attribute are defined by the ST_BevelPresetType simple type (§5.1.12.9).
w (Width)	Specifies the width of the bevel, or how far into the shape it is applied. [*Example*: Consider the following example bevel  Bevel Width In this example, we see the width of an example bevel on a shape. *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Bevel">
   <attribute name="w" type="ST_PositiveCoordinate" use="optional" default="76200"/>
   <attribute name="h" type="ST_PositiveCoordinate" use="optional" default="76200"/>
   <attribute name="prst" type="ST_BevelPresetType" use="optional" default="circle"/>
</complexType>
```

5.1.7.4 bevelT (Top Bevel)

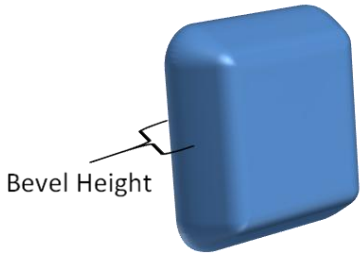
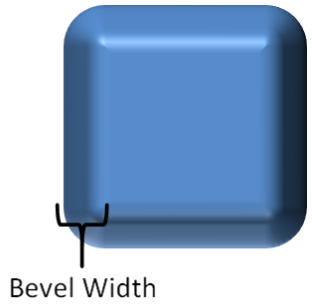
This element holds the properties associated with defining a bevel on the top or front face of a shape.

[*Example*: Consider the following example of an sp3d containing a top bevel.

```
<a:sp3d>
  <a:bevelT w="139700" h="127000" prst="coolSlant"/>
</a:sp3d>
```

In this example, we see a top bevel being defined with a preset bevel type along with a custom width and height. *end example*]

Parent Elements
sp3d (§5.1.7.12); sp3d (§5.9.5.6)

Attributes	Description
h (Height)	Specifies the height of the bevel, or how far above the shape it is applied. [*Example*: Consider the following example bevel  In this example, we see the height of an example bevel on a shape. *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).
prst (Preset Bevel)	Specifies the preset bevel type which defines the look of the bevel. The possible values for this attribute are defined by the ST_BevelPresetType simple type (§5.1.12.9).
w (Width)	Specifies the width of the bevel, or how far into the shape it is applied. [*Example*: Consider the following example bevel  In this example, we see the width of an example bevel on a shape. *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Bevel">
   <attribute name="w" type="ST_PositiveCoordinate" use="optional" default="76200"/>
   <attribute name="h" type="ST_PositiveCoordinate" use="optional" default="76200"/>
   <attribute name="prst" type="ST_BevelPresetType" use="optional" default="circle"/>
</complexType>
```

5.1.7.5 camera (Camera)

This element defines the placement and properties of the camera in the 3D scene. The camera position and properties modify the view of the scene.

[*Example*: Consider the following example of a camera in DrawingML:

```
<a:camera prst="orthographicFront">
  <a:rot lat="19902513" lon="17826689" rev="1362739"/>
</a:camera>
```

In this example, we see a preset camera being defined along with a rotation containing latitude, longitude, and revolution overrides provided that further rotate the camera around the scene. The effect of this camera can be seen on the following shape:

end example]

Parent Elements
scene3d (§5.1.4.1.26); scene3d (§5.9.5.5)

Child Elements	Subclause
rot (Rotation)	§5.1.7.11

Attributes	Description
fov (Field of View)	Provides an override for the default field of view for the camera. Different perspectives can be obtained by modifying this attribute. [*Example*: Consider the following example of a fov in DrawingML:

Attributes	Description
	<a:camera prst="perspectiveContrastingRightFacing" fov="6900000"> <a:rot lat="1200000" lon="18000000" rev="1200000"/> </a:camera> In this example, we see a fov being defined which modifies the default fov for the preset camera. *end example*] The possible values for this attribute are defined by the ST_FOVAngle simple type (§5.1.12.24).
prst (Preset Camera Type)	Defines the preset camera that is being used by the camera element. The preset camera defines a starting point for common preset rotations in space. [*Example*: Consider the following example of a prst in DrawingML: <a:camera prst="perspectiveContrastingRightFacing" fov="6900000"> <a:rot lat="1200000" lon="18000000" rev="1200000"/> </a:camera> In this example, we see a prst being defined as perspectiveContrastingRightFacing. *end example*] The possible values for this attribute are defined by the ST_PresetCameraType simple type (§5.1.12.47).
zoom (Zoom)	Defines the zoom factor of a given camera element. The zoom modifies the scene as a whole and zooms in or out accordingly. [*Example*: Consider the following example of a zoom in DrawingML: <a:camera prst="perspectiveContrastingRightFacing" fov="6900000" zoom="200000"> <a:rot lat="1200000" lon="18000000" rev="1200000"/> /a:camera> In this example, we see a zoom being used which will zoom the scene by 200%. *end example*] The possible values for this attribute are defined by the ST_PositivePercentage simple type (§5.1.12.46).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Camera">
   <sequence>
      <element name="rot" type="CT_SphereCoords" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="prst" type="ST_PresetCameraType" use="required"/>
   <attribute name="fov" type="ST_FOVAngle" use="optional"/>
   <attribute name="zoom" type="ST_PositivePercentage" use="optional" default="100000"/>
</complexType>
```

5.1.7.6 contourClr (Contour Color)

This element defines the color for the contour on a shape. The contour of a shape is a solid filled line which surrounds the outer edges of the shape.

[*Example*: Consider the following example of a contour defined on a shape which includes a contourClr. Lighting characteristics applied to the shape are ignored when it comes to the contour on the shape.

```
<a:sp3d contourW="101600" prstMaterial="plastic">
  <a:bevelT w="254000" h="254000"/>
  <a:bevelB w="254000" h="254000"/>
  <a:contourClr>
    <a:schemeClr val="bg1"/>
  </a:contourClr>
</a:sp3d>
```

In this example, we see a contour defined on a shape with a top and bottom bevel defined. In the image below, the contour is the white ring around the shape.

end example]

Parent Elements
sp3d (§5.1.7.12); sp3d (§5.9.5.6)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

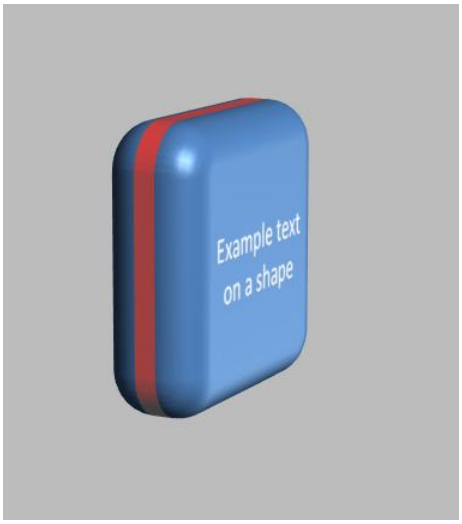
5.1.7.7 extrusionClr (Extrusion Color)

This element defines the color of the extrusion applied to a shape. The extrusion on a shape is an artificial height applied to the geometry.

[*Example*: Consider the following example of an extrusion which takes advantage of the extrusionClr. Lighting characteristics that are applied to the shape are also applied to the extrusion on the shape.

```
<a:sp3d extrusionH="139700" prstMaterial="plastic">
  <a:bevelT w="254000" h="254000"/>
  <a:bevelB w="254000" h="254000"/>
  <a:extrusionClr>
  <a:srgbClr val="FF0000"/>
  </a:extrusionClr>
</a:sp3d>
```

In this example, we see the extrusion color defined as red which can also be shown applied to the shape in the following image:



end example]

Parent Elements
sp3d (§5.1.7.12); sp3d (§5.9.5.6)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

5.1.7.8 flatTx (No text in 3D scene)

Keep text out of 3D scene entirely.

Parent Elements
bodyPr (§5.1.5.1.1); txPr (§5.9.5.12)

Attributes	Description
z (Z Coordinate)	Specifies the Z coordinate to be used in positioning the flat text within the 3D scene. The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FlatText">
   <attribute name="z" type="ST_Coordinate" use="optional" default="0"/>
</complexType>
```

5.1.7.9 lightRig (Light Rig)

This element defines the light rig associated with the table. The light rig comes into play when there is a 3D bevel applied to a cell. When 3D is used, the light rig defines the lighting properties associated with the scene.

Parent Elements
cell3D (§5.1.6.1); scene3d (§5.1.4.1.26); scene3d (§5.9.5.5)

Child Elements	Subclause
rot (Rotation)	§5.1.7.11

Attributes	Description
dir (Direction)	Defines the direction from which the light rig is oriented in relation to the scene. [*Example*: Consider the following example of dir being used in a light rig: `<a:lightRig rig="threePt" dir="t"/>` In this example, we define the direction to be `top`. *end example*] The possible values for this attribute are defined by the ST_LightRigDirection simple type (§5.1.12.29).
rig (Rig Preset)	Defines the preset type of light rig which is to be applied to the scene. [*Example*: Consider the following example of rig being used in a light rig: `<a:lightRig rig="threePt" dir="t"/>` In this example, we define the rig to be a `threePt` rig. *end example*] The possible values for this attribute are defined by the ST_LightRigType simple type (§5.1.12.30).

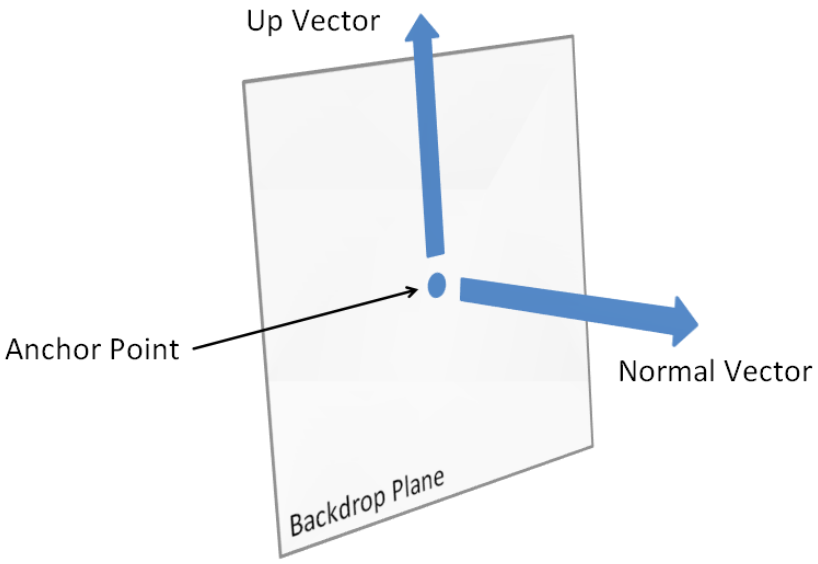
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LightRig">
   <sequence>
      <element name="rot" type="CT_SphereCoords" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="rig" type="ST_LightRigType" use="required"/>
   <attribute name="dir" type="ST_LightRigDirection" use="required"/>
</complexType>
```

5.1.7.10 norm (Normal)

This element defines a normal vector. To be more precise, this attribute defines a vector normal to the face of the backdrop plane.

[*Example*: Consider the following image as an example of what a normal vector is in relation to the backdrop plane:



end example]

Parent Elements
backdrop (§5.1.7.2)

Attributes	Description
dx (Distance along X-axis in 3D)	Distance along X-axis in 3D The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).
dy (Distance along Y-axis in 3D)	Distance along Y-axis in 3D

Attributes	Description
	The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).
dz (Distance along Z-axis in 3D)	Distance along Z-axis in 3D The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Vector3D">
   <attribute name="dx" type="ST_Coordinate" use="required"/>
   <attribute name="dy" type="ST_Coordinate" use="required"/>
   <attribute name="dz" type="ST_Coordinate" use="required"/>
</complexType>
```

5.1.7.11 rot (Rotation)

This element defines a rotation in 3D space. A rotation in DrawingML is defined through the use of a latitude coordinate, a longitude coordinate, and a revolution about the axis as the latitude and longitude coordinates.

[*Example*: Consider the following example of a rotation defined by the rot elements being used in a lightRig in DrawingML:

```
<a:lightRig rig="twoPt" dir="t">
  <a:rot lat="0" lon="0" rev="6000000"/>
</a:lightRig>
```

In this example, we have only a revolution applied to the light rig rich rotates it around it's center axis. *end example*]

Parent Elements
camera (§5.1.7.5); lightRig (§5.1.7.9)

Attributes	Description
lat (Latitude)	Defines the latitude value of the rotation. [*Example*: Consider the following example of a rot in DrawingML: `<a:rot lat="0" lon="0" rev="6000000"/>` In this example, we set the lat to be equal to `0`. *end example*] The possible values for this attribute are defined by the ST_PositiveFixedAngle simple type (§5.1.12.44).
lon (Longitude)	Defines the longitude value of the rotation.

Attributes	Description
	[*Example*: Consider the following example of a rot in DrawingML: `<a:rot lat="0" lon="0" rev="6000000"/>` In this example, we set the lon to be equal to `0`. *end example*] The possible values for this attribute are defined by the ST_PositiveFixedAngle simple type (§5.1.12.44).
rev (Revolution)	This attributes defines the revolution around the central axis in the rotation. [*Example*: Consider the following example of a rot in DrawingML: `<a:rot lat="0" lon="0" rev="6000000"/>` In this example, we set the rev to be equal to `6000000`. *end example*] The possible values for this attribute are defined by the ST_PositiveFixedAngle simple type (§5.1.12.44).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SphereCoords">
   <attribute name="lat" type="ST_PositiveFixedAngle" use="required"/>
   <attribute name="lon" type="ST_PositiveFixedAngle" use="required"/>
   <attribute name="rev" type="ST_PositiveFixedAngle" use="required"/>
</complexType>
```

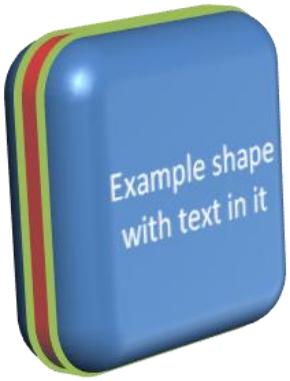
5.1.7.12 sp3d (Apply 3D shape properties)

This element defines the 3D properties associated with a particular shape in DrawingML. The 3D properties which can be applied to a shape are top and bottom bevels, a contour and an extrusion.

[*Example*: Consider the following example of an sp3d in DrawingML:

```
<a:sp3d extrusionH="165100" contourW="50800" prstMaterial="plastic">
  <a:bevelT w="254000" h="254000"/>
  <a:bevelB w="254000" h="254000"/>
  <a:extrusionClr>
    <a:srgbClr val="FF0000"/>
  </a:extrusionClr>
  <a:contourClr>
    <a:schemeClr val="accent3"/>
  </a:contourClr>
</a:sp3d>
```

In this example, we see an sp3d defined which contains information defining both a top and bottom bevel, along with an extrusion and contour on the shape. The following image illustrates a shape with the applied sp3d:



end example]

Parent Elements
bodyPr (§5.1.5.1.1); effectStyle (§5.1.4.1.11); spPr (§5.6.2.29); spPr (§5.7.2.198); spPr (§5.8.2.23); spPr (§5.1.2.1.35); spPr (§4.4.1.41); spPr (§5.9.3.7); spPr (§5.2.2.6); txPr (§5.9.5.12)

Child Elements	Subclause
bevelB (Bottom Bevel)	§5.1.7.3
bevelT (Top Bevel)	§5.1.7.4
contourClr (Contour Color)	§5.1.7.6
extLst (Extension List)	§5.1.2.1.15
extrusionClr (Extrusion Color)	§5.1.7.7

Attributes	Description
contourW (Contour Width)	Defines the width of the contour on the shape. [*Example*: Consider the following example of a contourW in use within the sp3d element: `<a:sp3d extrusionH="165100" contourW="50800" prstMaterial="plastic"><a:bevelT w="254000" h="254000"/><a:bevelB w="254000" h="254000"/><a:extrusionClr><a:srgbClr val="FF0000"/></a:extrusionClr>`

Attributes	Description
	<a:contourClr> <a:schemeClr val="accent3"/> </a:contourClr> >/a:sp3d> In this example, we see a countourW defined as 50800. *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).
extrusionH (Extrusion Height)	Defines the height of the extrusion applied to the shape. [*Example*: Consider the following example of an extrusionH in use within the sp3d element: <a:sp3d extrusionH="165100" contourW="50800" prstMaterial="plastic"> < <a:bevelT w="254000" h="254000"/> <a:bevelB w="254000" h="254000"/> <a:extrusionClr> <a:srgbClr val="FF0000"/> </a:extrusionClr> <a:contourClr> <a:schemeClr val="accent3"/> </a:contourClr> </a:sp3d> In this example, we see a extrusionH defined as 165100. *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).
prstMaterial (Preset Material Type)	Defines the preset material which is combined with the lighting properties to give the final look and feel of a shape. [*Example*: Consider the following example of a prstMaterial in use within the sp3d element: <a:sp3d extrusionH="165100" contourW="50800" prstMaterial="plastic"> <a:bevelT w="254000" h="254000"/> <a:bevelB w="254000" h="254000"/> <a:extrusionClr> <a:srgbClr val="FF0000"/> </a:extrusionClr> <a:contourClr> <a:schemeClr val="accent3"/> </a:contourClr>

Attributes	Description
	</a:sp3d> In this example, we see a prstMaterial defined as `plastic`. *end example*] The possible values for this attribute are defined by the ST_PresetMaterialType simple type (§5.1.12.50).
z (Shape Depth)	Defines the z coordinate for the 3D shape. The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).

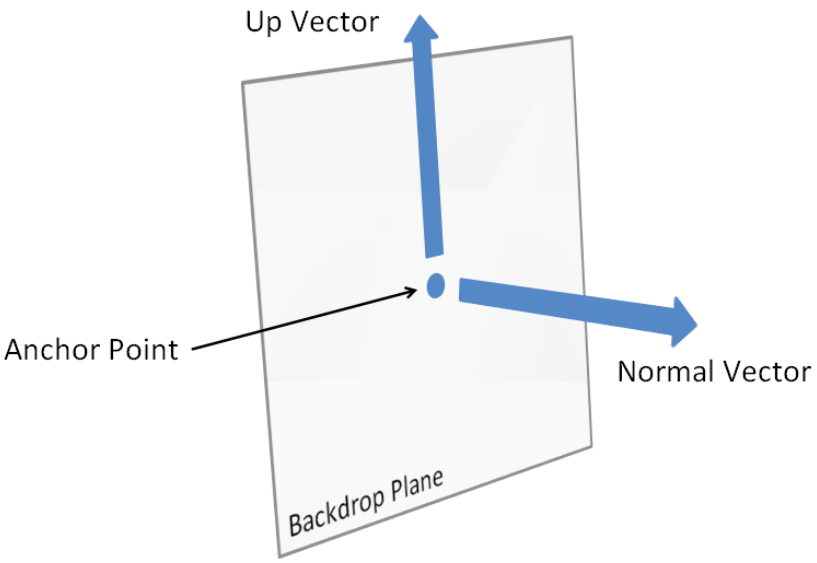
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Shape3D">
   <sequence>
      <element name="bevelT" type="CT_Bevel" minOccurs="0" maxOccurs="1"/>
      <element name="bevelB" type="CT_Bevel" minOccurs="0" maxOccurs="1"/>
      <element name="extrusionClr" type="CT_Color" minOccurs="0" maxOccurs="1"/>
      <element name="contourClr" type="CT_Color" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="z" type="ST_Coordinate" use="optional" default="0"/>
   <attribute name="extrusionH" type="ST_PositiveCoordinate" use="optional" default="0"/>
   <attribute name="contourW" type="ST_PositiveCoordinate" use="optional" default="0"/>
   <attribute name="prstMaterial" type="ST_PresetMaterialType" use="optional" default="warmMatte"/>
</complexType>
```

5.1.7.13 up (Up Vector)

This element defines a vector representing up. To be more precise, this attribute defines a vector representing up in relation to the face of the backdrop plane.

[*Example*: Consider the following image as an example of what an up vector is in relation to the backdrop plane:



end example]

Parent Elements
backdrop (§5.1.7.2)

Attributes	Description
dx (Distance along X-axis in 3D)	Distance along X-axis in 3D The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).
dy (Distance along Y-axis in 3D)	Distance along Y-axis in 3D The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).
dz (Distance along Z-axis in 3D)	Distance along Z-axis in 3D The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Vector3D">
   <attribute name="dx" type="ST_Coordinate" use="required"/>
   <attribute name="dy" type="ST_Coordinate" use="required"/>
   <attribute name="dz" type="ST_Coordinate" use="required"/>
</complexType>
```

5.1.8 Shared Style Sheet

The shared style sheet aspects contained within DrawingML are responsible for containing formatting options and styles which can be used by applications to define a certain look or feel to documents. The shared style sheet can be used by any document type (for example, a presentation) to pull visual information from which formats the document in a certain way, or theme. The shared style sheet contains information that is not document type specific.

5.1.8.1 clrMap (Color Map)

This element specifics the color mapping layer which allows a user to define colors for background and text. This allows for swapping out of light/dark colors for backgrounds and the text on top of the background in order to maintain readability of the text On a deeper level, this specifies exactly which colors the first 12 values refer to in the color scheme.

[*Example*: Consider the following example of a color map in use:

```
<clrMap bg1="lt1" tx1="dk1" bg2="lt2" tx2="dk2" accent1="accent1"
  accent2="accent2" accent3="accent3" accent4="accent4" accent5="accent5"
  accent6="accent6" hlink="hlink" folHlink="folHlink"/>
```

In this example, we see that bg1 is mapped to lt1, tx1 is mapped to dk1, and so on. *end example*]

Parent Elements
extraClrScheme (§5.1.8.4)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15

Attributes	Description
accent1 (Accent 1)	Specifies a color defined which is associated as the accent 1 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
accent2 (Accent 2)	Specifies a color defined which is associated as the accent 2 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
accent3 (Accent 3)	Specifies a color defined which is associated as the accent 3 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
accent4 (Accent 4)	Specifies a color defined which is associated as the accent 4 color.

Attributes	Description
	The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
accent5 (Accent 5)	Specifies a color defined which is associated as the accent 5 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
accent6 (Accent 6)	Specifies a color defined which is associated as the accent 6 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
bg1 (Background 1)	A color defined which is associated as the first background color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
bg2 (Background 2)	Specifies a color defined which is associated as the second background color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
folHlink (Followed Hyperlink)	Specifies a color defined which is associated as the color for a followed hyperlink. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
hlink (Hyperlink)	Specifies a color defined which is associated as the color for a hyperlink. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
tx1 (Text 1)	Specifies a color defined which is associated as the first text color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
tx2 (Text 2)	Specifies a color defined which is associated as the second text color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ColorMapping">
   <sequence>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="bg1" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="tx1" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="bg2" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="tx2" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent1" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent2" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent3" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent4" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent5" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent6" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="hlink" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="folHlink" type="ST_ColorSchemeIndex" use="required"/>
</complexType>
```

5.1.8.2 clrScheme (Color Scheme)

This element defines a set of colors which are referred to as a color scheme. The color scheme is responsible for defining a list of twelve colors. The twelve colors consist of six accent colors, two dark colors, two light colors and a color for each of a hyperlink and followed hyperlink.

[*Example*: Consider the following example of a color scheme defined in DrawingML:

```
<clrScheme name="sample">
  <dk1>
    <sysClr val="windowText"/>
  </dk1>
  <lt1>
    <sysClr val="window"/>
  </lt1>
  <dk2>
    <srgbClr val="04617B"/>
  </dk2>
  <lt2>
    <srgbClr val="DBF5F9"/>
  </lt2>
  <accent1>
    <srgbClr val="0F6FC6"/>
  </accent1>
  <accent2>
    <srgbClr val="009DD9"/>
  </accent2>
```

```
    <accent3>
      <srgbClr val="0BD0D9"/>
    </accent3>
    <accent4>
      <srgbClr val="10CF9B"/>
    </accent4>
    <accent5>
      <srgbClr val="7CCA62"/>
    </accent5>
    <accent6>
      <srgbClr val="A5C249"/>
    </accent6>
    <hlink>
      <srgbClr val="FF9800"/>
    </hlink>
    <folHlink>
      <srgbClr val="F45511"/>
    </folHlink>
  </clrScheme>
```

In this example, are defined the 12 theme colors in the sample color scheme. *end example*]

Parent Elements
extraClrScheme (§5.1.8.4); themeElements (§5.1.8.10); themeOverride (§5.1.8.12)

Child Elements	Subclause
accent1 (Accent 1)	§5.1.4.1.1
accent2 (Accent 2)	§5.1.4.1.2
accent3 (Accent 3)	§5.1.4.1.3
accent4 (Accent 4)	§5.1.4.1.4
accent5 (Accent 5)	§5.1.4.1.5
accent6 (Accent 6)	§5.1.4.1.6
dk1 (Dark 1)	§5.1.4.1.9
dk2 (Dark 2)	§5.1.4.1.10
extLst (Extension List)	§5.1.2.1.15
folHlink (Followed Hyperlink)	§5.1.4.1.15
hlink (Hyperlink)	§5.1.4.1.19
lt1 (Light 1)	§5.1.4.1.22
lt2 (Light 2)	§5.1.4.1.23

Attributes	Description
name (Name)	The common name for this color scheme. This name can show up in the user interface in a list of color schemes. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ColorScheme">
   <sequence>
      <element name="dk1" type="CT_Color" minOccurs="1" maxOccurs="1"/>
      <element name="lt1" type="CT_Color" minOccurs="1" maxOccurs="1"/>
      <element name="dk2" type="CT_Color" minOccurs="1" maxOccurs="1"/>
      <element name="lt2" type="CT_Color" minOccurs="1" maxOccurs="1"/>
      <element name="accent1" type="CT_Color" minOccurs="1" maxOccurs="1"/>
      <element name="accent2" type="CT_Color" minOccurs="1" maxOccurs="1"/>
      <element name="accent3" type="CT_Color" minOccurs="1" maxOccurs="1"/>
      <element name="accent4" type="CT_Color" minOccurs="1" maxOccurs="1"/>
      <element name="accent5" type="CT_Color" minOccurs="1" maxOccurs="1"/>
      <element name="accent6" type="CT_Color" minOccurs="1" maxOccurs="1"/>
      <element name="hlink" type="CT_Color" minOccurs="1" maxOccurs="1"/>
      <element name="folHlink" type="CT_Color" minOccurs="1" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="name" type="xsd:string" use="required"/>
</complexType>
```

5.1.8.3 custClrLst (Custom Color List)

This element allows for a custom color palette to be created and which shows up alongside other color schemes. This can be very useful, for example, when someone would like to maintain a corporate color palette.

Parent Elements
theme (§5.1.8.9)

Child Elements	Subclause
custClr (Custom color)	§5.1.4.1.8

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustomColorList">
   <sequence>
      <element name="custClr" type="CT_CustomColor" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.1.8.4 extraClrScheme (Extra Color Scheme)

This element defines an auxiliary color scheme, which includes both a color scheme and color mapping. This is mainly used for backward compatibility concerns and roundtrips information required by earlier versions.

[*Example*: Consider the following example of an extra color scheme in use in DrawingML:

```
<extraClrScheme>
  <clrScheme name="extraColorSchemeSample">
    <dk1>
      <sysClr val="windowText"/>
    </dk1>
    <lt1>
      <sysClr val="window"/>
    </lt1>
    <dk2>
      <srgbClr val="04617B"/>
    </dk2>
    <lt2>
      <srgbClr val="DBF5F9"/>
    </lt2>
    <accent1>
      <srgbClr val="0F6FC6"/>
    </accent1>
    <accent2>
      <srgbClr val="009DD9"/>
    </accent2>
    <accent3>
      <srgbClr val="0BD0D9"/>
    </accent3>
    <accent4>
      <srgbClr val="10CF9B"/>
    </accent4>
    <accent5>
      <srgbClr val="7CCA62"/>
    </accent5>
    <accent6>
      <srgbClr val="A5C249"/>
    </accent6>
    <hlink>
      <srgbClr val="FF9800"/>
    </hlink>
    <folHlink>
      <srgbClr val="F45511"/>
    </folHlink>
```

```
    </clrScheme>
    <clrMap bg1="lt1" tx1="dk1" bg2="lt2" tx2="dk2" accent1="accent1"
    accent2="accent2" accent3="accent3" accent4="accent4" accent5="accent5"
    accent6="accent6" hlink="hlink" folHlink="folHlink"/>
  </extraClrScheme>
```

In this example, the extra color scheme contains a color scheme and a color map for that color scheme. *end example*]

Parent Elements
extraClrSchemeLst (§5.1.8.5)

Child Elements	Subclause
clrMap (Color Map)	§5.1.8.1
clrScheme (Color Scheme)	§5.1.8.2

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ColorSchemeAndMapping">
   <sequence>
      <element name="clrScheme" type="CT_ColorScheme" minOccurs="1" maxOccurs="1"/>
      <element name="clrMap" type="CT_ColorMapping" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.8.5 extraClrSchemeLst (Extra Color Scheme List)

This element is a container for the list of extra color schemes present in a document.

Parent Elements
theme (§5.1.8.9)

Child Elements	Subclause
extraClrScheme (Extra Color Scheme)	§5.1.8.4

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ColorSchemeList">
   <sequence>
      <element name="extraClrScheme" type="CT_ColorSchemeAndMapping" minOccurs="0"
        maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.1.8.6 masterClrMapping (Master Color Mapping)

This element is a part of a choice for which color mapping is used within the document. There is also defined an overrideClrMapping (§1.1.8) element which, when specified, the override is used rather than the color mapping defined in the master. If this element is specified, then we specifically use the color mapping defined in the master.

Parent Elements
clrMapOvr (§4.4.1.7)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EmptyElement"/>
```

5.1.8.7 objectDefaults (Object Defaults)

This element allows for the definition of default shape, line, and textbox formatting properties. An application can use this information to format a shape (or text) initially on insertion into a document.

Parent Elements
theme (§5.1.8.9)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
lnDef (Line Default)	§5.1.4.1.20
spDef (Shape Default)	§5.1.4.1.27
txDef (Text Default)	§5.1.4.1.28

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ObjectStyleDefaults">
   <sequence>
      <element name="spDef" type="CT_DefaultShapeDefinition" minOccurs="0" maxOccurs="1"/>
      <element name="lnDef" type="CT_DefaultShapeDefinition" minOccurs="0" maxOccurs="1"/>
      <element name="txDef" type="CT_DefaultShapeDefinition" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.8.8 overrideClrMapping (Override Color Mapping)

This element provides an override for the color mapping in a document. When defined, this color mapping is used in place of the already defined color mapping, or master color mapping. This color mapping is defined in the same manner as the other mappings within this document.

[*Example*: Consider the following example of an override color mapping in DrawingML:

```
<overrideClrMapping bg1="lt1" tx1="dk1" bg2="lt2" tx2="dk2" accent1="accent1"
  accent2="accent2" accent3="accent3" accent4="accent4" accent5="accent5"
  accent6="accent6" hlink="hlink" folHlink="folHlink"/>
```

end example]

Parent Elements
clrMapOvr (§4.4.1.7)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15

Attributes	Description
accent1 (Accent 1)	Specifies a color defined which is associated as the accent 1 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
accent2 (Accent 2)	Specifies a color defined which is associated as the accent 2 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
accent3 (Accent 3)	Specifies a color defined which is associated as the accent 3 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
accent4 (Accent 4)	Specifies a color defined which is associated as the accent 4 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
accent5 (Accent 5)	Specifies a color defined which is associated as the accent 5 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
accent6 (Accent 6)	Specifies a color defined which is associated as the accent 6 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
bg1 (Background 1)	A color defined which is associated as the first background color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).

Attributes	Description
bg2 (Background 2)	Specifies a color defined which is associated as the second background color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
folHlink (Followed Hyperlink)	Specifies a color defined which is associated as the color for a followed hyperlink. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
hlink (Hyperlink)	Specifies a color defined which is associated as the color for a hyperlink. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
tx1 (Text 1)	Specifies a color defined which is associated as the first text color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
tx2 (Text 2)	Specifies a color defined which is associated as the second text color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).

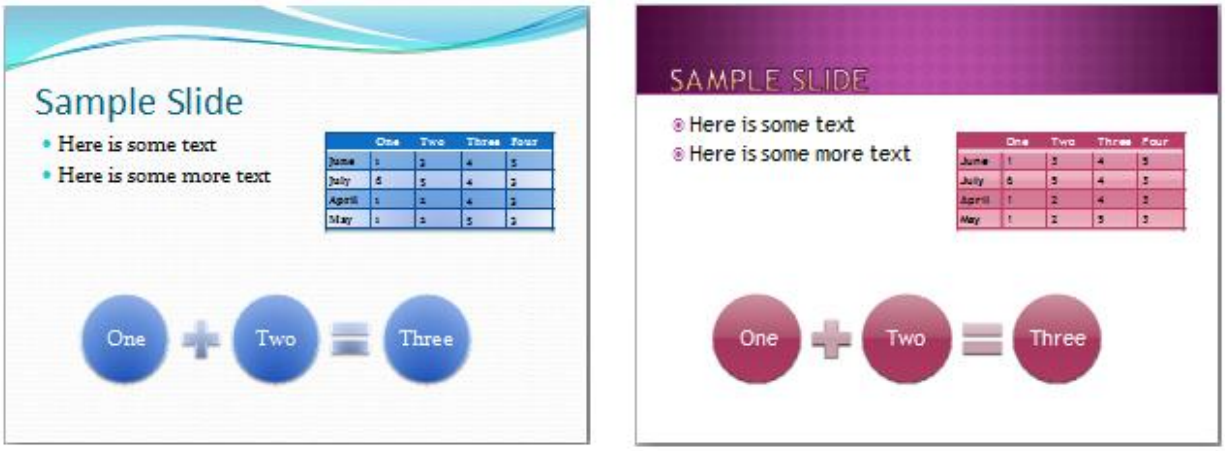
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ColorMapping">
   <sequence>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="bg1" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="tx1" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="bg2" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="tx2" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent1" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent2" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent3" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent4" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent5" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent6" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="hlink" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="folHlink" type="ST_ColorSchemeIndex" use="required"/>
</complexType>
```

5.1.8.9 theme (Theme)

This element defines the root level complex type associated with a shared style sheet (or theme). This element holds all the different formatting options available to a document through a theme and defines the overall look and feel of the document when themed objects are used within the document.

[*Example*: Consider the following image as an example of different themes in use applied to a presentation:



In this example, we see how a theme can affect font, colors, backgrounds, fills, and effects for different objects in a presentation. *end example*]

Parent Elements
Root element of DrawingML Theme part

Child Elements	Subclause
custClrLst (Custom Color List)	§5.1.8.3
extLst (Extension List)	§5.1.2.1.15
extraClrSchemeLst (Extra Color Scheme List)	§5.1.8.5
objectDefaults (Object Defaults)	§5.1.8.7
themeElements (Theme Elements)	§5.1.8.10

Attributes	Description
name (Name)	Specifies the name given to the theme. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OfficeStyleSheet">
   <sequence>
      <element name="themeElements" type="CT_BaseStyles" minOccurs="1" maxOccurs="1"/>
      <element name="objectDefaults" type="CT_ObjectStyleDefaults" minOccurs="0" maxOccurs="1"/>
      <element name="extraClrSchemeLst" type="CT_ColorSchemeList" minOccurs="0" maxOccurs="1"/>
      <element name="custClrLst" type="CT_CustomColorList" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="name" type="xsd:string" use="optional" default=""/>
</complexType>
```

5.1.8.10 themeElements (Theme Elements)

This element defines the theme formatting options for the theme and is the workhorse of the theme. This is where the bulk of the shared theme information is contained and used by a document. This element contains the color scheme, font scheme, and format scheme elements which define the different formatting aspects of what a theme defines.

[*Example*: Consider the following example of a theme elements defined in DrawingML:

```
<themeElements>
  <clrScheme name="sample">
…
  </clrScheme>
  <fontScheme name="sample">
…
  </fontScheme>
  <fmtScheme name="sample">
    <fillStyleLst>
…
    </fillStyleLst>
    <lnStyleLst>
…
    </lnStyleLst>
    <effectStyleLst>
…
    </effectStyleLst>
    <bgFillStyleLst>
…
    </bgFillStyleLst>
  </fmtScheme>
</themeElements>
```

In this example, we see the basic structure of how a theme elements is defined and have left out the true guts of each individual piece to save room. Each part (color scheme, font scheme, format scheme) is defined elsewhere within DrawingML. *end example*]

Parent Elements
theme (§5.1.8.9)

Child Elements	Subclause
clrScheme (Color Scheme)	§5.1.8.2
extLst (Extension List)	§5.1.2.1.15
fmtScheme (Format Scheme)	§5.1.4.1.14

Child Elements	Subclause
fontScheme (Font Scheme)	§5.1.4.1.18

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BaseStyles">
   <sequence>
      <element name="clrScheme" type="CT_ColorScheme" minOccurs="1" maxOccurs="1"/>
      <element name="fontScheme" type="CT_FontScheme" minOccurs="1" maxOccurs="1"/>
      <element name="fmtScheme" type="CT_StyleMatrix" minOccurs="1" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.8.11 themeManager (Theme Manager)

The starting part for a theme file.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EmptyElement"/>
```

5.1.8.12 themeOverride (Theme Override)

This element allows for an override which changes just the colors, fonts, or effects of a single object, like a table for example. Currently it is used only to control overrides on the non-top-level masters within a presentation.

Parent Elements
Root element of DrawingML Theme Override part

Child Elements	Subclause
clrScheme (Color Scheme)	§5.1.8.2
fmtScheme (Format Scheme)	§5.1.4.1.14
fontScheme (Font Scheme)	§5.1.4.1.18

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BaseStylesOverride">
   <sequence>
      <element name="clrScheme" type="CT_ColorScheme" minOccurs="0" maxOccurs="1"/>
      <element name="fontScheme" type="CT_FontScheme" minOccurs="0" maxOccurs="1"/>
      <element name="fmtScheme" type="CT_StyleMatrix" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.9 Coordinate Systems and Transformations

The following elements are used to reflect dimensions, scaling, location, rotation, and flip information on groups and individual shapes respectively.

5.1.9.1 chExt (Child Extents)

This element specifies the size dimensions of the child extents rectangle and is used for calculations of grouping, scaling, and rotation behavior of shapes placed within a group.

Parent Elements
xfrm (§5.1.9.5)

Attributes	Description
cx (Extent Length)	Specifies the length of the extents rectangle in EMUs. This rectangle shall dictate the size of the object as displayed (the result of any scaling to the original object). [*Example*: Consider a DrawingML object specified as follows: `<… cx="1828800" cy="200000"/>` The cx attributes specifies that this object has a height of `1828800` EMUs (English Metric Units). *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).
cy (Extent Width)	Specifies the width of the extents rectangle in EMUs. This rectangle shall dictate the size of the object as displayed (the result of any scaling to the original object). [*Example*: Consider a DrawingML object specified as follows: `< … cx="1828800" cy="200000"/>` The cy attribute specifies that this object has a width of `200000` EMUs (English Metric Units). *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PositiveSize2D">
   <attribute name="cx" type="ST_PositiveCoordinate" use="required"/>
   <attribute name="cy" type="ST_PositiveCoordinate" use="required"/>
</complexType>
```

5.1.9.2 chOff (Child Offset)

This element specifies the location of the child extents rectangle and is used for calculations of grouping, scaling, and rotation behavior of shapes placed within a group.

Parent Elements

Parent Elements
xfrm (§5.1.9.5)

Attributes	Description
x (X-Axis Coordinate)	Specifies a coordinate on the x-axis. The origin point for this coordinate shall be specified by the parent XML element. [*Example*: Consider the following point on a basic wrapping polygon for a DrawingML object: `<wp:… x="0" y="100" />` The x attribute defines an x-coordinate of `0`. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).
y (Y-Axis Coordinate)	Specifies a coordinate on the x-axis. The origin point for this coordinate shall be specified by the parent XML element. [*Example*: Consider the following point on a basic wrapping polygon for a DrawingML object: `<wp:… x="0" y="100" />` The y attribute defines a y-coordinate of `100`. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Point2D">
   <attribute name="x" type="ST_Coordinate" use="required"/>
   <attribute name="y" type="ST_Coordinate" use="required"/>
</complexType>
```

5.1.9.3 ext (Extents)

This element specifies the size of the bounding box enclosing the referenced object.

Parent Elements
xfrm (§4.4.1.49); xfrm (§5.1.9.6); xfrm (§5.6.2.35); xfrm (§5.8.2.28); xfrm (§5.1.9.5)

Attributes	Description
cx (Extent Length)	Specifies the length of the extents rectangle in EMUs. This rectangle shall dictate the size

Attributes	Description
	of the object as displayed (the result of any scaling to the original object). [*Example*: Consider a DrawingML object specified as follows: `<… cx="1828800" cy="200000"/>` The cx attributes specifies that this object has a height of 1828800 EMUs (English Metric Units). *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).
cy (Extent Width)	Specifies the width of the extents rectangle in EMUs. This rectangle shall dictate the size of the object as displayed (the result of any scaling to the original object). [*Example*: Consider a DrawingML object specified as follows: `< … cx="1828800" cy="200000"/>` The cy attribute specifies that this object has a width of 200000 EMUs (English Metric Units). *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PositiveSize2D">
   <attribute name="cx" type="ST_PositiveCoordinate" use="required"/>
   <attribute name="cy" type="ST_PositiveCoordinate" use="required"/>
</complexType>
```

5.1.9.4 off (Offset)

This element specifies the location of the bounding box of an object. Effects on an object are not included in this bounding box.

Parent Elements
xfrm (§4.4.1.49); xfrm (§5.1.9.6); xfrm (§5.6.2.35); xfrm (§5.8.2.28); xfrm (§5.1.9.5)

Attributes	Description
x (X-Axis Coordinate)	Specifies a coordinate on the x-axis. The origin point for this coordinate shall be specified by the parent XML element. [*Example*: Consider the following point on a basic wrapping polygon for a DrawingML object:

Attributes	Description
	<wp:… x="0" y="100" /> The x attribute defines an x-coordinate of 0. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).
y (Y-Axis Coordinate)	Specifies a coordinate on the x-axis. The origin point for this coordinate shall be specified by the parent XML element. [*Example*: Consider the following point on a basic wrapping polygon for a DrawingML object: <wp:… x="0" y="100" /> The y attribute defines a y-coordinate of 100. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Point2D">
   <attribute name="x" type="ST_Coordinate" use="required"/>
   <attribute name="y" type="ST_Coordinate" use="required"/>
</complexType>
```

5.1.9.5 xfrm (2D Transform for Grouped Objects)

This element is nearly identical to the representation of 2-D transforms for ordinary shapes (§5.1.9.6). The only addition is a member to represent the Child offset and the Child extents.

Parent Elements
grpSpPr (§5.8.2.14); grpSpPr (§4.4.1.20); grpSpPr (§5.1.2.1.22); grpSpPr (§5.6.2.17)

Child Elements	Subclause
chExt (Child Extents)	§5.1.9.1
chOff (Child Offset)	§5.1.9.2
ext (Extents)	§5.1.9.3
off (Offset)	§5.1.9.4

Attributes	Description

Attributes	Description
flipH (Horizontal Flip)	Horizontal flip. When true, this attribute defines that the group will be flipped horizontally about the center of its bounding box. [*Example* -- The following illustrates the effect of a horizontal flip.  end example] The possible values for this attribute are defined by the XML Schema boolean datatype.
flipV (Vertical Flip)	Vertical flip. When true, this attribute defines that the group will be flipped vertically about the center of its bounding box. [*Example* -- The following illustrates the effect of a vertical flip.  *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
rot (Rotation)	Rotation. Specifies the clockwise rotation of a group in 1/64000 of a degree. The possible values for this attribute are defined by the ST_Angle simple type (§5.1.12.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GroupTransform2D">
   <sequence>
      <element name="off" type="CT_Point2D" minOccurs="0" maxOccurs="1"/>
      <element name="ext" type="CT_PositiveSize2D" minOccurs="0" maxOccurs="1"/>
      <element name="chOff" type="CT_Point2D" minOccurs="0" maxOccurs="1"/>
      <element name="chExt" type="CT_PositiveSize2D" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="rot" type="ST_Angle" use="optional" default="0"/>
   <attribute name="flipH" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="flipV" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.1.9.6 xfrm (2D Transform for Individual Objects)

This element represents 2-D transforms for ordinary shapes.

Parent Elements
graphicFrame (§5.1.2.1.18); spPr (§5.6.2.29); spPr (§5.7.2.198); spPr (§5.8.2.23); spPr (§5.1.2.1.35); spPr (§4.4.1.41); spPr (§5.9.3.7); spPr (§5.2.2.6); txSp (§5.1.2.1.41)

Child Elements	Subclause
ext (Extents)	§5.1.9.3
off (Offset)	§5.1.9.4

Attributes	Description
flipH (Horizontal Flip)	Specifies a horizontal flip. When true, this attribute defines that the shape will be flipped horizontally about the center of its bounding box. [*Example*: The following illustrates the effect of a horizontal flip.  *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
flipV (Vertical Flip)	Specifies a vertical flip. When true, this attribute defines that the group will be flipped vertically about the center of its bounding box. [*Example*: The following illustrates the effect of a vertical flip.  *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
rot (Rotation)	Specifies the rotation of the Graphic Frame. The units for which this attribute is specified in reside within the simple type definition referenced below. The possible values for this attribute are defined by the ST_Angle simple type (§5.1.12.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Transform2D">
   <sequence>
      <element name="off" type="CT_Point2D" minOccurs="0" maxOccurs="1"/>
      <element name="ext" type="CT_PositiveSize2D" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="rot" type="ST_Angle" use="optional" default="0"/>
   <attribute name="flipH" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="flipV" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.1.10 Shape Fills, Effects, and Line Properties

This portion of the DrawingML framework describes effects defining the visual appearance of shapes and lines. Shapes may be filled in a variety of ways, with images, solid colors, gradients, or pattern fills. In addition, several visual effects may alter the appearance of a shape, and multiple effects may be combined together. Lines also may have special properties defining how they are rendered, included a dashed appearance or decorations at the line ends. This section documents the elements that define these properties and effects for shapes and lines.

5.1.10.1 alphaBiLevel (Alpha Bi-Level Effect)

This element represents an Alpha Bi-Level Effect.

Alpha (Opacity) values less than the threshold are changed to 0 (fully transparent) and alpha values greater than or equal to the threshold are changed to 100% (fully opaque).

Parent Elements
blip (§5.1.10.13); cont (§5.1.10.20); effectDag (§5.1.10.25)

Attributes	Description
thresh (Threshold)	Specifies the threshold value for the alpha bi-level effect. The possible values for this attribute are defined by the ST_PositiveFixedPercentage simple type (§5.1.12.45).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AlphaBiLevelEffect">
   <attribute name="thresh" type="ST_PositiveFixedPercentage" use="required"/>
</complexType>
```

5.1.10.2 alphaCeiling (Alpha Ceiling Effect)

This element represents an alpha ceiling effect.

Alpha (opacity) values greater than zero are changed to 100%. In other words, anything partially opaque becomes fully opaque.

Parent Elements
blip (§5.1.10.13); cont (§5.1.10.20); effectDag (§5.1.10.25)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AlphaCeilingEffect"/>
```

5.1.10.3 alphaFloor (Alpha Floor Effect)

This element represents an alpha floor effect.

Alpha (opacity) values less than 100% are changed to zero. In other words, anything partially transparent becomes fully transparent.

Parent Elements
blip (§5.1.10.13); cont (§5.1.10.20); effectDag (§5.1.10.25)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AlphaFloorEffect"/>
```

5.1.10.4 alphaInv (Alpha Inverse Effect)

This element represents an alpha inverse effect.

Alpha (opacity) values are inverted by subtracting from 100%.

Parent Elements
blip (§5.1.10.13); cont (§5.1.10.20); effectDag (§5.1.10.25)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AlphaInverseEffect">
   <sequence>
      <group ref="EG_ColorChoice" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.10.5 alphaMod (Alpha Modulate Effect)

This element represents an alpha modulate effect.

Effect alpha (opacity) values are multiplied by a fixed percentage. The effect container specifies an effect containing alpha values to modulate.

Parent Elements
blip (§5.1.10.13); cont (§5.1.10.20); effectDag (§5.1.10.25)

Child Elements	Subclause
cont (Effect Container)	§5.1.10.20

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AlphaModulateEffect">
   <sequence>
      <element name="cont" type="CT_EffectContainer" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.10.6 alphaModFix (Alpha Modulate Fixed Effect)

This element represents an alpha modulate fixed effect.

Effect alpha (opacity) values are multiplied by a fixed percentage.

Parent Elements
blip (§5.1.10.13); cont (§5.1.10.20); effectDag (§5.1.10.25)

Attributes	Description
amt (Amount)	Specifies the percentage amount to scale the alpha. The possible values for this attribute are defined by the ST_PositivePercentage simple type (§5.1.12.46).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AlphaModulateFixedEffect">
   <attribute name="amt" type="ST_PositivePercentage" use="optional" default="100000"/>
</complexType>
```

5.1.10.7 alphaOutset (Alpha Inset/Outset Effect)

This element specifies an alpha outset/inset effect.

This is equivalent to an alpha ceiling, followed by alpha blur, followed by either an alpha ceiling (positive radius) or alpha floor (negative radius).

Parent Elements
cont (§5.1.10.20); effectDag (§5.1.10.25)

Attributes	Description
rad (Radius)	Specifies the radius of outset/inset. The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AlphaOutsetEffect">
   <attribute name="rad" type="ST_Coordinate" use="optional" default="0"/>
</complexType>
```

5.1.10.8 alphaRepl (Alpha Replace Effect)

This element specifies an alpha replace effect.

Effect alpha (opacity) values are replaced by a fixed alpha.

Parent Elements
blip (§5.1.10.13); cont (§5.1.10.20); effectDag (§5.1.10.25)

Attributes	Description
a (Alpha)	Specifies the new opacity value. The possible values for this attribute are defined by the ST_PositiveFixedPercentage simple type (§5.1.12.45).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AlphaReplaceEffect">
   <attribute name="a" type="ST_PositiveFixedPercentage" use="required"/>
</complexType>
```

5.1.10.9 bevel (Line Join Bevel)

This element specifies a Bevel Line Join.

A bevel joint specifies that an angle joint is used to connect lines.

Parent Elements
ln (§5.1.2.1.24); lnB (§5.1.6.3); lnBlToTr (§5.1.6.4); lnL (§5.1.6.5); lnR (§5.1.6.6); lnT (§5.1.6.7); lnTlToBr (§5.1.6.8); uLn (§5.1.5.3.14)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LineJoinBevel"/>
```

5.1.10.10 bgClr (Background color)

This element specifies the background color of a Pattern fill.

Parent Elements
pattFill (§5.1.10.47)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

5.1.10.11 biLevel (Bi-Level (Black/White) Effect)

This element specifies a bi-level (black/white) effect. Input colors whose luminance is less than the specified threshold value are changed to black. Input colors whose luminance are greater than or equal the specified value are set to white. The alpha effect values are unaffected by this effect.

Parent Elements
blip (§5.1.10.13); cont (§5.1.10.20); effectDag (§5.1.10.25)

Attributes	Description
thresh (Threshold)	Specifies the luminance threshold for the Bi-Level effect. Values greater than or equal to the threshold are set to white. Values lesser than the threshold are set to black. The possible values for this attribute are defined by the ST_PositiveFixedPercentage simple type (§5.1.12.45).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BiLevelEffect">
   <attribute name="thresh" type="ST_PositiveFixedPercentage" use="required"/>
</complexType>
```

5.1.10.12 blend (Blend Effect)

This element specifies a blend of several effects. The container specifies the raw effects to blend while the blend mode specifies how the effects are to be blended.

Parent Elements
cont (§5.1.10.20); effectDag (§5.1.10.25)

Child Elements	Subclause
cont (Effect Container)	§5.1.10.20

Attributes	Description
blend (Blend Mode)	Specifies how to blend the two effects. The possible values for this attribute are defined by the ST_BlendMode simple type (§5.1.12.11).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BlendEffect">
   <sequence>
      <element name="cont" type="CT_EffectContainer" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="blend" type="ST_BlendMode" use="required"/>
</complexType>
```

5.1.10.13 blip (Blip)

This element specifies the existence of an image (binary large image or picture) and contains a reference to the image data.

Parent Elements
blipFill (§5.8.2.2); blipFill (§5.1.10.14); blipFill (§5.6.2.2); blipFill (§5.2.2.1); blipFill (§4.4.1.4); buBlip

Parent Elements
(§5.1.5.4.2)

Child Elements	Subclause
alphaBiLevel (Alpha Bi-Level Effect)	§5.1.10.1
alphaCeiling (Alpha Ceiling Effect)	§5.1.10.2
alphaFloor (Alpha Floor Effect)	§5.1.10.3
alphaInv (Alpha Inverse Effect)	§5.1.10.4
alphaMod (Alpha Modulate Effect)	§5.1.10.5
alphaModFix (Alpha Modulate Fixed Effect)	§5.1.10.6
alphaRepl (Alpha Replace Effect)	§5.1.10.8
biLevel (Bi-Level (Black/White) Effect)	§5.1.10.11
blur (Blur Effect)	§5.1.10.15
clrChange (Color Change Effect)	§5.1.10.16
clrRepl (Solid Color Replacement)	§5.1.10.18
duotone (Duotone Effect)	§5.1.10.23
extLst (Extension List)	§5.1.2.1.15
fillOverlay (Fill Overlay Effect)	§5.1.10.29
grayscl (Gray Scale Effect)	§5.1.10.34
hsl (Hue Saturation Luminance Effect)	§5.1.10.39
lum (Luminance Effect)	§5.1.10.42
tint (Tint Effect)	§5.1.10.60

Attributes	Description
cstate (Compression State)	Specifies the compression state with which the picture is stored. This allows the application to specify the amount of compression that has been applied to a picture. The possible values for this attribute are defined by the ST_BlipCompression simple type (§5.1.12.12).
embed (Embedded Picture Reference) Namespace: .../officeDocument/2006/relationships	Specifies the identification information for an embedded picture. This attribute is used to specify an image that resides locally within the file. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
link (Linked Picture	Specifies the identification information for a linked picture. This attribute is used to

Attributes	Description
Reference) Namespace: .../officeDocument/2006/relationships	specify an image that does not reside within this file. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Blip">
   <sequence>
      <choice minOccurs="0" maxOccurs="unbounded">
         <element name="alphaBiLevel" type="CT_AlphaBiLevelEffect" minOccurs="1" maxOccurs="1"/>
         <element name="alphaCeiling" type="CT_AlphaCeilingEffect" minOccurs="1" maxOccurs="1"/>
         <element name="alphaFloor" type="CT_AlphaFloorEffect" minOccurs="1" maxOccurs="1"/>
         <element name="alphaInv" type="CT_AlphaInverseEffect" minOccurs="1" maxOccurs="1"/>
         <element name="alphaMod" type="CT_AlphaModulateEffect" minOccurs="1" maxOccurs="1"/>
         <element name="alphaModFix" type="CT_AlphaModulateFixedEffect" minOccurs="1"
           maxOccurs="1"/>
         <element name="alphaRepl" type="CT_AlphaReplaceEffect" minOccurs="1" maxOccurs="1"/>
         <element name="biLevel" type="CT_BiLevelEffect" minOccurs="1" maxOccurs="1"/>
         <element name="blur" type="CT_BlurEffect" minOccurs="1" maxOccurs="1"/>
         <element name="clrChange" type="CT_ColorChangeEffect" minOccurs="1" maxOccurs="1"/>
         <element name="clrRepl" type="CT_ColorReplaceEffect" minOccurs="1" maxOccurs="1"/>
         <element name="duotone" type="CT_DuotoneEffect" minOccurs="1" maxOccurs="1"/>
         <element name="fillOverlay" type="CT_FillOverlayEffect" minOccurs="1" maxOccurs="1"/>
         <element name="grayscl" type="CT_GrayscaleEffect" minOccurs="1" maxOccurs="1"/>
         <element name="hsl" type="CT_HSLEffect" minOccurs="1" maxOccurs="1"/>
         <element name="lum" type="CT_LuminanceEffect" minOccurs="1" maxOccurs="1"/>
         <element name="tint" type="CT_TintEffect" minOccurs="1" maxOccurs="1"/>
      </choice>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attributeGroup ref="AG_Blob"/>
   <attribute name="cstate" type="ST_BlipCompression" use="optional" default="none"/>
</complexType>
```

5.1.10.14 blipFill (Picture Fill)

This element specifies the type of picture fill that the picture object will have. Because a picture has a picture fill already by default, it is possible to have two fills specified for a picture object. An example of this is shown below.

[*Example*: Consider the picture below that has a blip fill applied to it. The image used to fill this picture object has transparent pixels instead of white pixels.

```
<p:pic>
  …
  <p:blipFill>
    <a:blip r:embed="rId2"/>
    <a:stretch>
      <a:fillRect/>
    </a:stretch>
  </p:blipFill>
  …
</p:pic>
```

The above picture object is shown as an example of this fill type. *end example*]

[*Example*: Consider now the same picture object but with an additional gradient fill applied within the shape properties portion of the picture.

```
<p:pic>
  …
  <p:blipFill>
    <a:blip r:embed="rId2"/>
    <a:stretch>
      <a:fillRect/>
    </a:stretch>
  </p:blipFill>
  <p:spPr>
    <a:gradFill>
      <a:gsLst>
        <a:gs pos="0">
          <a:schemeClr val="tx2">
            <a:shade val="50000"/>
          </a:schemeClr>
        </a:gs>
```

```
        <a:gs pos="39999">
          <a:schemeClr val="tx2">
            <a:tint val="20000"/>
          </a:schemeClr>
        </a:gs>
        <a:gs pos="70000">
          <a:srgbClr val="C4D6EB"/>
        </a:gs>
        <a:gs pos="100000">
          <a:schemeClr val="bg1"/>
        </a:gs>
      </a:gsLst>
    </a:gradFill>
  </p:spPr>
  …
</p:pic>
```

The above picture object is shown as an example of this double fill type. *end example*]

Parent Elements
bg (§5.9.3.1); bgFillStyleLst (§5.1.4.1.7); bgPr (§4.4.1.2); defRPr (§5.1.5.3.2); endParaRPr (§5.1.5.2.3); fill (§5.1.10.28); fill (§5.1.4.2.9); fillOverlay (§5.1.10.29); fillStyleLst (§5.1.4.1.13); grpSpPr (§5.8.2.14); grpSpPr (§4.4.1.20); grpSpPr (§5.1.2.1.22); grpSpPr (§5.6.2.17); pic (§5.1.2.1.30); rPr (§5.1.5.3.9); spPr (§5.6.2.29); spPr (§5.7.2.198); spPr (§5.8.2.23); spPr (§5.1.2.1.35); spPr (§4.4.1.41); spPr (§5.9.3.7); spPr (§5.2.2.6); tblPr (§5.1.6.13); tcPr (§5.1.6.15); uFill (§5.1.5.3.12)

Child Elements	Subclause
blip (Blip)	§5.1.10.13
srcRect (Source Rectangle)	§5.1.10.55
stretch (Stretch)	§5.1.10.56

Child Elements	Subclause
tile (Tile)	§5.1.10.58

Attributes	Description
dpi (DPI Setting)	Specifies the DPI (dots per inch) used to calculate the size of the blip. If not present or zero, the DPI in the blip is used. [*Note*: This attribute is primarily used to keep track of the picture quality within a document. There are different levels of quality needed for print than on-screen viewing and thus a need to track this information. *end note*] The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
rotWithShape (Rotate With Shape)	Specifies that the fill should rotate with the shape. That is, when the shape that has been filled with a picture and the containing shape (say a rectangle) is transformed with a rotation then the fill will be transformed with the same rotation. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BlipFillProperties">
   <sequence>
      <element name="blip" type="CT_Blip" minOccurs="0" maxOccurs="1"/>
      <element name="srcRect" type="CT_RelativeRect" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_FillModeProperties" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="dpi" type="xsd:unsignedInt" use="optional"/>
   <attribute name="rotWithShape" type="xsd:boolean" use="optional"/>
</complexType>
```

5.1.10.15 blur (Blur Effect)

This element specifies a blur effect that is applied to the entire shape, including its fill. All color channels, including alpha, are affected.

Parent Elements
blip (§5.1.10.13); cont (§5.1.10.20); effectDag (§5.1.10.25); effectLst (§5.1.10.26)

Attributes	Description
grow (Grow Bounds)	Specifies whether the bounds of the object should be grown as a result of the blurring. True indicates the bounds are grown while false indicates that they are not. [*Example*:

Attributes	Description
	With grow set to false, the blur effect does not extend beyond the original bounds of the object:  With grow set to true, the blur effect may extend beyond the original bounds of the object:  *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
rad (Radius)	Specifies the radius of blur. The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BlurEffect">
   <attribute name="rad" type="ST_PositiveCoordinate" use="optional" default="0"/>
   <attribute name="grow" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.1.10.16 clrChange (Color Change Effect)

This element specifies a Color Change Effect. Instances of clrFrom are replaced with instances of clrTo.

Parent Elements
blip (§5.1.10.13); cont (§5.1.10.20); effectDag (§5.1.10.25)

Child Elements	Subclause
clrFrom (Change Color From)	§5.1.10.17

Child Elements	Subclause
clrTo (Change Color To)	§5.1.10.19

Attributes	Description
useA (Consider Alpha Values)	Specifies whether alpha values are considered for the effect. Effect alpha values are considered if useA is true, else they are ignored. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ColorChangeEffect">
   <sequence>
      <element name="clrFrom" type="CT_Color" minOccurs="1" maxOccurs="1"/>
      <element name="clrTo" type="CT_Color" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="useA" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.1.10.17 clrFrom (Change Color From)

This element specifies a color getting removed in a color change effect. It is the "from" or source input color.

Parent Elements
clrChange (§5.1.10.16)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

5.1.10.18 clrRepl (Solid Color Replacement)

This element specifies a solid color replacement value. All effect colors are changed to a fixed color. Alpha values are unaffected.

Parent Elements
blip (§5.1.10.13); cont (§5.1.10.20); effectDag (§5.1.10.25)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ColorReplaceEffect">
   <sequence>
      <group ref="EG_ColorChoice" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.10.19 clrTo (Change Color To)

This element specifies the color which replaces the clrFrom in a clrChange effect. This is the "target" or "to" color in the color change effect.

Parent Elements
clrChange (§5.1.10.16)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

5.1.10.20 cont (Effect Container)

This element specifies an Effect Container. It is a list of effects.

Parent Elements
alphaMod (§5.1.10.5); blend (§5.1.10.12); cont (§5.1.10.20); effectDag (§5.1.10.25)

Child Elements	Subclause
alphaBiLevel (Alpha Bi-Level Effect)	§5.1.10.1
alphaCeiling (Alpha Ceiling Effect)	§5.1.10.2
alphaFloor (Alpha Floor Effect)	§5.1.10.3
alphaInv (Alpha Inverse Effect)	§5.1.10.4
alphaMod (Alpha Modulate Effect)	§5.1.10.5
alphaModFix (Alpha Modulate Fixed Effect)	§5.1.10.6
alphaOutset (Alpha Inset/Outset Effect)	§5.1.10.7
alphaRepl (Alpha Replace Effect)	§5.1.10.8
biLevel (Bi-Level (Black/White) Effect)	§5.1.10.11
blend (Blend Effect)	§5.1.10.12
blur (Blur Effect)	§5.1.10.15
clrChange (Color Change Effect)	§5.1.10.16
clrRepl (Solid Color Replacement)	§5.1.10.18
cont (Effect Container)	§5.1.10.20
duotone (Duotone Effect)	§5.1.10.23
effect (Effect)	§5.1.10.24
fill (Fill)	§5.1.10.28
fillOverlay (Fill Overlay Effect)	§5.1.10.29
glow (Glow Effect)	§5.1.10.32
grayscl (Gray Scale Effect)	§5.1.10.34
hsl (Hue Saturation Luminance Effect)	§5.1.10.39
innerShdw (Inner Shadow Effect)	§5.1.10.40
lum (Luminance Effect)	§5.1.10.42

Child Elements	Subclause
outerShdw (Outer Shadow Effect)	§5.1.10.45
prstShdw (Preset Shadow)	§5.1.10.49
reflection (Reflection Effect)	§5.1.10.50
relOff (Relative Offset Effect)	§5.1.10.51
softEdge (Soft Edge Effect)	§5.1.10.53
tint (Tint Effect)	§5.1.10.60
xfrm (Transform Effect)	§5.1.10.61

Attributes	Description
name (Name)	Specifies an optional name for this list of effects, so that it can be referred to later. Must be unique across all effect trees and effect containers. The possible values for this attribute are defined by the XML Schema token datatype.
type (Effect Container Type)	Specifies the type of container, either sibling or tree. The possible values for this attribute are defined by the ST_EffectContainerType simple type (§5.1.12.20).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EffectContainer">
   <group ref="EG_Effect" minOccurs="0" maxOccurs="unbounded"/>
   <attribute name="type" type="ST_EffectContainerType" use="optional" default="sib"/>
   <attribute name="name" type="xsd:token" use="optional"/>
</complexType>
```

5.1.10.21 custDash (Custom Dash)

This element specifies a custom dashing scheme. It is a list of dash stop elements which represent building block atoms upon which the custom dashing scheme is built.

Parent Elements
ln (§5.1.2.1.24); lnB (§5.1.6.3); lnBlToTr (§5.1.6.4); lnL (§5.1.6.5); lnR (§5.1.6.6); lnT (§5.1.6.7); lnTlToBr (§5.1.6.8); uLn (§5.1.5.3.14)

Child Elements	Subclause
ds (Dash Stop)	§5.1.10.22

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DashStopList">
   <sequence>
      <element name="ds" type="CT_DashStop" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.1.10.22 ds (Dash Stop)

This element specifies a dash stop primitive. Dashing schemes are built by specifying an ordered list of dash stop primitive. A dash stop primitive consists of a dash and a space.

Parent Elements
custDash (§5.1.10.21)

Attributes	Description
d (Dash Length)	Specifies the length of the dash relative to the line width. The possible values for this attribute are defined by the ST_PositivePercentage simple type (§5.1.12.46).
sp (Space Length)	Specifies the length of the space relative to the line width. The possible values for this attribute are defined by the ST_PositivePercentage simple type (§5.1.12.46).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DashStop">
   <attribute name="d" type="ST_PositivePercentage" use="required"/>
   <attribute name="sp" type="ST_PositivePercentage" use="required"/>
</complexType>
```

5.1.10.23 duotone (Duotone Effect)

This element specifies a duotone effect.

For each pixel, combines clr1 and clr2 through a linear interpolation to determine the new color for that pixel.

Parent Elements
blip (§5.1.10.13); cont (§5.1.10.20); effectDag (§5.1.10.25)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22

Child Elements	Subclause
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DuotoneEffect">
   <sequence>
      <group ref="EG_ColorChoice" minOccurs="2" maxOccurs="2"/>
   </sequence>
</complexType>
```

5.1.10.24 effect (Effect)

This element specifies a reference to an existing effect container.

Parent Elements
cont (§5.1.10.20); effectDag (§5.1.10.25)

Attributes	Description
ref (Reference)	Specifies the reference. Its value may be the name of an effect container, or one of four special references: fill - refers to the fill effect line - refers to the line effect fillLine - refers to the combined fill and line effects children - refers to the combined effects from logical child shapes or text The possible values for this attribute are defined by the XML Schema token datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EffectReference">
   <attribute name="ref" type="xsd:token"/>
</complexType>
```

5.1.10.25 effectDag (Effect Container)

This element specifies a list of effects. Effects are applied in the order specified by the container type (sibling or tree).

[*Note:* An effectDag element may contain multiple effect containers as child elements. Effect containers with different types may be combined in an effectDag to define a directed acyclic graph (DAG) that specifies the order in which all effects are applied. *end note*]

Parent Elements

Parent Elements
bg (§5.9.3.1); bgPr (§4.4.1.2); defRPr (§5.1.5.3.2); effect (§5.1.4.2.7); effectStyle (§5.1.4.1.11); endParaRPr (§5.1.5.2.3); grpSpPr (§5.8.2.14); grpSpPr (§4.4.1.20); grpSpPr (§5.1.2.1.22); grpSpPr (§5.6.2.17); rPr (§5.1.5.3.9); spPr (§5.6.2.29); spPr (§5.7.2.198); spPr (§5.8.2.23); spPr (§5.1.2.1.35); spPr (§4.4.1.41); spPr (§5.9.3.7); spPr (§5.2.2.6); tblPr (§5.1.6.13); whole (§5.9.3.9)

Child Elements	Subclause
alphaBiLevel (Alpha Bi-Level Effect)	§5.1.10.1
alphaCeiling (Alpha Ceiling Effect)	§5.1.10.2
alphaFloor (Alpha Floor Effect)	§5.1.10.3
alphaInv (Alpha Inverse Effect)	§5.1.10.4
alphaMod (Alpha Modulate Effect)	§5.1.10.5
alphaModFix (Alpha Modulate Fixed Effect)	§5.1.10.6
alphaOutset (Alpha Inset/Outset Effect)	§5.1.10.7
alphaRepl (Alpha Replace Effect)	§5.1.10.8
biLevel (Bi-Level (Black/White) Effect)	§5.1.10.11
blend (Blend Effect)	§5.1.10.12
blur (Blur Effect)	§5.1.10.15
clrChange (Color Change Effect)	§5.1.10.16
clrRepl (Solid Color Replacement)	§5.1.10.18
cont (Effect Container)	§5.1.10.20
duotone (Duotone Effect)	§5.1.10.23
effect (Effect)	§5.1.10.24
fill (Fill)	§5.1.10.28
fillOverlay (Fill Overlay Effect)	§5.1.10.29
glow (Glow Effect)	§5.1.10.32
grayscl (Gray Scale Effect)	§5.1.10.34
hsl (Hue Saturation Luminance Effect)	§5.1.10.39
innerShdw (Inner Shadow Effect)	§5.1.10.40
lum (Luminance Effect)	§5.1.10.42
outerShdw (Outer Shadow Effect)	§5.1.10.45
prstShdw (Preset Shadow)	§5.1.10.49
reflection (Reflection Effect)	§5.1.10.50
relOff (Relative Offset Effect)	§5.1.10.51
softEdge (Soft Edge Effect)	§5.1.10.53

Child Elements	Subclause
tint (Tint Effect)	§5.1.10.60
xfrm (Transform Effect)	§5.1.10.61

Attributes	Description
name (Name)	Specifies an optional name for this list of effects, so that it can be referred to later. Must be unique across all effect trees and effect containers. The possible values for this attribute are defined by the XML Schema token datatype.
type (Effect Container Type)	Specifies the type of container, either sibling or tree. The possible values for this attribute are defined by the ST_EffectContainerType simple type (§5.1.12.20).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EffectContainer">
   <group ref="EG_Effect" minOccurs="0" maxOccurs="unbounded"/>
   <attribute name="type" type="ST_EffectContainerType" use="optional" default="sib"/>
   <attribute name="name" type="xsd:token" use="optional"/>
</complexType>
```

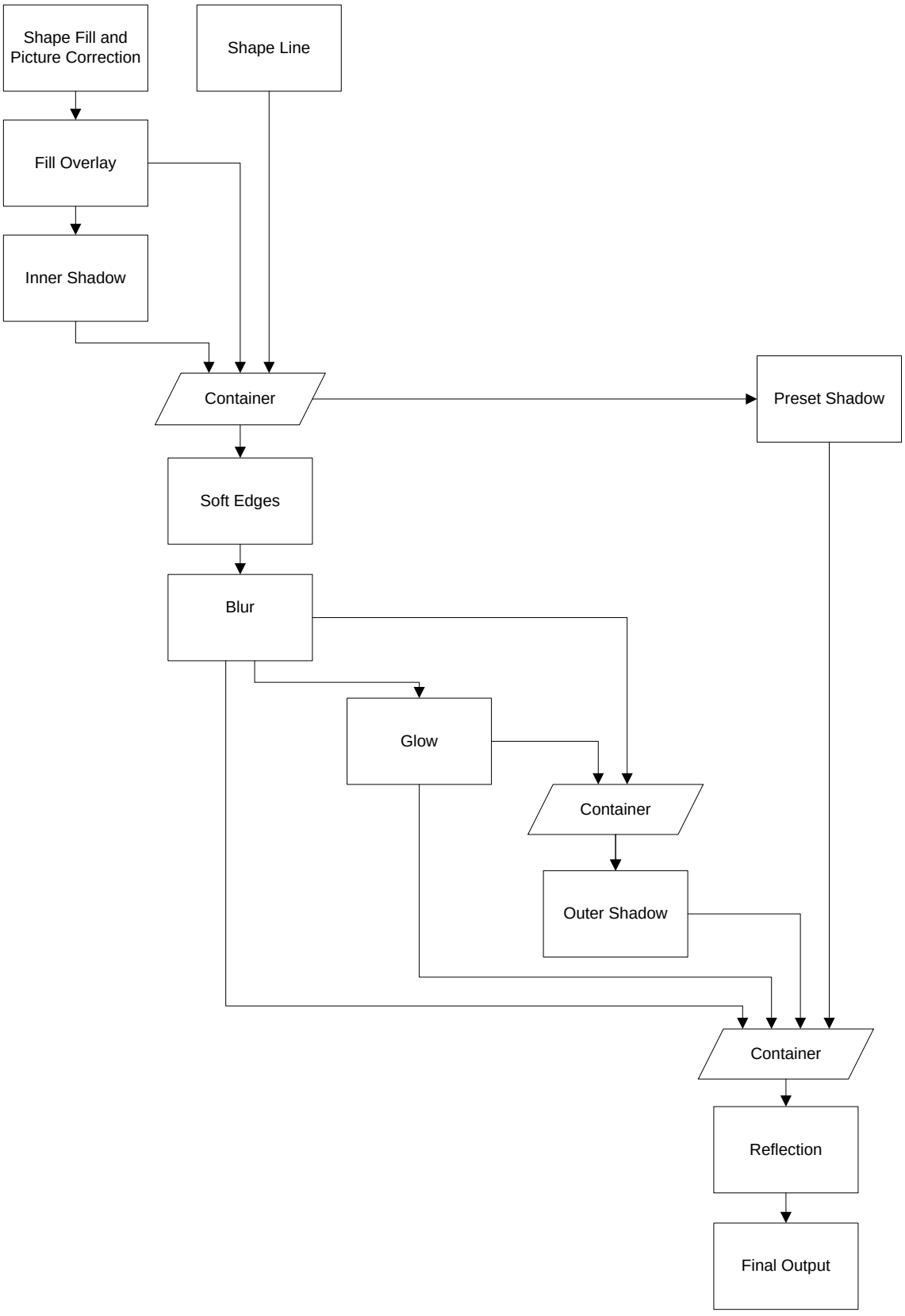
5.1.10.26 effectLst (Effect Container)

This element specifies a list of effects. Effects in an effectLst are applied in the default order by the rendering engine. The following diagrams illustrate the order in which effects are to be applied, both for shapes and for group shapes.

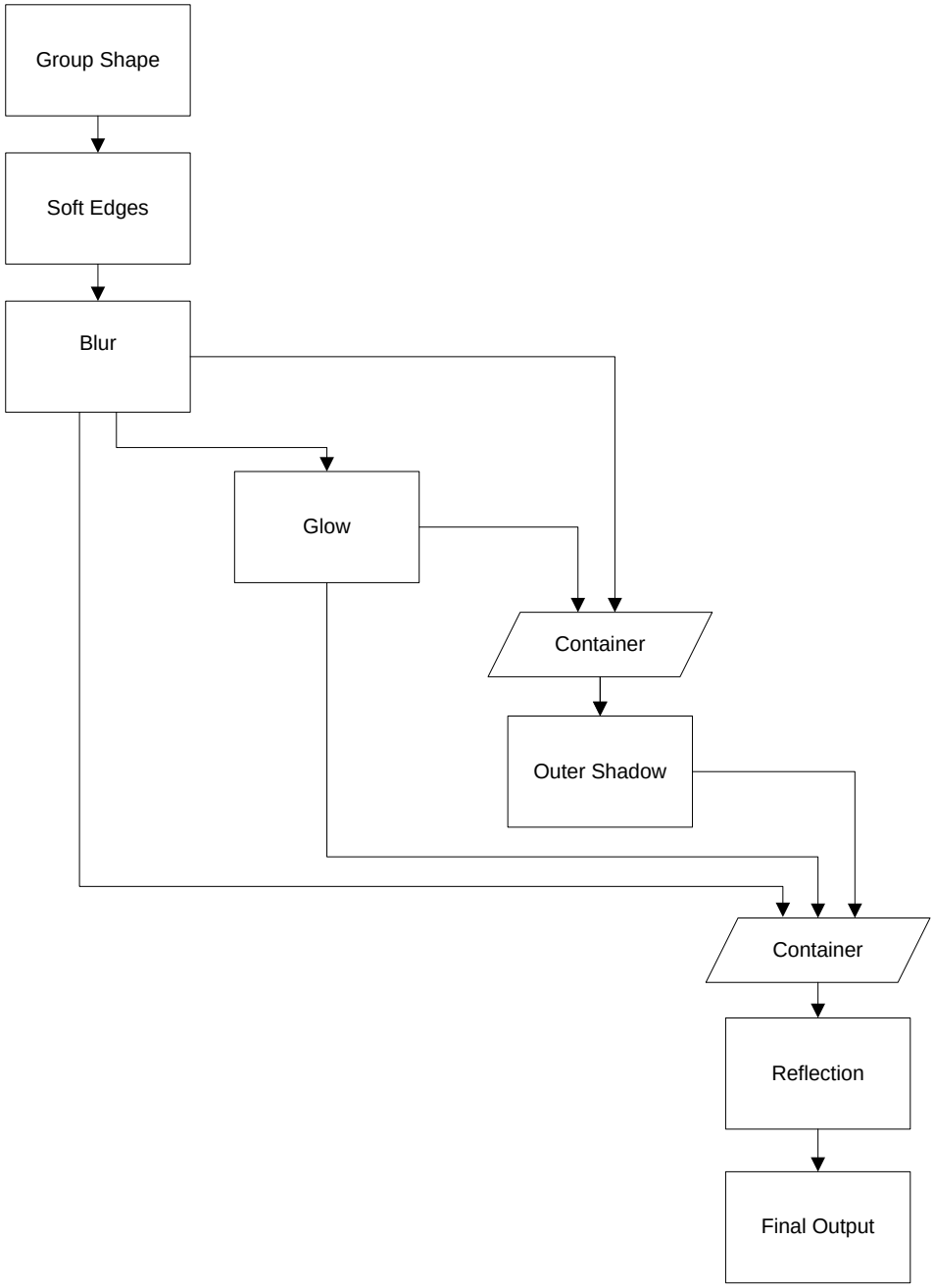
[*Note:* The output of many effects does not include the input shape. For effects that should be applied to the result of previous effects as well as the original shape, a container is used to group the inputs together. *end note*]

[*Example:* Outer Shadow is applied both to the original shape and the original shape's glow. The result of blur contains the original shape, while the result of glow contains only the added glow. Therefore, a container that groups the blur result with the glow result is used as the input to Outer Shadow. *end example*]

effectLst Processing for Shapes



effectLst Processing for Group Shapes



Parent Elements
bg (§5.9.3.1); bgPr (§4.4.1.2); defRPr (§5.1.5.3.2); effect (§5.1.4.2.7); effectStyle (§5.1.4.1.11); endParaRPr (§5.1.5.2.3); grpSpPr (§5.8.2.14); grpSpPr (§4.4.1.20); grpSpPr (§5.1.2.1.22); grpSpPr (§5.6.2.17); rPr (§5.1.5.3.9); spPr (§5.6.2.29); spPr (§5.7.2.198); spPr (§5.8.2.23); spPr (§5.1.2.1.35); spPr (§4.4.1.41); spPr (§5.9.3.7); spPr (§5.2.2.6); tblPr (§5.1.6.13); whole (§5.9.3.9)

Child Elements	Subclause
blur (Blur Effect)	§5.1.10.15
fillOverlay (Fill Overlay Effect)	§5.1.10.29
glow (Glow Effect)	§5.1.10.32
innerShdw (Inner Shadow Effect)	§5.1.10.40
outerShdw (Outer Shadow Effect)	§5.1.10.45
prstShdw (Preset Shadow)	§5.1.10.49
reflection (Reflection Effect)	§5.1.10.50
softEdge (Soft Edge Effect)	§5.1.10.53

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EffectList">
   <sequence>
      <element name="blur" type="CT_BlurEffect" minOccurs="0" maxOccurs="1"/>
      <element name="fillOverlay" type="CT_FillOverlayEffect" minOccurs="0" maxOccurs="1"/>
      <element name="glow" type="CT_GlowEffect" minOccurs="0" maxOccurs="1"/>
      <element name="innerShdw" type="CT_InnerShadowEffect" minOccurs="0" maxOccurs="1"/>
      <element name="outerShdw" type="CT_OuterShadowEffect" minOccurs="0" maxOccurs="1"/>
      <element name="prstShdw" type="CT_PresetShadowEffect" minOccurs="0" maxOccurs="1"/>
      <element name="reflection" type="CT_ReflectionEffect" minOccurs="0" maxOccurs="1"/>
      <element name="softEdge" type="CT_SoftEdgesEffect" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.10.27 fgClr (Foreground color)

This element specifies the foreground color of a pattern fill.

Parent Elements
pattFill (§5.1.10.47)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Color">
   <sequence>
      <group ref="EG_ColorChoice"/>
   </sequence>
</complexType>
```

5.1.10.28 fill (Fill)

This element specifies a fill which is one of blipFill, gradFill, grpFill, noFill, pattFill or solidFill.

Parent Elements
cont (§5.1.10.20); effectDag (§5.1.10.25)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
solidFill (Solid Fill)	§5.1.10.54

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FillEffect">
   <sequence>
      <group ref="EG_FillProperties" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.10.29 fillOverlay (Fill Overlay Effect)

This element specifies a fill overlay effect. A fill overlay may be used to specify an additional fill for an object and blend the two fills together.

Parent Elements
blip (§5.1.10.13); cont (§5.1.10.20); effectDag (§5.1.10.25); effectLst (§5.1.10.26)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35
noFill (No Fill)	§5.1.10.44

Child Elements	Subclause
pattFill (Pattern Fill)	§5.1.10.47
solidFill (Solid Fill)	§5.1.10.54

Attributes	Description
blend (Blend)	Specifies how to blend the fill with the base effect. The possible values for this attribute are defined by the ST_BlendMode simple type (§5.1.12.11).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FillOverlayEffect">
   <sequence>
      <group ref="EG_FillProperties" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="blend" type="ST_BlendMode" use="required"/>
</complexType>
```

5.1.10.30 fillRect (Fill Rectangle)

This element specifies a fill rectangle. When stretching of an image is specified, a source rectangle, srcRect, is scaled to fit the specified fill rectangle.

Each edge of the fill rectangle is defined by a percentage offset from the corresponding edge of the shape's bounding box. A positive percentage specifies an inset, while a negative percentage specifies an outset. For example, a left offset of 25% specifies that the left edge of the fill rectangle is located to the right of the bounding box's left edge by an amount equal to 25% of the bounding box's width.

[*Example*:

```
<a:blipFill>
  <a:blip r:embed="rId2"/>
  <a:stretch>
      <a:fillRect b="10000" r="25000"/>
  </a:stretch>
</a:blipFill>
```

The above image is stretched to fill the entire rectangle except for the bottom 10% and rightmost 25%.

end example]

Parent Elements
stretch (§5.1.10.56)

Attributes	Description
b (Bottom Offset)	Specifies the bottom edge of the rectangle. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
l (Left Offset)	Specifies the left edge of the rectangle. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
r (Right Offset)	Specifies the right edge of the rectangle. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
t (Top Offset)	Specifies the top edge of the rectangle. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

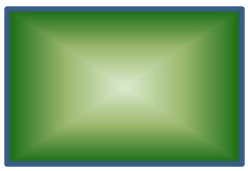
```
<complexType name="CT_RelativeRect">
    <attribute name="l" type="ST_Percentage" use="optional" default="0"/>
    <attribute name="t" type="ST_Percentage" use="optional" default="0"/>
    <attribute name="r" type="ST_Percentage" use="optional" default="0"/>
    <attribute name="b" type="ST_Percentage" use="optional" default="0"/>
</complexType>
```

5.1.10.31 fillToRect (Fill To Rectangle)

This element defines the "focus" rectangle for the center shade, specified relative to the fill tile rectangle. The center shade fills the entire tile except the margins specified by each attribute.

Each edge of the center shade rectangle is defined by a percentage offset from the corresponding edge of the tile rectangle. A positive percentage specifies an inset, while a negative percentage specifies an outset. For example, a left offset of 25% specifies that the left edge of the center shade rectangle is located to the right of the tile rectangle's left edge by an amount equal to 25% of the tile rectangle's width.

[*Example*:



```
<a:path path="rect">
  <a:fillToRect l="50000" r="50000" t="50000" b="50000"/>
</a:path>
```

In the above shape, the rectangle defined by fillToRect is a single point in the center of the shape. This creates the effect of the center shade focusing at a point in the center of the region.

end example]

[*Example*:

```
<a:path path="rect">
  <a:fillToRect l="25000" t="25000" r="25000" b="0"/>
</a:path>
```

The center shade occupies the rectangle defined by excluding the topmost, leftmost, and rightmost 25% of the region. Therefore, the gradient fills the remaining leftmost 25%, topmost 25%, and rightmost 25% of the region.

end example]

Parent Elements
path (§5.1.10.46)

Attributes	Description
b (Bottom Offset)	Specifies the bottom edge of the rectangle. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
l (Left Offset)	Specifies the left edge of the rectangle. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
r (Right Offset)	Specifies the right edge of the rectangle.

Attributes	Description
	The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
t (Top Offset)	Specifies the top edge of the rectangle. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RelativeRect">
   <attribute name="l" type="ST_Percentage" use="optional" default="0"/>
   <attribute name="t" type="ST_Percentage" use="optional" default="0"/>
   <attribute name="r" type="ST_Percentage" use="optional" default="0"/>
   <attribute name="b" type="ST_Percentage" use="optional" default="0"/>
</complexType>
```

5.1.10.32 glow (Glow Effect)

This element specifies a glow effect, in which a color blurred outline is added outside the edges of the object.

Parent Elements
cont (§5.1.10.20); effectDag (§5.1.10.25); effectLst (§5.1.10.26)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

Attributes	Description
rad (Radius)	Specifies the radius of the glow. The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GlowEffect">
   <sequence>
      <group ref="EG_ColorChoice" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="rad" type="ST_PositiveCoordinate" use="optional" default="0"/>
</complexType>
```

5.1.10.33 gradFill (Gradient Fill)

This element defines a gradient fill.

A gradient fill is a fill which is characterized by a smooth gradual transition from one color to the next. At its simplest, it is a fill which transitions between two colors; or more generally, it may be a transition of any number of colors.

The desired transition colors and locations are defined in the gradient stop list (gsLst) child element.

The other child element defines the properties of the gradient fill (there are two styles-- a linear shade style as well as a path shade style)

[*Example*:

The following is a sample gradient fill, varying from blue to white:

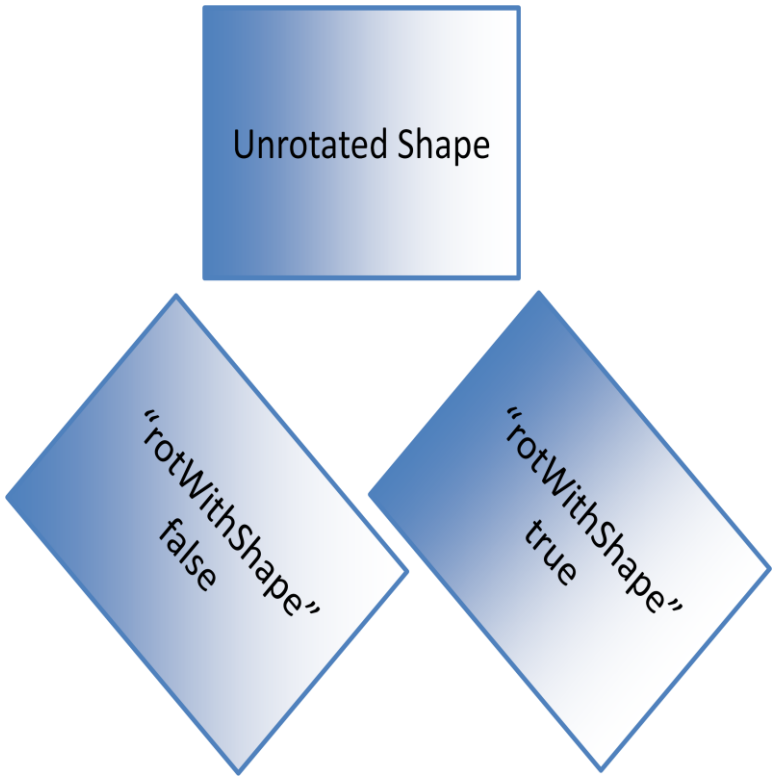
end example]

Parent Elements
bg (§5.9.3.1); bgFillStyleLst (§5.1.4.1.7); bgPr (§4.4.1.2); defRPr (§5.1.5.3.2); endParaRPr (§5.1.5.2.3); fill (§5.1.10.28); fill (§5.1.4.2.9); fillOverlay (§5.1.10.29); fillStyleLst (§5.1.4.1.13); grpSpPr (§5.8.2.14); grpSpPr (§4.4.1.20); grpSpPr (§5.1.2.1.22); grpSpPr (§5.6.2.17); ln (§5.1.2.1.24); lnB (§5.1.6.3); lnBlToTr (§5.1.6.4); lnL (§5.1.6.5); lnR (§5.1.6.6); lnT (§5.1.6.7); lnTlToBr (§5.1.6.8); rPr (§5.1.5.3.9); spPr (§5.6.2.29); spPr (§5.7.2.198); spPr (§5.8.2.23); spPr (§5.1.2.1.35); spPr (§4.4.1.41); spPr (§5.9.3.7); spPr (§5.2.2.6); tblPr (§5.1.6.13); tcPr (§5.1.6.15); uFill (§5.1.5.3.12); uLn (§5.1.5.3.14)

Child Elements	Subclause
gsLst (Gradient Stop List)	§5.1.10.37
lin (Linear Gradient Fill)	§5.1.10.41
path (Path Gradient)	§5.1.10.46

Child Elements	Subclause
tileRect (Tile Rectangle)	§5.1.10.59

Attributes	Description
flip (Tile Flip)	Specifies the direction(s) in which to flip the gradient while tiling. Normally a gradient fill encompasses the entire bounding box of the shape which contains the fill. However, with the tileRect element, it is possible to define a "tile" rectangle which is smaller than the bounding box. In this situation, the gradient fill is encompassed within the tile rectangle, and the tile rectangle is tiled across the bounding box to fill the entire area. The possible values for this attribute are defined by the ST_TileFlipMode simple type (§5.1.12.86).
rotWithShape (Rotate With Shape)	Specifies if a fill will rotate along with a shape when the shape is rotated. [*Example*: The following is a fill with the flip attribute set to "x". The black interior rectangle indicates the tile rectangle. Notice that the adjacent rectangle to the right in the tile has been flipped along the x-axis.

Attributes	Description
	 *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GradientFillProperties">
   <sequence>
      <element name="gsLst" type="CT_GradientStopList" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_ShadeProperties" minOccurs="0" maxOccurs="1"/>
      <element name="tileRect" type="CT_RelativeRect" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="flip" type="ST_TileFlipMode" use="optional"/>
   <attribute name="rotWithShape" type="xsd:boolean" use="optional"/>
</complexType>
```

5.1.10.34 grayscl (Gray Scale Effect)

This element specifies a gray scale effect. Converts all effect color values to a shade of gray, corresponding to their luminance. Effect alpha (opacity) values are unaffected.

Parent Elements
blip (§5.1.10.13); cont (§5.1.10.20); effectDag (§5.1.10.25)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GrayscaleEffect"/>
```

5.1.10.35 grpFill (Group Fill)

This element specifies a group fill. When specified, this setting indicates that the parent element is part of a group and should inherit the fill properties of the group.

Parent Elements
bg (§5.9.3.1); bgFillStyleLst (§5.1.4.1.7); bgPr (§4.4.1.2); defRPr (§5.1.5.3.2); endParaRPr (§5.1.5.2.3); fill (§5.1.10.28); fill (§5.1.4.2.9); fillOverlay (§5.1.10.29); fillStyleLst (§5.1.4.1.13); grpSpPr (§5.8.2.14); grpSpPr (§4.4.1.20); grpSpPr (§5.1.2.1.22); grpSpPr (§5.6.2.17); rPr (§5.1.5.3.9); spPr (§5.6.2.29); spPr (§5.7.2.198); spPr (§5.8.2.23); spPr (§5.1.2.1.35); spPr (§4.4.1.41); spPr (§5.9.3.7); spPr (§5.2.2.6); tblPr (§5.1.6.13); tcPr (§5.1.6.15); uFill (§5.1.5.3.12)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GroupFillProperties"/>
```

5.1.10.36 gs (Gradient stops)

This element defines a gradient stop. A gradient stop consists of a position where the stop appears in the color band.

Parent Elements
gsLst (§5.1.10.37)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

Attributes	Description
pos (Position)	Specifies where this gradient stop should appear in the color band. This position is specified in the range [0%, 100%], which corresponds to the beginning and the end of the color band respectively. The possible values for this attribute are defined by the ST_PositiveFixedPercentage simple type (§5.1.12.45).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GradientStop">
   <sequence>
      <group ref="EG_ColorChoice" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="pos" type="ST_PositiveFixedPercentage" use="required"/>
</complexType>
```

5.1.10.37 gsLst (Gradient Stop List)

The list of gradient stops that specifies the gradient colors and their relative positions in the color band.

Parent Elements
gradFill (§5.1.10.33)

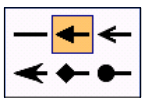
Child Elements	Subclause
gs (Gradient stops)	§5.1.10.36

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GradientStopList">
   <sequence>
      <element name="gs" type="CT_GradientStop" minOccurs="2" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.1.10.38 headEnd (Line Head/End Style)

This element specifies decorations which can be added to the head of a line.



Parent Elements
ln (§5.1.2.1.24); lnB (§5.1.6.3); lnBlToTr (§5.1.6.4); lnL (§5.1.6.5); lnR (§5.1.6.6); lnT (§5.1.6.7); lnTlToBr (§5.1.6.8); uLn (§5.1.5.3.14)

Attributes	Description
len (Length of Head/End)	Specifies the line end length in relation to the line width. The possible values for this attribute are defined by the ST_LineEndLength simple type (§5.1.12.32).
type (Line Head/End Type)	Specifies the line end decoration, such as a triangle or arrowhead. The possible values for this attribute are defined by the ST_LineEndType simple type (§5.1.12.33).
w (Width of Head/End)	Specifies the line end width in relation to the line width. The possible values for this attribute are defined by the ST_LineEndWidth simple type (§5.1.12.34).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LineEndProperties">
   <attribute name="type" type="ST_LineEndType" use="optional"/>
   <attribute name="w" type="ST_LineEndWidth" use="optional"/>
   <attribute name="len" type="ST_LineEndLength" use="optional"/>
</complexType>
```

5.1.10.39 hsl (Hue Saturation Luminance Effect)

This element specifies a hue/saturation/luminance effect. The hue, saturation, and luminance may each be adjusted relative to its current value.

Parent Elements
blip (§5.1.10.13); cont (§5.1.10.20); effectDag (§5.1.10.25)

Attributes	Description
hue (Hue)	Specifies the number of degrees by which the hue is adjusted. The possible values for this attribute are defined by the ST_PositiveFixedAngle simple type (§5.1.12.44).
lum (Luminance)	Specifies the percentage by which the luminance is adjusted. The possible values for this attribute are defined by the ST_FixedPercentage simple type (§5.1.12.22).
sat (Saturation)	Specifies the percentage by which the saturation is adjusted. The possible values for this attribute are defined by the ST_FixedPercentage simple type (§5.1.12.22).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HSLEffect">
   <attribute name="hue" type="ST_PositiveFixedAngle" use="optional" default="0"/>
   <attribute name="sat" type="ST_FixedPercentage" use="optional" default="0"/>
   <attribute name="lum" type="ST_FixedPercentage" use="optional" default="0"/>
</complexType>
```

5.1.10.40 innerShdw (Inner Shadow Effect)

This element specifies an inner shadow effect. A shadow is applied within the edges of the object according to the parameters given by the attributes.

Parent Elements
cont (§5.1.10.20); effectDag (§5.1.10.25); effectLst (§5.1.10.26)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

Attributes	Description
blurRad (Blur Radius)	Specifies the blur radius. The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).
dir (Direction)	Specifies the direction to offset the shadow. The possible values for this attribute are defined by the ST_PositiveFixedAngle simple type (§5.1.12.44).
dist (Distance)	Specifies how far to offset the shadow.

Attributes	Description
	The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_InnerShadowEffect">
   <sequence>
      <group ref="EG_ColorChoice" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="blurRad" type="ST_PositiveCoordinate" use="optional" default="0"/>
   <attribute name="dist" type="ST_PositiveCoordinate" use="optional" default="0"/>
   <attribute name="dir" type="ST_PositiveFixedAngle" use="optional" default="0"/>
</complexType>
```

5.1.10.41 lin (Linear Gradient Fill)

This element specifies a linear gradient.

Parent Elements
gradFill (§5.1.10.33)

Attributes	Description
ang (Angle)	Specifies the direction of color change for the gradient. To define this angle, let its value be x measured clockwise. Then (-sin x, cos x) is a vector parallel to the line of constant color in the gradient fill. The possible values for this attribute are defined by the ST_PositiveFixedAngle simple type (§5.1.12.44).
scaled (Scaled)	Whether the gradient angle scales with the fill region. Mathematically, if this flag is true, then we scale the gradient vector (cos x , sin x) by the width (w) and height (h) of the fill region, so that the vector becomes (w cos x, h sin x) (before normalization). Observe that now if the gradient angle is 45 degrees, the gradient vector is (w, h), which goes from top-left to bottom-right of the fill region. If this flag is false, the gradient angle is independent of the fill region and will not be scaled using the manipulation described above. So a 45-degree gradient angle always give a gradient band whose line of constant color is parallel to the vector (1, -1). The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LinearShadeProperties">
   <attribute name="ang" type="ST_PositiveFixedAngle" use="optional"/>
   <attribute name="scaled" type="xsd:boolean" use="optional"/>
</complexType>
```

5.1.10.42 lum (Luminance Effect)

This element specifies a luminance effect. Brightness linearly shifts all colors closer to white or black. Contrast scales all colors to be either closer or further apart.

Parent Elements
blip (§5.1.10.13); cont (§5.1.10.20); effectDag (§5.1.10.25)

Attributes	Description
bright (Brightness)	Specifies the percent to change the brightness. The possible values for this attribute are defined by the ST_FixedPercentage simple type (§5.1.12.22).
contrast (Contrast)	Specifies the percent to change the contrast. The possible values for this attribute are defined by the ST_FixedPercentage simple type (§5.1.12.22).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LuminanceEffect">
   <attribute name="bright" type="ST_FixedPercentage" use="optional" default="0"/>
   <attribute name="contrast" type="ST_FixedPercentage" use="optional" default="0"/>
</complexType>
```

5.1.10.43 miter (Miter Line Join)

This element specifies that a line join shall be mitered.

[*Example*: The following sample illustrated two lines which are joined using a mitered style

end example]

Parent Elements
ln (§5.1.2.1.24); lnB (§5.1.6.3); lnBlToTr (§5.1.6.4); lnL (§5.1.6.5); lnR (§5.1.6.6); lnT (§5.1.6.7); lnTlToBr (§5.1.6.8); uLn (§5.1.5.3.14)

Attributes	Description
lim (Miter Join Limit)	Specifies the amount by which lines will be extended to form a miter join - otherwise miter joins can extend infinitely far (for lines which are almost parallel).

Attributes	Description
	The possible values for this attribute are defined by the ST_PositivePercentage simple type (§5.1.12.46).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LineJoinMiterProperties">
   <attribute name="lim" type="ST_PositivePercentage" use="optional"/>
</complexType>
```

5.1.10.44 noFill (No Fill)

This element specifies that no fill will be applied to the parent element.

Parent Elements
bg (§5.9.3.1); bgFillStyleLst (§5.1.4.1.7); bgPr (§4.4.1.2); defRPr (§5.1.5.3.2); endParaRPr (§5.1.5.2.3); fill (§5.1.10.28); fill (§5.1.4.2.9); fillOverlay (§5.1.10.29); fillStyleLst (§5.1.4.1.13); grpSpPr (§5.8.2.14); grpSpPr (§4.4.1.20); grpSpPr (§5.1.2.1.22); grpSpPr (§5.6.2.17); ln (§5.1.2.1.24); lnB (§5.1.6.3); lnBlToTr (§5.1.6.4); lnL (§5.1.6.5); lnR (§5.1.6.6); lnT (§5.1.6.7); lnTlToBr (§5.1.6.8); rPr (§5.1.5.3.9); spPr (§5.6.2.29); spPr (§5.7.2.198); spPr (§5.8.2.23); spPr (§5.1.2.1.35); spPr (§4.4.1.41); spPr (§5.9.3.7); spPr (§5.2.2.6); tblPr (§5.1.6.13); tcPr (§5.1.6.15); uFill (§5.1.5.3.12); uLn (§5.1.5.3.14)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NoFillProperties"/>
```

5.1.10.45 outerShdw (Outer Shadow Effect)

This element specifies an Outer Shadow Effect.

[*Example*: The following is an example of an outer shadow effect.

end example]

Parent Elements
cont (§5.1.10.20); effectDag (§5.1.10.25); effectLst (§5.1.10.26)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13

Child Elements	Subclause
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

Attributes	Description
algn (Shadow Alignment)	Specifies shadow alignment; alignment happens first, effectively setting the origin for scale, skew, and offset. The possible values for this attribute are defined by the ST_RectAlignment simple type (§5.1.12.53).
blurRad (Blur Radius)	Specifies the blur radius of the shadow. The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).
dir (Shadow Direction)	Specifies the direction to offset the shadow. The possible values for this attribute are defined by the ST_PositiveFixedAngle simple type (§5.1.12.44).
dist (Shadow Offset Distance)	Specifies the how far to offset the shadow. The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).
kx (Horizontal Skew)	Specifies the horizontal skew angle. The possible values for this attribute are defined by the ST_FixedAngle simple type (§5.1.12.21).
ky (Vertical Skew)	Specifies the vertical skew angle. The possible values for this attribute are defined by the ST_FixedAngle simple type (§5.1.12.21).
rotWithShape (Rotate With Shape)	Specifies whether the shadow should rotate with the shape if the shape is rotated. The possible values for this attribute are defined by the XML Schema boolean datatype.
sx (Horizontal Scaling Factor)	Specifies the horizontal scaling factor; negative scaling causes a flip. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
sy (Vertical Scaling	Specifies the vertical scaling factor; negative scaling causes a flip.

Attributes	Description
Factor)	The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OuterShadowEffect">
   <sequence>
      <group ref="EG_ColorChoice" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="blurRad" type="ST_PositiveCoordinate" use="optional" default="0"/>
   <attribute name="dist" type="ST_PositiveCoordinate" use="optional" default="0"/>
   <attribute name="dir" type="ST_PositiveFixedAngle" use="optional" default="0"/>
   <attribute name="sx" type="ST_Percentage" use="optional" default="100000"/>
   <attribute name="sy" type="ST_Percentage" use="optional" default="100000"/>
   <attribute name="kx" type="ST_FixedAngle" use="optional" default="0"/>
   <attribute name="ky" type="ST_FixedAngle" use="optional" default="0"/>
   <attribute name="algn" type="ST_RectAlignment" use="optional" default="b"/>
   <attribute name="rotWithShape" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.1.10.46 path (Path Gradient)

This element defines that a gradient fill will follow a path vs. a linear line.

[*Example*:

The examples above illustrate gradient fills following a circular, rectangular or shape path.

end example]

Parent Elements
gradFill (§5.1.10.33)

Child Elements	Subclause
fillToRect (Fill To Rectangle)	§5.1.10.31

Attributes	Description
path (Gradient Fill Path)	Specifies the shape of the path to follow. The possible values for this attribute are defined by the ST_PathShadeType simple type (§5.1.12.39).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PathShadeProperties">
    <sequence>
        <element name="fillToRect" type="CT_RelativeRect" minOccurs="0" maxOccurs="1"/>
    </sequence>
    <attribute name="path" type="ST_PathShadeType" use="optional"/>
</complexType>
```

5.1.10.47 pattFill (Pattern Fill)

This element specifies a pattern fill. A repeated pattern is used to fill the object.

Parent Elements
bg (§5.9.3.1); bgFillStyleLst (§5.1.4.1.7); bgPr (§4.4.1.2); defRPr (§5.1.5.3.2); endParaRPr (§5.1.5.2.3); fill (§5.1.10.28); fill (§5.1.4.2.9); fillOverlay (§5.1.10.29); fillStyleLst (§5.1.4.1.13); grpSpPr (§5.8.2.14); grpSpPr (§4.4.1.20); grpSpPr (§5.1.2.1.22); grpSpPr (§5.6.2.17); ln (§5.1.2.1.24); lnB (§5.1.6.3); lnBlToTr (§5.1.6.4); lnL (§5.1.6.5); lnR (§5.1.6.6); lnT (§5.1.6.7); lnTlToBr (§5.1.6.8); rPr (§5.1.5.3.9); spPr (§5.6.2.29); spPr (§5.7.2.198); spPr (§5.8.2.23); spPr (§5.1.2.1.35); spPr (§4.4.1.41); spPr (§5.9.3.7); spPr (§5.2.2.6); tblPr (§5.1.6.13); tcPr (§5.1.6.15); uFill (§5.1.5.3.12); uLn (§5.1.5.3.14)

Child Elements	Subclause
bgClr (Background color)	§5.1.10.10
fgClr (Foreground color)	§5.1.10.27

Attributes	Description
prst (Preset Pattern)	Specifies one of a set of preset patterns to fill the object. The possible values for this attribute are defined by the ST_PresetPatternVal simple type (§5.1.12.51).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PatternFillProperties">
    <sequence>
        <element name="fgClr" type="CT_Color" minOccurs="0" maxOccurs="1"/>
        <element name="bgClr" type="CT_Color" minOccurs="0" maxOccurs="1"/>
    </sequence>
    <attribute name="prst" type="ST_PresetPatternVal" use="optional"/>
</complexType>
```

5.1.10.48 prstDash (Preset Dash)

This element specifies that a preset line dashing scheme should be used.

Parent Elements
ln (§5.1.2.1.24); lnB (§5.1.6.3); lnBlToTr (§5.1.6.4); lnL (§5.1.6.5); lnR (§5.1.6.6); lnT (§5.1.6.7); lnTlToBr

Parent Elements
(§5.1.6.8); uLn (§5.1.5.3.14)

Attributes	Description
val (Value)	Specifies which preset dashing scheme is to be used. The possible values for this attribute are defined by the ST_PresetLineDashVal simple type (§5.1.12.49).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PresetLineDashProperties">
   <attribute name="val" type="ST_PresetLineDashVal" use="optional"/>
</complexType>
```

5.1.10.49 prstShdw (Preset Shadow)

This element specifies that a preset shadow is to be used. Each preset shadow is equivalent to a specific outer shadow effect. For each preset shadow, the color element, direction attribute, and distance attribute represent the color, direction, and distance parameters of the corresponding outer shadow. Additionally, the rotateWithShape attribute of corresponding outer shadow is always false. Other non-default parameters of the outer shadow are dependent on the prst attribute.

Parent Elements
cont (§5.1.10.20); effectDag (§5.1.10.25); effectLst (§5.1.10.26)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

Attributes	Description
dir (Direction)	Specifies the direction to offset the shadow. The possible values for this attribute are defined by the ST_PositiveFixedAngle simple type (§5.1.12.44).
dist (Distance)	Specifies how far to offset the shadow.

Attributes	Description
	The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).
prst (Preset Shadow)	Specifies which preset shadow to use. The possible values for this attribute are defined by the ST_PresetShadowVal simple type (§5.1.12.52).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PresetShadowEffect">
   <sequence>
      <group ref="EG_ColorChoice" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="prst" type="ST_PresetShadowVal" use="required"/>
   <attribute name="dist" type="ST_PositiveCoordinate" use="optional" default="0"/>
   <attribute name="dir" type="ST_PositiveFixedAngle" use="optional" default="0"/>
</complexType>
```

5.1.10.50 reflection (Reflection Effect)

This element specifies a reflection effect.

[*Example*:

end example]

Parent Elements
cont (§5.1.10.20); effectDag (§5.1.10.25); effectLst (§5.1.10.26)

Attributes	Description
algn (Shadow Alignment)	Specifies shadow alignment. The possible values for this attribute are defined by the ST_RectAlignment simple type

Attributes	Description
	(§5.1.12.53).
blurRad (Blur Radius)	Specifies the blur radius. The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).
dir (Direction)	Specifies the direction of the alpha gradient ramp relative to the shape itself. The possible values for this attribute are defined by the ST_PositiveFixedAngle simple type (§5.1.12.44).
dist (Distance)	Specifies how far to distance the shadow. The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).
endA (End Alpha)	Specifies the ending reflection opacity. The possible values for this attribute are defined by the ST_PositiveFixedPercentage simple type (§5.1.12.45).
endPos (End Position)	Specifies the end position (along the alpha gradient ramp) of the end alpha value. The possible values for this attribute are defined by the ST_PositiveFixedPercentage simple type (§5.1.12.45).
fadeDir (Fade Direction)	Specifies the direction to offset the reflection. The possible values for this attribute are defined by the ST_PositiveFixedAngle simple type (§5.1.12.44).
kx (Horizontal Skew)	Specifies the horizontal skew angle. The possible values for this attribute are defined by the ST_FixedAngle simple type (§5.1.12.21).
ky (Vertical Skew)	Specifies the vertical skew angle. The possible values for this attribute are defined by the ST_FixedAngle simple type (§5.1.12.21).
rotWithShape (Rotate With Shape)	Specifies if the reflection should rotate with the shape. The possible values for this attribute are defined by the XML Schema boolean datatype.
stA (Start Opacity)	starting reflection opacity. The possible values for this attribute are defined by the ST_PositiveFixedPercentage simple type (§5.1.12.45).
stPos (Start Position)	Specifies the start position (along the alpha gradient ramp) of the start alpha value.

Attributes	Description
	The possible values for this attribute are defined by the ST_PositiveFixedPercentage simple type (§5.1.12.45).
sx (Horizontal Ratio)	Specifies the horizontal scaling factor. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
sy (Vertical Ratio)	Specifies the vertical scaling factor. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ReflectionEffect">
    <attribute name="blurRad" type="ST_PositiveCoordinate" use="optional" default="0"/>
    <attribute name="stA" type="ST_PositiveFixedPercentage" use="optional" default="100000"/>
    <attribute name="stPos" type="ST_PositiveFixedPercentage" use="optional" default="0"/>
    <attribute name="endA" type="ST_PositiveFixedPercentage" use="optional" default="0"/>
    <attribute name="endPos" type="ST_PositiveFixedPercentage" use="optional" default="100000"/>
    <attribute name="dist" type="ST_PositiveCoordinate" use="optional" default="0"/>
    <attribute name="dir" type="ST_PositiveFixedAngle" use="optional" default="0"/>
    <attribute name="fadeDir" type="ST_PositiveFixedAngle" use="optional" default="5400000"/>
    <attribute name="sx" type="ST_Percentage" use="optional" default="100000"/>
    <attribute name="sy" type="ST_Percentage" use="optional" default="100000"/>
    <attribute name="kx" type="ST_FixedAngle" use="optional" default="0"/>
    <attribute name="ky" type="ST_FixedAngle" use="optional" default="0"/>
    <attribute name="algn" type="ST_RectAlignment" use="optional" default="b"/>
    <attribute name="rotWithShape" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.1.10.51 relOff (Relative Offset Effect)

This element specifies a relative offset effect. Sets up a new origin by offsetting relative to the size of the previous effect.

Parent Elements
cont (§5.1.10.20); effectDag (§5.1.10.25)

Attributes	Description
tx (Offset X)	Specifies the X offset. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
ty (Offset Y)	Specifies the Y offset. The possible values for this attribute are defined by the ST_Percentage simple type

Attributes	Description
	(§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RelativeOffsetEffect">
   <attribute name="tx" type="ST_Percentage" use="optional" default="0"/>
   <attribute name="ty" type="ST_Percentage" use="optional" default="0"/>
</complexType>
```

5.1.10.52 round (Round Line Join)

This element specifies that lines joined together will have a round join.

[*Example*:

end example]

Parent Elements
ln (§5.1.2.1.24); lnB (§5.1.6.3); lnBlToTr (§5.1.6.4); lnL (§5.1.6.5); lnR (§5.1.6.6); lnT (§5.1.6.7); lnTlToBr (§5.1.6.8); uLn (§5.1.5.3.14)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LineJoinRound"/>
```

5.1.10.53 softEdge (Soft Edge Effect)

This element specifies a soft edge effect. The edges of the shape are blurred, while the fill is not affected.

Parent Elements
cont (§5.1.10.20); effectDag (§5.1.10.25); effectLst (§5.1.10.26)

Attributes	Description
rad (Radius)	Specifies the radius of blur to apply to the edges. The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SoftEdgesEffect">
   <attribute name="rad" type="ST_PositiveCoordinate" use="required"/>
</complexType>
```

5.1.10.54 solidFill (Solid Fill)

This element specifies a solid color fill. The shape is filled entirely with the specified color.

Parent Elements
bg (§5.9.3.1); bgFillStyleLst (§5.1.4.1.7); bgPr (§4.4.1.2); defRPr (§5.1.5.3.2); endParaRPr (§5.1.5.2.3); fill (§5.1.10.28); fill (§5.1.4.2.9); fillOverlay (§5.1.10.29); fillStyleLst (§5.1.4.1.13); grpSpPr (§5.8.2.14); grpSpPr (§4.4.1.20); grpSpPr (§5.1.2.1.22); grpSpPr (§5.6.2.17); ln (§5.1.2.1.24); lnB (§5.1.6.3); lnBlToTr (§5.1.6.4); lnL (§5.1.6.5); lnR (§5.1.6.6); lnT (§5.1.6.7); lnTlToBr (§5.1.6.8); rPr (§5.1.5.3.9); spPr (§5.6.2.29); spPr (§5.7.2.198); spPr (§5.8.2.23); spPr (§5.1.2.1.35); spPr (§4.4.1.41); spPr (§5.9.3.7); spPr (§5.2.2.6); tblPr (§5.1.6.13); tcPr (§5.1.6.15); uFill (§5.1.5.3.12); uLn (§5.1.5.3.14)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SolidColorFillProperties">
   <sequence>
      <group ref="EG_ColorChoice" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.10.55 srcRect (Source Rectangle)

This element specifies the portion of the blip used for the fill.

Each edge of the source rectangle is defined by a percentage offset from the corresponding edge of the bounding box. A positive percentage specifies an inset, while a negative percentage specifies an outset. For example, a left offset of 25% specifies that the left edge of the source rectangle is located to the right of the bounding box's left edge by an amount equal to 25% of the bounding box's width.

Parent Elements
blipFill (§5.8.2.2); blipFill (§5.1.10.14); blipFill (§5.6.2.2); blipFill (§5.2.2.1); blipFill (§4.4.1.4)

Attributes	Description
b (Bottom Offset)	Specifies the bottom edge of the rectangle. The possible values for this attribute are defined by the ST_Percentage simple type

Attributes	Description
	(§5.1.12.41).
l (Left Offset)	Specifies the left edge of the rectangle. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
r (Right Offset)	Specifies the right edge of the rectangle. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
t (Top Offset)	Specifies the top edge of the rectangle. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RelativeRect">
   <attribute name="l" type="ST_Percentage" use="optional" default="0"/>
   <attribute name="t" type="ST_Percentage" use="optional" default="0"/>
   <attribute name="r" type="ST_Percentage" use="optional" default="0"/>
   <attribute name="b" type="ST_Percentage" use="optional" default="0"/>
</complexType>
```

5.1.10.56 stretch (Stretch)

This element specifies that a BLIP should be stretched to fill the target rectangle. The other option is a tile where a BLIP is tiled to fill the available area.

Parent Elements
blipFill (§5.8.2.2); blipFill (§5.1.10.14); blipFill (§5.6.2.2); blipFill (§5.2.2.1); blipFill (§4.4.1.4)

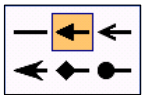
Child Elements	Subclause
fillRect (Fill Rectangle)	§5.1.10.30

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_StretchInfoProperties">
   <sequence>
      <element name="fillRect" type="CT_RelativeRect" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.10.57 tailEnd (Tail line end style)

This element specifies decorations which can be added to the tail of a line.



Parent Elements
ln (§5.1.2.1.24); lnB (§5.1.6.3); lnBlToTr (§5.1.6.4); lnL (§5.1.6.5); lnR (§5.1.6.6); lnT (§5.1.6.7); lnTlToBr (§5.1.6.8); uLn (§5.1.5.3.14)

Attributes	Description
len (Length of Head/End)	Specifies the line end length in relation to the line width. The possible values for this attribute are defined by the ST_LineEndLength simple type (§5.1.12.32).
type (Line Head/End Type)	Specifies the line end decoration, such as a triangle or arrowhead. The possible values for this attribute are defined by the ST_LineEndType simple type (§5.1.12.33).
w (Width of Head/End)	Specifies the line end width in relation to the line width. The possible values for this attribute are defined by the ST_LineEndWidth simple type (§5.1.12.34).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LineEndProperties">
   <attribute name="type" type="ST_LineEndType" use="optional"/>
   <attribute name="w" type="ST_LineEndWidth" use="optional"/>
   <attribute name="len" type="ST_LineEndLength" use="optional"/>
</complexType>
```

5.1.10.58 tile (Tile)

This element specifies that a BLIP should be tiled to fill the available space. This element defines a "tile" rectangle within the bounding box. The image is encompassed within the tile rectangle, and the tile rectangle is tiled across the bounding box to fill the entire area.

[*Example*:

The following is a fill with the flip attribute set to "x". The black interior rectangle indicates the tile rectangle. Notice that the adjacent rectangle to the right in the tile has been flipped along the x-axis.

end example]

Parent Elements
blipFill (§5.8.2.2); blipFill (§5.1.10.14); blipFill (§5.6.2.2); blipFill (§5.2.2.1); blipFill (§4.4.1.4)

Attributes	Description
algn (Alignment)	Specifies where to align the first tile with respect to the shape. Alignment happens after the scaling, but before the additional offset. The possible values for this attribute are defined by the ST_RectAlignment simple type (§5.1.12.53).
flip (Tile Flipping)	Specifies the direction(s) in which to flip the source image while tiling. Images may be flipped horizontally, vertically, or in both directions to fill the entire region. The possible values for this attribute are defined by the ST_TileFlipMode simple type (§5.1.12.86).
sx (Horizontal Ratio)	Specifies the amount to horizontally scale the srcRect. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
sy (Vertical Ratio)	Specifies the amount to vertically scale the srcRect. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
tx (Horizontal Offset)	Specifies additional horizontal offset after alignment. The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).
ty (Vertical Offset)	Specifies additional vertical offset after alignment. The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TileInfoProperties">
   <attribute name="tx" type="ST_Coordinate" use="optional"/>
   <attribute name="ty" type="ST_Coordinate" use="optional"/>
   <attribute name="sx" type="ST_Percentage" use="optional"/>
   <attribute name="sy" type="ST_Percentage" use="optional"/>
   <attribute name="flip" type="ST_TileFlipMode" use="optional"/>
   <attribute name="algn" type="ST_RectAlignment" use="optional"/>
</complexType>
```

5.1.10.59 tileRect (Tile Rectangle)

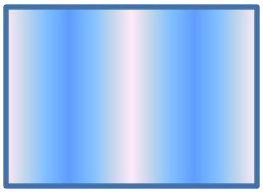
This element specifies a rectangular region of the shape to which the gradient is applied. This region is then tiled across the remaining area of the shape to complete the fill. The tile rectangle is defined by percentage offsets from the sides of the shape's bounding box.

Each edge of the tile rectangle is defined by a percentage offset from the corresponding edge of the bounding box. A positive percentage specifies an inset, while a negative percentage specifies an outset. For example, a left offset of 25% specifies that the left edge of the tile rectangle is located to the right of the bounding box's left edge by an amount equal to 25% of the bounding box's width.

[*Example*:

The image above depicts a horizontal gradient with no tileRect element.

The image above depicts the same gradient with a tileRect element specifying l="50000" (50%). The right half of the shape is the tile to which the gradient is applied, and the left half of the shape contains a tiled copy of that gradient fill.



The image above depicts the same gradient with a tileRect element specifying l="75000" (75%). The rightmost 25% of the shape contains the tile rectangle to which the gradient is applied. This gradient is tiled three times to

cover the leftmost 75% of the shape. Note that the tile rectangle is flipped horizontally when covering the shape.

end example]

Parent Elements
gradFill (§5.1.10.33)

Attributes	Description
b (Bottom Offset)	Specifies the bottom edge of the rectangle. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
l (Left Offset)	Specifies the left edge of the rectangle. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
r (Right Offset)	Specifies the right edge of the rectangle. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
t (Top Offset)	Specifies the top edge of the rectangle. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RelativeRect">
   <attribute name="l" type="ST_Percentage" use="optional" default="0"/>
   <attribute name="t" type="ST_Percentage" use="optional" default="0"/>
   <attribute name="r" type="ST_Percentage" use="optional" default="0"/>
   <attribute name="b" type="ST_Percentage" use="optional" default="0"/>
</complexType>
```

5.1.10.60 tint (Tint Effect)

This element specifies a tint effect. Shifts effect color values towards/away from hue by the specified amount.

Parent Elements
blip (§5.1.10.13); cont (§5.1.10.20); effectDag (§5.1.10.25)

Attributes	Description
amt (Amount)	Specifies by how much the color value is shifted.

Attributes	Description
	The possible values for this attribute are defined by the ST_FixedPercentage simple type (§5.1.12.22).
hue (Hue)	Specifies the hue towards which to tint. The possible values for this attribute are defined by the ST_PositiveFixedAngle simple type (§5.1.12.44).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TintEffect">
   <attribute name="hue" type="ST_PositiveFixedAngle" use="optional" default="0"/>
   <attribute name="amt" type="ST_FixedPercentage" use="optional" default="0"/>
</complexType>
```

5.1.10.61 xfrm (Transform Effect)

This element specifies a transform effect. The transform is applied to each point in the shape's geometry using the following matrix:

$$\begin{bmatrix} sx & \tan(kx) & tx \\ \tan(ky) & sy & ty \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Parent Elements
cont (§5.1.10.20); effectDag (§5.1.10.25)

Attributes	Description
kx (Horizontal Skew)	Specifies the horizontal skew angle, defined as the agnle between the top-left corner and bottom-left corner of the object's original bounding box. If positive, the bottom edge of the shape will be positioned to the right relative to the top edge. The possible values for this attribute are defined by the ST_FixedAngle simple type (§5.1.12.21).
ky (Vertical Skew)	Specifies the vertical skew angle, defined as the angle between the top-left corner and top-right corner of the object's original bounding box. If positive, the right edge of the object will be positioned lower relative to the left edge. The possible values for this attribute are defined by the ST_FixedAngle simple type (§5.1.12.21).
sx (Horizontal Ratio)	Specifies a percentage by which to horizontally scale the object. The possible values for this attribute are defined by the ST_Percentage simple type

Attributes	Description
	(§5.1.12.41).
sy (Vertical Ratio)	Specifies a percentage by which to vertically scale the object. The possible values for this attribute are defined by the ST_Percentage simple type (§5.1.12.41).
tx (Horizontal Shift)	Specifies an amount by which to shift the object along the x-axis. The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).
ty (Vertical Shift)	Specifies an amount by which to shift the object along the y-axis. The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TransformEffect">
   <attribute name="sx" type="ST_Percentage" use="optional" default="100000"/>
   <attribute name="sy" type="ST_Percentage" use="optional" default="100000"/>
   <attribute name="kx" type="ST_FixedAngle" use="optional" default="0"/>
   <attribute name="ky" type="ST_FixedAngle" use="optional" default="0"/>
   <attribute name="tx" type="ST_Coordinate" use="optional" default="0"/>
   <attribute name="ty" type="ST_Coordinate" use="optional" default="0"/>
</complexType>
```

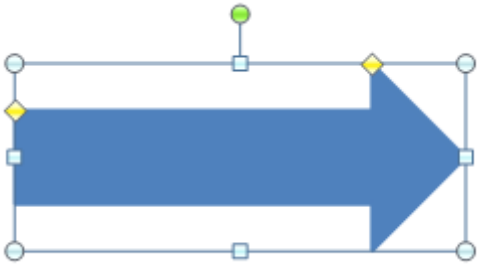
5.1.11 Shape Definitions and Attributes

The Shape Definitions and Attributes portion of the DrawingML framework deals with all geometric properties for shapes within a document. This includes both preset geometries that publicly are interpreted by the generating application and custom geometries that have their points and curves explicitly specified. In addition to the underlying geometry of the shape there are also other coordinate-based properties for each shape that this framework describes.

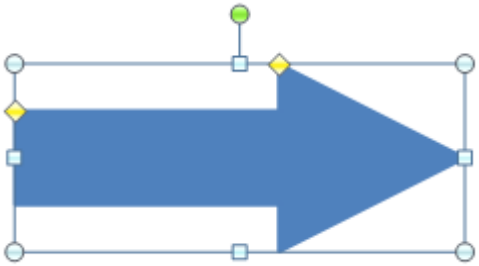
5.1.11.1 ahLst (List of Shape Adjust Handles)

This element specifies the adjust handles that will be applied to a custom geometry. These adjust handles will specify points within the geometric shape that can be used to perform certain transform operations on the shape.

[*Example*: Consider the scenario where a custom geometry, an arrow in this case, has been drawn and adjust handles have been placed at the top left corner of both the arrow head and arrow body. The user interface can then be made to transform only certain parts of the shape by using the corresponding adjust handle.



For instance if the user wished to change only the width of the arrow head then they would use the adjust handle located on the top left of the arrow head. The result of adjusting this will transform the shape as shown below.



end example]

Parent Elements
custGeom (§5.1.11.8)

Child Elements	Subclause
ahPolar (Polar Adjust Handle)	§5.1.11.2
ahXY (XY Adjust Handle)	§5.1.11.3

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AdjustHandleList">
   <choice minOccurs="0" maxOccurs="unbounded">
      <element name="ahXY" type="CT_XYAdjustHandle" minOccurs="1" maxOccurs="1"/>
      <element name="ahPolar" type="CT_PolarAdjustHandle" minOccurs="1" maxOccurs="1"/>
   </choice>
</complexType>
```

5.1.11.2 ahPolar (Polar Adjust Handle)

This element specifies a polar adjust handle for a custom shape. The position of this adjust handle will be specified by the corresponding pos child element. The allowed adjustment of this adjust handle are specified via it's min and max type attributes. Based on the adjustment of this adjust handle certain corresponding guides will be updated to contain these values.

Parent Elements
ahLst (§5.1.11.1)

Child Elements	Subclause
pos (Shape Position Coordinate)	§5.1.11.17

Attributes	Description
gdRefAng (Angle Adjustment Guide)	Specifies the name of the guide that will be updated with the adjustment angle from this adjust handle. The possible values for this attribute are defined by the ST_GeomGuideName simple type (§5.1.12.26).
gdRefR (Radial Adjustment Guide)	Specifies the name of the guide that will be updated with the adjustment radius from this adjust handle. The possible values for this attribute are defined by the ST_GeomGuideName simple type (§5.1.12.26).
maxAng (Maximum Angle Adjustment)	Specifies the maximum angle position that is allowed for this adjustment handle. If this attribute is omitted, then it will be assumed that this adjust handle cannot move angularly. That is the maxAng and minAng will be equal. The possible values for this attribute are defined by the ST_AdjAngle simple type (§5.1.12.1).
maxR (Maximum Radial Adjustment)	Specifies the maximum radial position that is allowed for this adjustment handle. If this attribute is omitted, then it will be assumed that this adjust handle cannot move radially. That is the maxR and minR will be equal. The possible values for this attribute are defined by the ST_AdjCoordinate simple type (§5.1.12.2).
minAng (Minimum Angle Adjustment)	Specifies the minimum angle position that is allowed for this adjustment handle. If this attribute is omitted, then it will be assumed that this adjust handle cannot move angularly. That is the maxAng and minAng will be equal. The possible values for this attribute are defined by the ST_AdjAngle simple type (§5.1.12.1).
minR (Minimum Radial Adjustment)	Specifies the minimum radial position that is allowed for this adjustment handle. If this attribute is omitted, then it will be assumed that this adjust handle cannot move radially. That is the maxR and minR will be equal. The possible values for this attribute are defined by the ST_AdjCoordinate simple type (§5.1.12.2).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PolarAdjustHandle">
   <sequence>
      <element name="pos" type="CT_AdjPoint2D" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="gdRefR" type="ST_GeomGuideName" use="optional"/>
   <attribute name="minR" type="ST_AdjCoordinate" use="optional"/>
   <attribute name="maxR" type="ST_AdjCoordinate" use="optional"/>
   <attribute name="gdRefAng" type="ST_GeomGuideName" use="optional"/>
   <attribute name="minAng" type="ST_AdjAngle" use="optional"/>
   <attribute name="maxAng" type="ST_AdjAngle" use="optional"/>
</complexType>
```

5.1.11.3 ahXY (XY Adjust Handle)

This element specifies an XY-based adjust handle for a custom shape. The position of this adjust handle will be specified by the corresponding pos child element. The allowed adjustment of this adjust handle are specified via it's min and max type attributes. Based on the adjustment of this adjust handle certain corresponding guides will be updated to contain these values.

Parent Elements
ahLst (§5.1.11.1)

Child Elements	Subclause
pos (Shape Position Coordinate)	§5.1.11.17

Attributes	Description
gdRefX (Horizontal Adjustment Guide)	Specifies the name of the guide that will be updated with the adjustment x position from this adjust handle. The possible values for this attribute are defined by the ST_GeomGuideName simple type (§5.1.12.26).
gdRefY (Vertical Adjustment Guide)	Specifies the name of the guide that will be updated with the adjustment y position from this adjust handle. The possible values for this attribute are defined by the ST_GeomGuideName simple type (§5.1.12.26).
maxX (Maximum Horizontal Adjustment)	Specifies the maximum horizontal position that is allowed for this adjustment handle. If this attribute is omitted, then it will be assumed that this adjust handle cannot move in the x direction. That is the maxX and minX will be equal. The possible values for this attribute are defined by the ST_AdjCoordinate simple type (§5.1.12.2).
maxY (Maximum	Specifies the maximum vertical position that is allowed for this adjustment handle. If this

Attributes	Description
Vertical Adjustment)	attribute is omitted, then it will be assumed that this adjust handle cannot move in the y direction. That is the maxY and minY will be equal. The possible values for this attribute are defined by the ST_AdjCoordinate simple type (§5.1.12.2).
minX (Minimum Horizontal Adjustment)	Specifies the minimum horizontal position that is allowed for this adjustment handle. If this attribute is omitted, then it will be assumed that this adjust handle cannot move in the x direction. That is the maxX and minX will be equal. The possible values for this attribute are defined by the ST_AdjCoordinate simple type (§5.1.12.2).
minY (Minimum Vertical Adjustment)	Specifies the minimum vertical position that is allowed for this adjustment handle. If this attribute is omitted, then it will be assumed that this adjust handle cannot move in the y direction. That is the maxY and minY will be equal. The possible values for this attribute are defined by the ST_AdjCoordinate simple type (§5.1.12.2).

The following XML Schema fragment defines the contents of this element:

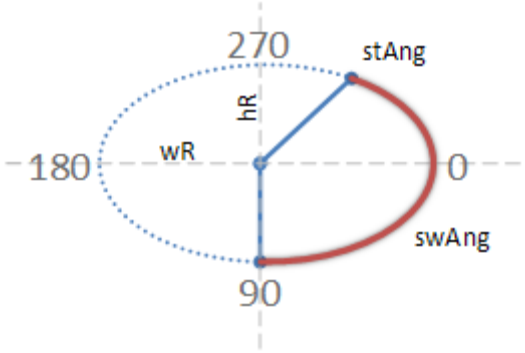
```
<complexType name="CT_XYAdjustHandle">
   <sequence>
      <element name="pos" type="CT_AdjPoint2D" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="gdRefX" type="ST_GeomGuideName" use="optional"/>
   <attribute name="minX" type="ST_AdjCoordinate" use="optional"/>
   <attribute name="maxX" type="ST_AdjCoordinate" use="optional"/>
   <attribute name="gdRefY" type="ST_GeomGuideName" use="optional"/>
   <attribute name="minY" type="ST_AdjCoordinate" use="optional"/>
   <attribute name="maxY" type="ST_AdjCoordinate" use="optional"/>
</complexType>
```

5.1.11.4 arcTo (Draw Arc To)

This element specifies the existence of an arc within a shape path. It draws an arc with the specified parameters from the current pen position to the new point specified. An arc is a line that is bent based on the shape of a supposed circle. The length of this arc is determined by specifying both a start angle and an ending angle that act together to effectively specify an end point for the arc.

[*Example*: The diagram shown below represents a single arc that has a start angle of 300 degrees and a swing angle of 150 degrees. This arc will be drawn using the supposed circle that is described using the hR and wR attributes as shown below. The degrees by which the stAng must abide is shown along the circumference of the circle. These degrees are to be specified in 60,000ths of a degree. If this arc were part of a shape the start angle point along the circle would be the starting point along the path and the ending point would be the ending of the angle swing along this supposed circle. That is any shape geometry coming before this arc in the shape path

would be joined with the upper point of this arc and consequently any geometry coming after this arc in the path would be joined with the lower point of this arc.



end example]

Parent Elements
path (§5.1.11.15)

Attributes	Description
hR (Shape Arc Height Radius)	This attribute will specify the height radius of the supposed circle being used to draw the arc. This will give the circle a total height of (2 * hR). This total height could also be called it's vertical diameter as it is the diameter for the y axis only. The possible values for this attribute are defined by the ST_AdjCoordinate simple type (§5.1.12.2).
stAng (Shape Arc Start Angle)	Specifies the start angle for an arc. This angle will specify what angle along the supposed circle path will be used as the start position for drawing the arc. This start angle will be locked to the last known pen position in the shape path. Thus guaranteeing a continuos shape path. The possible values for this attribute are defined by the ST_AdjAngle simple type (§5.1.12.1).
swAng (Shape Arc Swing Angle)	Specifies the swing angle for an arc. This angle will specify how far angle-wise along the supposed cicle path the arc will be extended. The extension from the start angle will always be in the clockwise direction around the supposed circle. The possible values for this attribute are defined by the ST_AdjAngle simple type (§5.1.12.1).
wR (Shape Arc Width Radius)	This attribute will specify the width radius of the supposed circle being used to draw the arc. This will give the circle a total width of (2 * wR). This total width could also be called it's horizontal diameter as it is the diameter for the x axis only. The possible values for this attribute are defined by the ST_AdjCoordinate simple type

Attributes	Description
	(§5.1.12.2).

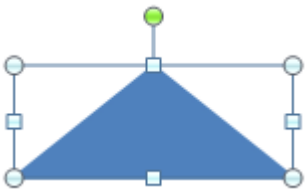
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Path2DArcTo">
   <attribute name="wR" type="ST_AdjCoordinate" use="required"/>
   <attribute name="hR" type="ST_AdjCoordinate" use="required"/>
   <attribute name="stAng" type="ST_AdjAngle" use="required"/>
   <attribute name="swAng" type="ST_AdjAngle" use="required"/>
</complexType>
```

5.1.11.5 avLst (List of Shape Adjust Values)

This element specifies the adjust values that will be applied to the specified shape. An adjust value is simply a guide that has a value based formula specified. That is, no calculation takes place for an adjust value guide. Instead, this guide specifies a parameter value that is used for calculations within the shape guides.

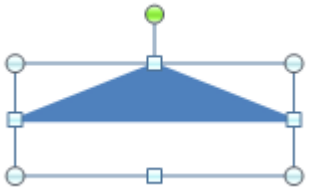
[*Example*: Consider the case where the user would like to specify a triangle with it's bottom edge defined not by static points but by using a varying parameter, namely an adjust value. Consider the diagrams and DrawingML shown below. This first triangle has been drawn with a bottom edge that is equal to the height, namely 2. Thus we see in the figure below that the bottom of the triangle matches the bottom of the shape bounding box.



```
<a:xfrm>
  <a:off x="3200400" y="1600200"/>
  <a:ext cx="1705233" cy="679622"/>
</a:xfrm>
<a:custGeom>
  <a:avLst>
    <a:gd name="myGuide" fmla="val 2"/>
  </a:avLst>
  <a:gdLst/>
  <a:ahLst/>
  <a:cxnLst/>
  <a:rect l="0" t="0" r="0" b="0"/>
  <a:pathLst>
    <a:path w="2" h="2">
      <a:moveTo>
        <a:pt x="0" y="myGuide"/>
      </a:moveTo>
```

```
        <a:lnTo>
          <a:pt x="2" y="myGuide"/>
        </a:lnTo>
        <a:lnTo>
          <a:pt x="1" y="0"/>
        </a:lnTo>
        <a:close/>
      </a:path>
    </a:pathLst>
  </a:custGeom>
```

If however we change the adjust value to half that, namely 1. Then we see the entire bottom edge of the triangle move to now be placed along the vertical midpoint within the shape bounding box. This is because both of the bottom points in this triangle depend on this adjust value for their coordinate positions. The triangle and corresponding DrawingML shown below illustrate this point.



```
  <a:avLst>
    <a:gd name="myGuide" fmla="val 1"/>
  </a:avLst>
```

end example]

Parent Elements
custGeom (§5.1.11.8); prstGeom (§5.1.11.18); prstTxWarp (§5.1.11.19)

Child Elements	Subclause
gd (Shape Guide)	§5.1.11.11

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GeomGuideList">
   <sequence>
      <element name="gd" type="CT_GeomGuide" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.1.11.6 close (Close Shape Path)

This element specifies the ending of a series of lines and curves in the creation path of a custom geometric shape. When this element is encountered, the generating application should consider the corresponding path closed. That is, any further lines or curves that follow this element should be ignored.

[*Note*: It is valid to have a path be specified and not closed. A path such as this cannot however have any fill associated with it as it has not been considered a closed geometric path.

[*Example*: Consider the following DrawingML.

```
<a:custGeom>
  <a:pathLst>
    <a:path w="2824222" h="590309">
      <a:moveTo>
        <a:pt x="0" y="428263"/>
      </a:moveTo>
      <a:lnTo>
        <a:pt x="1620455" y="590309"/>
      </a:lnTo>
      <a:lnTo>
        <a:pt x="2824222" y="173620"/>
      </a:lnTo>
      <a:lnTo>
        <a:pt x="1562582" y="0"/>
      </a:lnTo>
      <a:close/>
    </a:path>
  </a:pathLst>
</a:custGeom>
```

In the above example there is specified a four sided geometric shape that has all straight sides. While we only see three lines being drawn via the lnTo element there are actually four sides because the last point of (x=1562585, y=0) is connected to the first point in the creation path via a lnTo element. *end example*]

[*Note*: When the last point in the creation path does not meet with the first point in the creation path the generating application should connect the last point with the first via a straight line, thus creating a closed shape geometry. *end note*]

Parent Elements
path (§5.1.11.15)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Path2DClose"/>
```

5.1.11.7 cubicBezTo (Draw Cubic Bezier Curve To)

This element specifies to draw a cubic bezier curve along the specified points. To specify a cubic bezier curve there needs to be 3 points specified. The first two are control points used in the cubic bezier calculation and the last is the ending point for the curve. The coordinate system used for this type of curve is the path coordinate system as this element is path specific.

Parent Elements
path (§5.1.11.15)

Child Elements	Subclause
pt (Shape Path Point)	§5.1.11.20

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Path2DCubicBezierTo">
   <sequence>
      <element name="pt" type="CT_AdjPoint2D" minOccurs="3" maxOccurs="3"/>
   </sequence>
</complexType>
```

5.1.11.8 custGeom (Custom Geometry)

This element specifies the existence of a custom geometric shape. This shape will consist of a series of lines and curves described within a creation path. In addition to this there may also be adjust values, guides, adjust handles, connection sites and an inscribed rectangle specified for this custom geometric shape.

[*Example*: Consider the scenario when a preset geometry does not accurately depict what must be displayed in the document. For this a custom geometry can be used to define most any 2-dimensional geometric shape. Shown below is an example of such a custom geometry.

```
<a:custGeom>
  <a:avLst/>
  <a:gdLst/>
  <a:ahLst/>
  <a:cxnLst/>
  <a:rect l="0" t="0" r="0" b="0"/>
  <a:pathLst>
    <a:path w="2650602" h="1261641">
      <a:moveTo>
        <a:pt x="0" y="1261641"/>
      </a:moveTo>
      <a:lnTo>
        <a:pt x="2650602" y="1261641"/>
      </a:lnTo>
```

```
      <a:lnTo>
        <a:pt x="1226916" y="0"/>
      </a:lnTo>
      <a:close/>
    </a:path>
  </a:pathLst>
</a:custGeom>
```

The custom geometry above is drawn by first moving to a specific starting point with the moveTo element. Then a series of lnTo elements in the creation path specify the lines that make up the borders of the shape and finally a close element is used to specify the end of the creation path. The resulting shape is shown above. *end example*]

Parent Elements
spPr (§5.6.2.29); spPr (§5.7.2.198); spPr (§5.8.2.23); spPr (§5.1.2.1.35); spPr (§4.4.1.41); spPr (§5.9.3.7); spPr (§5.2.2.6)

Child Elements	Subclause
ahLst (List of Shape Adjust Handles)	§5.1.11.1
avLst (List of Shape Adjust Values)	§5.1.11.5
cxnLst (List of Shape Connection Sites)	§5.1.11.10
gdLst (List of Shape Guides)	§5.1.11.12
pathLst (List of Shape Paths)	§5.1.11.16
rect (Shape Text Rectangle)	§5.1.11.22

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustomGeometry2D">
   <sequence>
      <element name="avLst" type="CT_GeomGuideList" minOccurs="0" maxOccurs="1"/>
      <element name="gdLst" type="CT_GeomGuideList" minOccurs="0" maxOccurs="1"/>
      <element name="ahLst" type="CT_AdjustHandleList" minOccurs="0" maxOccurs="1"/>
      <element name="cxnLst" type="CT_ConnectionSiteList" minOccurs="0" maxOccurs="1"/>
      <element name="rect" type="CT_GeomRect" minOccurs="0" maxOccurs="1"/>
      <element name="pathLst" type="CT_Path2DList" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.11.9 cxn (Shape Connection Site)

This element specifies the existence of a connection site on a custom shape. A connection site allows a cxnSp to be attached to this shape. This connection will be maintined when the shape is repositioned within the document. It should be noted that this connection is placed within the shape bounding box using the transform coordinate system which is also called the shape coordinate system, as it encompasses the entire shape. The width and height for this coordinate system are specified within the ext transform element.

[*Note*: The transform coordinate system is different from a path coordinate system as it is per shape instead of per path within the shape. *end note*]

[*Example*: Consider the following custom geometry that has two connection sites specified. One connection is located at the bottom left of the shape and the other at the bottom right. The following DrawingML would describe such a custom geometry.

```
<a:xfrm>
  <a:off x="3200400" y="1600200"/>
  <a:ext cx="1705233" cy="679622"/>
</a:xfrm>
<a:custGeom>
  <a:avLst/>
  <a:gdLst/>
  <a:ahLst/>
  <a:cxnLst>
    <a:cxn ang="0">
      <a:pos x="0" y="679622"/>
    </a:cxn>
```

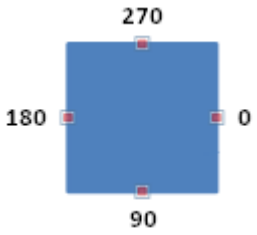
```
      <a:cxn ang="0">
        <a:pos x="1705233" y="679622"/>
      </a:cxn>
    </a:cxnLst>
    <a:rect l="0" t="0" r="0" b="0"/>
    <a:pathLst>
      <a:path w="2" h="2">
        <a:moveTo>
          <a:pt x="0" y="2"/>
        </a:moveTo>
        <a:lnTo>
          <a:pt x="2" y="2"/>
        </a:lnTo>
        <a:lnTo>
          <a:pt x="1" y="0"/>
        </a:lnTo>
        <a:close/>
      </a:path>
    </a:pathLst>
  </a:custGeom>
```

end example]

Parent Elements
cxnLst (§5.1.11.10)

Child Elements	Subclause
pos (Shape Position Coordinate)	§5.1.11.17

Attributes	Description
ang (Connection Site Angle)	Specifies the incoming connector angle. This angle is the angle around the connection site that an incoming connector will try to be routed to. This allows connectors to know where the shape is in relation to the connection site and route connectors so as to avoid any overlap with the shape. [*Example*: Consider a simple square. In order to not have any connectors routed over the shape, the collowing angles would be specified for their respective connection sites.

Attributes	Description
	 *end example*] The possible values for this attribute are defined by the ST_AdjAngle simple type (§5.1.12.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ConnectionSite">
    <sequence>
        <element name="pos" type="CT_AdjPoint2D" minOccurs="1" maxOccurs="1"/>
    </sequence>
    <attribute name="ang" type="ST_AdjAngle" use="required"/>
</complexType>
```

5.1.11.10 cxnLst (List of Shape Connection Sites)

This element specifies all the connection sites that will be used for this shape. A connection site is specified by defining a point within the shape bounding box that can have a cxnSp element attached to it. These connection sites are specified using the shape coordinate system that is specified within the ext transform element.

Parent Elements
custGeom (§5.1.11.8)

Child Elements	Subclause
cxn (Shape Connection Site)	§5.1.11.9

The following XML Schema fragment defines the contents of this element:

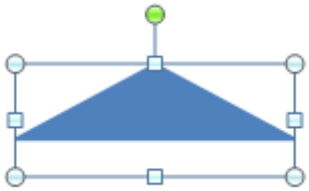
```
<complexType name="CT_ConnectionSiteList">
    <sequence>
        <element name="cxn" type="CT_ConnectionSite" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
</complexType>
```

5.1.11.11 gd (Shape Guide)

This element specifies the precense of a shape guide that will be used to govern the geometry of the specified shape. A shape guide consists of a formula and a name that the result of the formula is assigned to. Recognized formulas are listed with the fmla attribute documentation for this element.

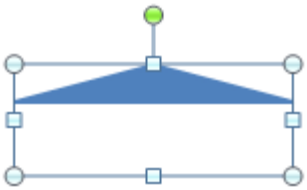
[*Note*: The order in which guides are specified determines the order in which their values will be calculated. For instance it is not possible to specify a guide that uses another guides result when that guide has not yet been calculated. *end note*]

[*Example*: Consider the case where the user would like to specify a triangle with it's bottom edge defined not by static points but by using a varying parameter, namely an guide. Consider the diagrams and DrawingML shown below. This first triangle has been drawn with a bottom edge that is equal to the 2/3 the value of the shape height. Thus we see in the figure below that the triangle appears to occupy 2/3 of the vertical space within the shape bounding box.



```
<a:xfrm>
  <a:off x="3200400" y="1600200"/>
  <a:ext cx="1705233" cy="679622"/>
</a:xfrm>
<a:custGeom>
  <a:avLst/>
  <a:gdLst>
    <a:gd name="myGuide" fmla="*/ h 2 3"/>
  </a:gdLst>
  <a:ahLst/>
  <a:cxnLst/>
  <a:rect l="0" t="0" r="0" b="0"/>
  <a:pathLst>
    <a:path w="1705233" h="679622">
      <a:moveTo>
        <a:pt x="0" y="myGuide"/>
      </a:moveTo>
      <a:lnTo>
        <a:pt x="1705233" y="myGuide"/>
      </a:lnTo>
      <a:lnTo>
        <a:pt x="852616" y="0"/>
      </a:lnTo>
      <a:close/>
    </a:path>
  </a:pathLst>
</a:custGeom>
```

If however we change the guide to half that, namely 1/3. Then we see the entire bottom edge of the triangle move to now only occupy 1/3 of the toal space within the shape bounding box. This is because both of the bottom points in this triangle depend on this guide for their coordinate positions. The triangle and corresponding DrawingML shown below illustrate this point.



```
<a:gdLst>
  <a:gd name="myGuide" fmla="*/ h 1 3"/>
</a:gdLst>
```

end example]

Parent Elements
avLst (§5.1.11.5); gdLst (§5.1.11.12)

Attributes	Description
fmla (Shape Guide Formula)	Specifies the formula that will be used to calculate the value for a guide. Each formula has a certain number of arguments and a specific set of operations to perform on these arguments in order to generate a value for a guide. There are a total of 17 different formulas available. These are shown below with the usage for each defined. **('*/') - Multiply Divide Formula** Arguments: 3 (fmla="*/ x y z") Usage: "*/ x y z" = ((x * y) / z) = value of this guide **('+-') - Add Subtract Formula** Arguments: 3 (fmla="+- x y z") Usage: "+- x y z" = ((x + y) - z) = value of this guide **('+/') - Add Divide Formula** Arguments: 3 (fmla="+/ x y z") Usage: "+/ x y z" = ((x + y) / z) = value of this guide **('?:') - If Else Formula** Arguments: 3 (fmla="?: x y z") Usage: "?: x y z" = if (x > 0), then y = value of this guide, else z = value of this guide **('abs') - Absolute Value Formula** Arguments: 1 (fmla="abs x") Usage: "abs x" = if (x < 0), then (-1) * x = value of this guide else x = value of this guide

Attributes	Description
	('at2') - ArcTan Formula Arguments: 2 (fmla="at2 x y") Usage: "at2 x y" = arctan(y / x) = value of this guide **('cat2') - Cosine ArcTan Formula** Arguments: 3 (fmla="cat2 x y z") Usage: "cat2 x y z" = (x*(cos(arctan(z / y))) = value of this guide **('cos') - Cosine Formula** Arguments: 2 (fmla="cos x y") Usage: "cos x y" = (x * cos(y)) = value of this guide **('max') - Maximum Value Formula** Arguments: 2 (fmla="max x y") Usage: "max x y" = if (x > y), then x = value of this guide else y = value of this guide **('min') - Minimum Value Formula** Arguments: 2 (fmla="min x y") Usage: "min x y" = if (x < y), then x = value of this guide else y = value of this guide **('mod') - Modulo Formula** Arguments: 3 (fmla="mod x y z") Usage: "mod x y z" = sqrt(x^2 + b^2 + c^2) = value of this guide **('pin') - Pin To Formula** Arguments: 3 (fmla="pin x y z") Usage: "pin x y z" = if (y < x), then x = value of this guide else if (y > z), then z = value of this guide else y = value of this guide **('sat2') - Sine ArcTan Formula** Arguments: 3 (fmla="sat2 x y z") Usage: "sat2 x y z" = (x*sin(arctan(z / y))) = value of this guide **('sin') - Sine Formula** Arguments: 2 (fmla="sin x y") Usage: "sin x y" = (x * sin(y)) = value of this guide **('sqrt') - Square Root Formula** Arguments: 1 (fmla="sqrt x") Usage: "sqrt x" = sqrt(x) = value of this guide **('tan') - Tangent Formula** Arguments: 2 (fmla="tan x y") Usage: "tan x y" = (x * tan(y)) = value of this guide **('val') - Literal Value Formula** Arguments: 1 (fmla="val x") Usage: "val x" = x = value of this guide

Attributes	Description
	[*Note*: Guides that have a literal value formula specified via fmla="val x" above should only be used within the avLst as an adjust value for the shape. This however is not strictly enforced. *end note*] The possible values for this attribute are defined by the ST_GeomGuideFormula simple type (§5.1.12.25).
name (Shape Guide Name)	Specifies the name that will be used to reference to this guide. This name may be used just as a variable would within an equation. That is this name may be substituted for literal values within other guides or the specification of the shape path. The possible values for this attribute are defined by the ST_GeomGuideName simple type (§5.1.12.26).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GeomGuide">
   <attribute name="name" type="ST_GeomGuideName" use="required"/>
   <attribute name="fmla" type="ST_GeomGuideFormula" use="required"/>
</complexType>
```

5.1.11.12 gdLst (List of Shape Guides)

This element specifies all the guides that will be used for this shape. A guide is specified by the gd element and defines a calculated value that may be used for the construction of the corresponding shape.

[*Note*: Guides that have a literal value formula specified via fmla="val x" above should only be used within the avLst as an adjust value for the shape. This however is not strictly enforced. *end note*]

Parent Elements
custGeom (§5.1.11.8)

Child Elements	Subclause
gd (Shape Guide)	§5.1.11.11

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GeomGuideList">
   <sequence>
      <element name="gd" type="CT_GeomGuide" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.1.11.13 lnTo (Draw Line To)

This element specifies the drawing of a straight line from the current pen position to the new point specified. This line becomes part of the shape geometry, representing a side of the shape. The coordinate system used when specifying this line is the path coordinate system.

Parent Elements
path (§5.1.11.15)

Child Elements	Subclause
pt (Shape Path Point)	§5.1.11.20

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Path2DLineTo">
   <sequence>
      <element name="pt" type="CT_AdjPoint2D" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.11.14 moveTo (Move Path To)

This element specifies a set of new coordinates to move the shape cursor to. This element is only used for drawing a custom geometry. When this element is utilized the pt element is used to specify a new set of shape coordinates that the shape cursor should be moved to. This will not draw a line or curve to this new position from the old position but simply move the cursor to a new starting position. It is only when a path drawing element such as lnTo is used that a portion of the path will be drawn.

[*Example*: Consider the case where a user wishes to begin drawing a custom geometry not at the default starting coordinates of x=0 , y=0 but at coordinates further inset into the shape coordinate space. The following DrawingML would specify such a case.

```
<a:custGeom>
  <a:pathLst>
    <a:path w="2824222" h="590309">
      <a:moveTo>
        <a:pt x="0" y="428263"/>
      </a:moveTo>
      <a:lnTo>
        <a:pt x="1620455" y="590309"/>
      </a:lnTo>
      <a:lnTo>
        <a:pt x="2824222" y="173620"/>
      </a:lnTo>
      <a:lnTo>
        <a:pt x="1562582" y="0"/>
      </a:lnTo>
      <a:close/>
    </a:path>
  </a:pathLst>
</a:custGeom>
```

Notice the moveTo element advances the y coordinates before any actual lines are drawn. *end example*]

Parent Elements
path (§5.1.11.15)

Child Elements	Subclause
pt (Shape Path Point)	§5.1.11.20

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Path2DMoveTo">
   <sequence>
      <element name="pt" type="CT_AdjPoint2D" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.1.11.15 path (Shape Path)

This element specifies a creation path consisting of a series of moves, lines and curves that when combined will form a geometric shape. This element will only be utilized if a custom geometry is specified.

[*Note*: Since multiple paths are allowed the rules for drawing are that the path specified later in the pathLst will be drawn on top of all previous paths. *end note*]

[*Example*: Consider the following DrawingML.

```
<a:custGeom>
  <a:pathLst>
    <a:path w="2824222" h="590309">
      <a:moveTo>
        <a:pt x="0" y="428263"/>
      </a:moveTo>
      <a:lnTo>
        <a:pt x="1620455" y="590309"/>
      </a:lnTo>
      <a:lnTo>
        <a:pt x="2824222" y="173620"/>
      </a:lnTo>
      <a:lnTo>
        <a:pt x="1562582" y="0"/>
      </a:lnTo>
      <a:close/>
    </a:path>
  </a:pathLst>
</a:custGeom>
```

In the above example there is specified a four sided geometric shape that has all straight sides. While we only see three lines being drawn via the lnTo element there are actually four sides because the last point of (x=1562585, y=0) is connected to the first point in the creation path via a lnTo element. *end example*]

Parent Elements
pathLst (§5.1.11.16)

Child Elements	Subclause
arcTo (Draw Arc To)	§5.1.11.4
close (Close Shape Path)	§5.1.11.6
cubicBezTo (Draw Cubic Bezier Curve To)	§5.1.11.7
lnTo (Draw Line To)	§5.1.11.13
moveTo (Move Path To)	§5.1.11.14
quadBezTo (Draw Quadratic Bezier Curve To)	§5.1.11.21

Attributes	Description
extrusionOk (3D Extrusion Allowed)	Specifies that the use of 3D extrusions are possible on this path. This allows the generating application to know whether 3D extrusion can be applied in any form. If this attribute is omitted then a value of 0, or false is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
fill (Path Fill)	Specifies how the corresponding path should be filled. If this attribute is omitted, a value of "norm" is assumed. The possible values for this attribute are defined by the ST_PathFillMode simple type (§5.1.12.38).
h (Path Height)	Specifies the height, or maximum y coordinate that should be used for within the path coordinate system. This value determines the vertical placement of all points within the corresponding path as they will all be calculated using this height attribute as the max y coordinate. The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).
stroke (Path Stroke)	Specifies if the corresponding path should have a path stroke shown. This is a boolean value that will effect on the outline of the path. If this attribute is omitted, a value of true is assumed. The possible values for this attribute are defined by the XML Schema boolean datatype.
w (Path Width)	Specifies the width, or maximum x coordinate that should be used for within the path coordinate system. This value determines the horizontal placement of all points within

Attributes	Description
	the corresponding path as they will all be calculated using this width attribute as the max x coordinate. The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Path2D">
   <choice minOccurs="0" maxOccurs="unbounded">
      <element name="close" type="CT_Path2DClose" minOccurs="1" maxOccurs="1"/>
      <element name="moveTo" type="CT_Path2DMoveTo" minOccurs="1" maxOccurs="1"/>
      <element name="lnTo" type="CT_Path2DLineTo" minOccurs="1" maxOccurs="1"/>
      <element name="arcTo" type="CT_Path2DArcTo" minOccurs="1" maxOccurs="1"/>
      <element name="quadBezTo" type="CT_Path2DQuadBezierTo" minOccurs="1" maxOccurs="1"/>
      <element name="cubicBezTo" type="CT_Path2DCubicBezierTo" minOccurs="1" maxOccurs="1"/>
   </choice>
   <attribute name="w" type="ST_PositiveCoordinate" use="optional" default="0"/>
   <attribute name="h" type="ST_PositiveCoordinate" use="optional" default="0"/>
   <attribute name="fill" type="ST_PathFillMode" use="optional" default="norm"/>
   <attribute name="stroke" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="extrusionOk" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.1.11.16 pathLst (List of Shape Paths)

This element specifies the entire path that is to make up a single geometric shape. The pathLst can consist of many individual paths within it.

[*Example*: Consider the following DrawingML.

```
<a:custGeom>
  <a:pathLst>
    <a:path w="2824222" h="590309">
      <a:moveTo>
        <a:pt x="0" y="428263"/>
      </a:moveTo>
      <a:lnTo>
        <a:pt x="1620455" y="590309"/>
      </a:lnTo>
      <a:lnTo>
        <a:pt x="2824222" y="173620"/>
      </a:lnTo>
      <a:lnTo>
        <a:pt x="1562582" y="0"/>
      </a:lnTo>
```

```
        <a:close/>
      </a:path>
    </a:pathLst>
  </a:custGeom>
```

In the above example there is specified a four sided geometric shape that has all straight sides. While we only see three lines being drawn via the lnTo element there are actually four sides because the last point of (x=1562585, y=0) is connected to the first point in the creation path via a lnTo element. *end example*]

[*Note*: A geometry with multiple paths within it should be treated visually as if each path were a distinct shape. That is each creation path will have its first point and last point joined to form a closed shape. However, the generating application should then connect the last point to the first point of the new shape. If a close element is encountered at the end of the previous creation path then this joining line should not be rendered by the generating application. The rendering should resume with the first line or curve on the new creation path. *end note*]

Parent Elements
custGeom (§5.1.11.8)

Child Elements	Subclause
path (Shape Path)	§5.1.11.15

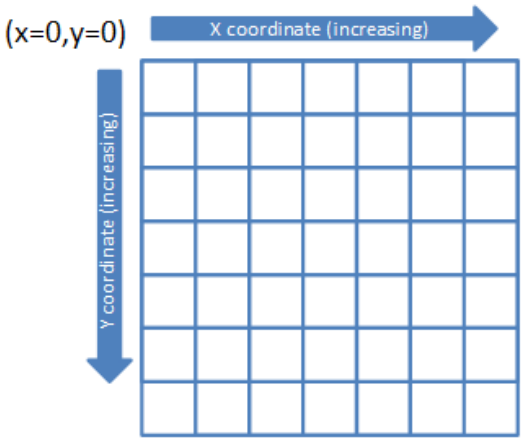
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Path2DList">
   <sequence>
      <element name="path" type="CT_Path2D" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.1.11.17 pos (Shape Position Coordinate)

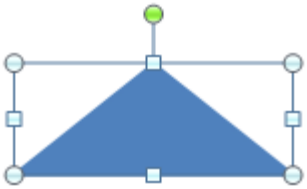
Specifies a position coordinate within the shape bounding box. It should be noted that this coordinate is placed within the shape bounding box using the transform coordinate system which is also called the shape coordinate system, as it encompasses the entire shape. The width and height for this coordinate system are specified within the ext transform element.

[*Note*: When specifying a point coordinate in path coordinate space it should be noted that the top left of the coordinate space is x=0, y=0 and the coordinate points for x grow to the right and for y grow down. This is illustrated in the diagram below.



end note]

[*Example*: To highlight the differences in the coordinate systems consider the drawing of the following triangle. Notice that the dimensions of the triangle are specified using the shape coordinate system with EMUs as the units via the ext transform element. Thus we see this shape is 1705233 EMUs wide by 679622 EMUs tall. However when looking at how the path for this shape is drawn we see that the x and y values fall between 0 and 2. This is because the path coordinate system has the arbitrary dimensions of 2 for the width and 2 for the height. Thus we see that a y coordinate of 2 within the path coordinate system will specify a y coordinate of 679622 within the shape coordinate system for this particular case.



```
<a:xfrm>
  <a:off x="3200400" y="1600200"/>
  <a:ext cx="1705233" cy="679622"/>
</a:xfrm>
<a:custGeom>
  <a:avLst/>
    <a:gdLst/>
  <a:ahLst/>
  <a:cxnLst/>
  <a:rect l="0" t="0" r="0" b="0"/>
```

```
    <a:pathLst>
      <a:path w="2" h="2">
        <a:moveTo>
          <a:pt x="0" y="2"/>
        </a:moveTo>
        <a:lnTo>
          <a:pt x="2" y="2"/>
        </a:lnTo>
        <a:lnTo>
          <a:pt x="1" y="0"/>
        </a:lnTo>
        <a:close/>
      </a:path>
    </a:pathLst>
  </a:custGeom>
```

end example]

Parent Elements
ahPolar (§5.1.11.2); ahXY (§5.1.11.3); cxn (§5.1.11.9)

Attributes	Description
x (X-Coordinate)	Specifies the x coordinate for this position coordinate. The units for this coordinate space are defined by the width of the path coordinate system. This coordinate system is overlayed on top of the shape coordinate system thus occupying the entire shape bounding box. Because the units for within this coordinate space are determined by the path width and height an exact measurement unit cannot be specified here. The possible values for this attribute are defined by the ST_AdjCoordinate simple type (§5.1.12.2).
y (Y-Coordinate)	Specifies the y coordinate for this position coordinate. The units for this coordinate space are defined by the height of the path coordinate system. This coordinate system is overlayed on top of the shape coordinate system thus occupying the entire shape bounding box. Because the units for within this coordinate space are determined by the path width and height an exact measurement unit cannot be specified here. The possible values for this attribute are defined by the ST_AdjCoordinate simple type (§5.1.12.2).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AdjPoint2D">
   <attribute name="x" type="ST_AdjCoordinate" use="required"/>
   <attribute name="y" type="ST_AdjCoordinate" use="required"/>
</complexType>
```

5.1.11.18 prstGeom (Preset geometry)

This element specifies when a preset geometric shape should be used instead of a custom geometric shape. The generating application should be able to render all preset geometries enumerated in the ST_ShapeType list.

[*Example*: Consider the scenario when a user does not wish to specify all the lines and curves that make up the desired shape but instead chooses to use a preset geometry. The following DrawingML would specify such a case.

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="4" name="My Preset Shape"/>
    <p:cNvSpPr/>
    <p:nvPr/>
  </p:nvSpPr>
  <p:spPr>
    <a:xfrm>
      <a:off x="1981200" y="533400"/>
      <a:ext cx="1143000" cy="1066800"/>
    </a:xfrm>
    <a:prstGeom prst="heart">
    </a:prstGeom>
  </p:spPr>
</p:sp>
```

The output shape rendered by this DrawingML is shown above. *end example*]

Parent Elements
spPr (§5.6.2.29); spPr (§5.7.2.198); spPr (§5.8.2.23); spPr (§5.1.2.1.35); spPr (§4.4.1.41); spPr (§5.9.3.7); spPr (§5.2.2.6)

Child Elements	Subclause
avLst (List of Shape Adjust Values)	§5.1.11.5

Attributes	Description
prst (Preset Shape)	Specifies the preset geometry that will be used for this shape. This preset can have any of

Attributes	Description

	the values in the enumerated list for ST_ShapeType. This attribute is required in order for a preset geometry to be rendered.
	
	[*Example*: Consider the sample DrawingML below.
	
	`<p:sp>`
	`  <p:nvSpPr>`
	`  <p:cNvPr id="4" name="Sun 3"/>`
	`  <p:cNvSpPr/>`
	`  <p:nvPr/>`
	`  </p:nvSpPr>`
	`  <p:spPr>`
	`  <a:xfrm>`
	`  <a:off x="1981200" y="533400"/>`
	`  <a:ext cx="1143000" cy="1066800"/>`
	`  </a:xfrm>`
	`  <a:prstGeom prst="sun">`
	`  </a:prstGeom>`
	`  </p:spPr>`
	`</p:sp>`
	
	In the above example a preset geometry has been used to define a shape. The shape utilized here is the sun shape. *end example*]
	
	The possible values for this attribute are defined by the ST_ShapeType simple type (§5.1.12.56).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PresetGeometry2D">
   <sequence>
      <element name="avLst" type="CT_GeomGuideList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="prst" type="ST_ShapeType" use="required"/>
</complexType>
```

5.1.11.19 prstTxWarp (Preset Text Warp)

This element specifies when a preset geometric shape should be used to transform a piece of text. This operation is known formally as a text warp. The generating application should be able to render all preset geometries enumerated in the ST_TextShapeType list.

[*Example*: Consider the case where the user wishes to accent a piece of text by warping it's shape. For this to occur a preset shape will be chosen from the ST_TextShapeType list and applied to the entire body of text.

```
<p:sp>
  <p:txBody>
    <a:bodyPr wrap="none" rtlCol="0">
      <a:prstTxWarp prst="textInflate">
      </a:prstTxWarp>
      <a:spAutoFit/>
    </a:bodyPr>
    <a:lstStyle/>
    <a:p>
…
      <a:t>Sample Text</a:t>
…
    </a:p>
  </p:txBody>
</p:sp>
```

The resulting text that has now had the Inflate text warp applied to it is shown above. *end example*]

Using any of the presets listed under the ST_TextShapeType list below it is possible to apply a text warp to a run of DrawingML text via the following steps.

If you look at any of the text warps in the file format you will notice that each consists of two paths. This corresponds to a top path (first one specified) and a bottom path (second one specified). Now the top path and the bottom path represent the top line and base line that the text needs to be warped to. How this is done is in the following way.

22. Compute the rectangle that the unwarped text resides in. (tightest possible rectangle around text, no white space except for “space characters”)
23. Take each of the quadratic and cubic Bezier curves that are used to calculate the original character and change their end points and control points by the following method…
24. Move a vertical line horizontally along the original text rectangle and find the horizontal percentage that a given end point or control point lives at. (.5 for the middle for instance)
25. Now do the same thing for this point vertically. Find the vertical percentage that this point lives at with the top and bottom of this text rectangle being the respective top and bottom bounds. (0.0 and 1.0 respectively)
26. Now that we have the percentages for a given point in a Bezier equation we can map that to the new point in the warped text environment.

27. Going back to the top and bottom paths specified in the file format we can take these and flatten them out to a straight arc (top and bottom might be different lengths)
28. After they are straight we can measure them both horizontally to find the same percentage point that we found within the original text rectangle. (0.5 let's say)
29. So then we measure 50% along the top path and 50% along the bottom path, putting the paths back to their original curvy shapes.
30. Once we have these two points we can draw a line between them that will serve as our vertical line in the original text rectangle (note: this might not be truly vertical as 50% on the top does not always line up with 50% on the bottom)
31. Taking this new line we then follow it from top to bottom the vertical percentage amount that we got from step 4.
32. This is then the new point that should be used in place of the old point in the original text rectangle.
33. We will then continue doing these same steps for each of the end points and control points within the body of text. (is applied to a whole body of text only)

[*Note*: Horizxontal percentages begin at 0.0 and continue to 1.0, left to right. Vertical percentages begin at 0.0 and continue to 1.0, top to bottom. *end note*]

[*Note*: Since this is a shape it does have both a shape coordinate system and a path coordinate system. *end note*]

Parent Elements
bodyPr (§5.1.5.1.1)

Child Elements	Subclause
avLst (List of Shape Adjust Values)	§5.1.11.5

Attributes	Description

prst (Preset Warp Shape)	Specifies the preset geometry that will be used for a shape warp on a piece of text. This preset can have any of the values in the enumerated list for ST_TextShapeType. This attribute is required in order for a text warp to be rendered.
	
	[*Example*: Consider the sample DrawingML below.
	
	`<p:sp>`
	`  <p:txBody>`
	`  <a:bodyPr wrap="none" rtlCol="0">`
	`  <a:prstTxWarp prst="textInflate">`
	`  </a:prstTxWarp>`
	`  <a:spAutoFit/>`
	`  </a:bodyPr>`
	`  <a:lstStyle/>`
	`  <a:p>`

Attributes	Description
	… <a:t>Sample Text</a:t> … </a:p> </p:txBody> </p:sp> In the above example a preset text shape geometry has been used to define the warping shape. The shape utilized here is the sun shape. *end example*] The possible values for this attribute are defined by the ST_TextShapeType simple type (§5.1.12.76).

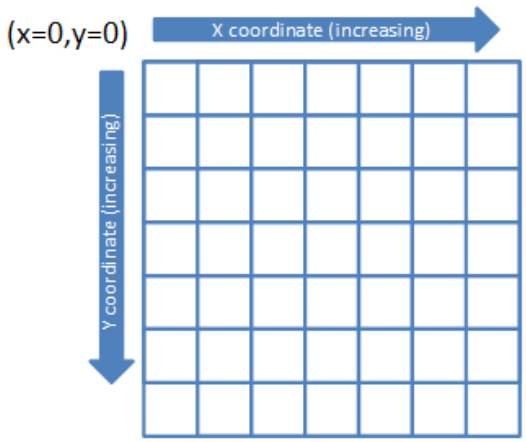
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PresetTextShape">
   <sequence>
      <element name="avLst" type="CT_GeomGuideList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="prst" type="ST_TextShapeType" use="required"/>
</complexType>
```

5.1.11.20 pt (Shape Path Point)

This element specifies an x-y coordinate within the path coordinate space. This coordinate space is determined by the width and height attributes defined within the path element. A point is utilized by one of it's parent elements to specify the next point of interest in custom geometry shape. Depending on the parent element used the point may either have a line drawn to it or the cursor may simply be moved to this new location.

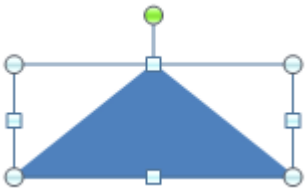
[*Note*: When specifying a point coordinate in path coordinate space it should be noted that the top left of the coordinate space is x=0, y=0 and the coordinate points for x grow to the right and for y grow down. This is illustrated in the diagram below.



end note]

Specifies a position coordinate within the shape bounding box. It should be noted that this coordinate is placed within the shape bounding box using the transform coordinate system which is also called the shape coordinate system, as it encompasses the entire shape. The width and height for this coordinate system are specified within the ext transform element.

[*Example*: To highlight the differences in the coordinate systems consider the drawing of the following triangle. Notice that the dimensions of the triangle are specified using the shape coordinate system with EMUs as the units via the ext transform element. Thus we see this shape is 1705233 EMUs wide by 679622 EMUs tall. However when looking at how the path for this shape is drawn we see that the x and y values fall between 0 and 2. This is because the path coordinate system has the arbitrary dimensions of 2 for the width and 2 for the height. Thus we see that a y coordinate of 2 within the path coordinate system will specify a y coordinate of 679622 within the shape coordinate system for this particular case.



```
<a:xfrm>
  <a:off x="3200400" y="1600200"/>
  <a:ext cx="1705233" cy="679622"/>
</a:xfrm>
<a:custGeom>
  <a:avLst/>
    <a:gdLst/>
  <a:ahLst/>
  <a:cxnLst/>
  <a:rect l="0" t="0" r="0" b="0"/>
  <a:pathLst>
    <a:path w="2" h="2">
      <a:moveTo>
        <a:pt x="0" y="2"/>
      </a:moveTo>
      <a:lnTo>
        <a:pt x="2" y="2"/>
      </a:lnTo>
      <a:lnTo>
        <a:pt x="1" y="0"/>
      </a:lnTo>
```

```
        <a:close/>
      </a:path>
    </a:pathLst>
  </a:custGeom>
```

end example]

Parent Elements
cubicBezTo (§5.1.11.7); lnTo (§5.1.11.13); moveTo (§5.1.11.14); quadBezTo (§5.1.11.21)

Attributes	Description
x (X-Coordinate)	Specifies the x coordinate for this position coordinate. The units for this coordinate space are defined by the width of the path coordinate system. This coordinate system is overlayed on top of the shape coordinate system thus occupying the entire shape bounding box. Because the units for within this coordinate space are determined by the path width and height an exact measurement unit cannot be specified here. The possible values for this attribute are defined by the ST_AdjCoordinate simple type (§5.1.12.2).
y (Y-Coordinate)	Specifies the y coordinate for this position coordinate. The units for this coordinate space are defined by the height of the path coordinate system. This coordinate system is overlayed on top of the shape coordinate system thus occupying the entire shape bounding box. Because the units for within this coordinate space are determined by the path width and height an exact measurement unit cannot be specified here. The possible values for this attribute are defined by the ST_AdjCoordinate simple type (§5.1.12.2).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AdjPoint2D">
   <attribute name="x" type="ST_AdjCoordinate" use="required"/>
   <attribute name="y" type="ST_AdjCoordinate" use="required"/>
</complexType>
```

5.1.11.21 quadBezTo (Draw Quadratic Bezier Curve To)

This element specifies to draw a quadratic bezier curve along the specified points. To specify a quadratic bezier curve there needs to be 2 points specified. The first is a control point used in the quadratic bezier calculation and the last is the ending point for the curve. The coordinate system used for this type of curve is the path coordinate system as this element is path specific.

Parent Elements
path (§5.1.11.15)

Child Elements	Subclause
pt (Shape Path Point)	§5.1.11.20

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Path2DQuadBezierTo">
   <sequence>
      <element name="pt" type="CT_AdjPoint2D" minOccurs="2" maxOccurs="2"/>
   </sequence>
</complexType>
```

5.1.11.22 rect (Shape Text Rectangle)

This element specifies the rectangular bounding box for text within a custGeom shape. The default for this rectangle is the bounding box for the shape. This can be modified using this elements four attributes to inset or extend the text bounding box.

[*Note*: Text specified to reside within this shape text rectangle may flow outside this bounding box. Depending on the autofit options within the txBody element the text may or may not entirely reside within this shape text rectangle. *end note*]

Parent Elements
custGeom (§5.1.11.8)

Attributes	Description
b (Bottom Position)	Specifies the y coordinate of the bottom edge for a shape text rectangle. The units for this edge is specified in EMUs as the positioning here is based on the shape coordinate system. The width and height for this coordinate system are specified within the ext transform element. The possible values for this attribute are defined by the ST_AdjCoordinate simple type (§5.1.12.2).
l (Left)	Specifies the x coordinate of the left edge for a shape text rectangle. The units for this edge is specified in EMUs as the positioning here is based on the shape coordinate system. The width and height for this coordinate system are specified within the ext transform element. The possible values for this attribute are defined by the ST_AdjCoordinate simple type (§5.1.12.2).
r (Right)	Specifies the x coordinate of the right edge for a shape text rectangle. The units for this edge is specified in EMUs as the positioning here is based on the shape coordinate system. The width and height for this coordinate system are specified within the ext transform element. The possible values for this attribute are defined by the ST_AdjCoordinate simple type

Attributes	Description
	(§5.1.12.2).
t (Top)	Specifies the y coordinate of the top edge for a shape text rectangle. The units for this edge is specified in EMUs as the positioning here is based on the shape coordinate system. The width and height for this coordinate system are specified within the ext transform element. The possible values for this attribute are defined by the ST_AdjCoordinate simple type (§5.1.12.2).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GeomRect">
   <attribute name="l" type="ST_AdjCoordinate" use="required"/>
   <attribute name="t" type="ST_AdjCoordinate" use="required"/>
   <attribute name="r" type="ST_AdjCoordinate" use="required"/>
   <attribute name="b" type="ST_AdjCoordinate" use="required"/>
</complexType>
```

5.1.12 Simple Types

This is the complete list of simple types in the http://schemas.openxmlformats.org/drawingml/2006/main namespace.

5.1.12.1 ST_AdjAngle (Adjustable Angle Methods)

This simple type is an adjustable angle, either an absolute angle or a reference to a geometry guide. The units for an adjustable angle are 60,000ths of a degree.

This simple type is defined as a union of the following types:

- TheST_Angle simple type (§5.1.12.3).
- TheST_GeomGuideName simple type (§5.1.12.26).

Referenced By
ahPolar@maxAng (§5.1.11.2); ahPolar@minAng (§5.1.11.2); arcTo@stAng (§5.1.11.4); arcTo@swAng (§5.1.11.4); cxn@ang (§5.1.11.9)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_AdjAngle">
   <union memberTypes="ST_Angle ST_GeomGuideName"/>
</simpleType>
```

5.1.12.2 ST_AdjCoordinate (Adjustable Coordinate Methods)

This simple type is an adjustable coordinate is either an absolute coordinate position or a reference to a geometry guide.

This simple type is defined as a union of the following types:

- TheST_Coordinate simple type (§5.1.12.16).
- TheST_GeomGuideName simple type (§5.1.12.26).

Referenced By
ahPolar@maxR (§5.1.11.2); ahPolar@minR (§5.1.11.2); ahXY@maxX (§5.1.11.3); ahXY@maxY (§5.1.11.3); ahXY@minX (§5.1.11.3); ahXY@minY (§5.1.11.3); arcTo@hR (§5.1.11.4); arcTo@wR (§5.1.11.4); pos@x (§5.1.11.17); pos@y (§5.1.11.17); pt@x (§5.1.11.20); pt@y (§5.1.11.20); rect@b (§5.1.11.22); rect@l (§5.1.11.22); rect@r (§5.1.11.22); rect@t (§5.1.11.22)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_AdjCoordinate">
   <union memberTypes="ST_Coordinate ST_GeomGuideName"/>
</simpleType>
```

5.1.12.3 ST_Angle (Angle)

This simple type represents an angle in 60,000ths of a degree. Positive angles are clockwise (i.e., towards the positive y axis); negative angles are counter-clockwise (i.e., towards the negative y axis).

This simple type's contents are a restriction of the XML Schema int datatype.

Referenced By
animMotion@rAng (§4.6.4); animRot@by (§4.6.5); animRot@from (§4.6.5); animRot@to (§4.6.5); bodyPr@rot (§5.1.5.1.1); hsl@h (§4.6.46); hueOff@val (§5.1.2.2.16); ST_AdjAngle (§5.1.12.1); ST_FixedAngle (§5.1.12.21); ST_FOVAngle (§5.1.12.24); ST_PositiveFixedAngle (§5.1.12.44); xfrm@rot (§4.4.1.49); xfrm@rot (§5.1.9.6); xfrm@rot (§5.6.2.35); xfrm@rot (§5.8.2.28); xfrm@rot (§5.1.9.5)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Angle">
   <restriction base="xsd:int"/>
</simpleType>
```

5.1.12.4 ST_AnimationBuildType (Animation Build Type)

This simple type specifies the ways that an animation can be built, or animated.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
allAtOnce (Animate At Once)	Animate all objects as one.

Referenced By
ST_AnimationChartBuildType (§5.1.12.5); ST_AnimationDgmBuildType (§5.1.12.7)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_AnimationBuildType">
   <restriction base="xsd:token">
      <enumeration value="allAtOnce"/>
   </restriction>
</simpleType>
```

5.1.12.5 ST_AnimationChartBuildType (Chart Animation Build Type)

This simple type specifies the ways that a chart animation can be built. That is, it specifies the way in which the objects within the chart should be animated.

This simple type is defined as a union of the following types:

- TheST_AnimationBuildType simple type (§5.1.12.4).
- TheST_AnimationChartOnlyBuildType simple type (§5.1.12.6).

Referenced By
bldChart@bld (§5.1.2.1.1)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_AnimationChartBuildType">
   <union memberTypes="ST_AnimationBuildType ST_AnimationChartOnlyBuildType"/>
</simpleType>
```

5.1.12.6 ST_AnimationChartOnlyBuildType (Chart only Animation Types)

This simple type specifies the build options available only for animating a chart. These options specify the manner in which the objects within the chart should be grouped and animated.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
category (Catefory)	Animate by each category
categoryEl (Category Element)	Animate by each element within the category
series (Series)	Animate by each series.
seriesEl (Series Element)	Animate by each element within the series

Referenced By
ST_AnimationChartBuildType (§5.1.12.5)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_AnimationChartOnlyBuildType">
   <restriction base="xsd:token">
      <enumeration value="series"/>
      <enumeration value="category"/>
      <enumeration value="seriesEl"/>
      <enumeration value="categoryEl"/>
   </restriction>
</simpleType>
```

5.1.12.7 ST_AnimationDgmBuildType (Diagram Animation Build Type)

This simple type specifies the ways that a diagram animation can be built. That is, it specifies the way in which the objects within the diagram graphical object should be animated.

This simple type is defined as a union of the following types:

- TheST_AnimationBuildType simple type (§5.1.12.4).
- TheST_AnimationDgmOnlyBuildType simple type (§5.1.12.8).

Referenced By
bldDgm@bld (§5.1.2.1.2)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_AnimationDgmBuildType">
   <union memberTypes="ST_AnimationBuildType ST_AnimationDgmOnlyBuildType"/>
</simpleType>
```

5.1.12.8 ST_AnimationDgmOnlyBuildType (Diagram only Animation Types)

This simple type specifies the build options available only for animating a diagram. These options specify the manner in which the objects within the chart should be grouped and animated.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
lvlAtOnce (Each Level at Once)	Animate the diagram one level at a time, animating the whole level as one object
lvlOne (Level One-by-One)	Animate the diagram by the elements within a level, animating them one level element at a time.
one (Elements One-by-One)	Animate the diagram by elements. For a tree diagram the animation will occur by branch within the diagram tree.

Referenced By
ST_AnimationDgmBuildType (§5.1.12.7)

The following XML Schema fragment defines the contents of this simple type:

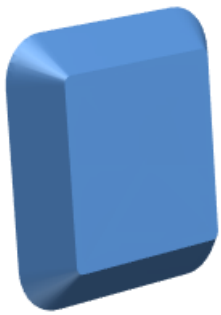
```
<simpleType name="ST_AnimationDgmOnlyBuildType">
   <restriction base="xsd:token">
      <enumeration value="one"/>
      <enumeration value="lvlOne"/>
      <enumeration value="lvlAtOnce"/>
   </restriction>
</simpleType>
```

5.1.12.9 ST_BevelPresetType (Bevel Presets)

Represents a preset for a type of bevel which can be applied to a shape in 3D. The bevel properties are applied differently depending on the type of bevel defined for a shape.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
angle (Angle)	[*Example*: Consider the following example of an `angle` bevel type applied to a shape:  *end example*]
artDeco (Art Deco)	[*Example*: Consider the following example of an `artDeco` bevel type applied to a shape:

Enumeration Value	Description
	 *end example*]
circle (Circle)	[*Example*: Consider the following example of an `circle` bevel type applied to a shape:  *end example*]
convex (Convex)	[*Example*: Consider the following example of an `convex` bevel type applied to a shape:  *end example*]
coolSlant (Cool Slant)	[*Example*: Consider the following example of an `coolSlant` bevel type applied to a shape:

Enumeration Value	Description
	 *end example*]
cross (Cross)	[*Example*: Consider the following example of an `cross` bevel type applied to a shape:  *end example*]
divot (Divot)	[*Example*: Consider the following example of an `divot` bevel type applied to a shape:  *end example*]
hardEdge (Hard Edge)	[*Example*: Consider the following example of an `hardEdge` bevel type applied to a shape:

Enumeration Value	Description
	 *end example*]
relaxedInset (Relaxed Inset)	[*Example*: Consider the following example of an `relaxedInset` bevel type applied to a shape:  *end example*]
riblet (Riblet)	[*Example*: Consider the following example of an `riblet` bevel type applied to a shape:  *end example*]
slope (Slope)	[*Example*: Consider the following example of an

Enumeration Value	Description
	`slope` bevel type applied to a shape:  *end example*]
softRound (Soft Round)	[*Example*: Consider the following example of an `softRound` bevel type applied to a shape:  *end example*]

Referenced By
bevel@prst (§5.1.4.2.5); bevelB@prst (§5.1.7.3); bevelT@prst (§5.1.7.4)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_BevelPresetType">
   <restriction base="xsd:token">
      <enumeration value="relaxedInset"/>
      <enumeration value="circle"/>
      <enumeration value="slope"/>
      <enumeration value="cross"/>
      <enumeration value="angle"/>
      <enumeration value="softRound"/>
      <enumeration value="convex"/>
      <enumeration value="coolSlant"/>
      <enumeration value="divot"/>
      <enumeration value="riblet"/>
      <enumeration value="hardEdge"/>
      <enumeration value="artDeco"/>
   </restriction>
</simpleType>
```

5.1.12.10 ST_BlackWhiteMode (Black and White Mode)

This simple type specifies how an object should be rendered when specified to be in black and white mode.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
auto (Automatic)	Object rendered with automatic coloring
black (Black)	Object rendered with black-only coloring
blackGray (Black and Gray)	Object rendered with black and gray coloring
blackWhite (Black and White)	Object rendered within black and white coloring
clr (Color)	Object rendered with normal coloring
gray (Gray)	Object rendered with gray coloring
grayWhite (Gray and White)	Object rendered within gray and white coloring
hidden (Hidden)	Object rendered with hidden coloring
invGray (Inverse Gray)	Object rendered with inverse gray coloring
ltGray (Light Gray)	Object rendered with light gray coloring
white (White)	Object rendered within white coloirng

Referenced By
bg@bwMode (§4.4.1.1); grpSpPr@bwMode (§5.8.2.14); grpSpPr@bwMode (§4.4.1.20); grpSpPr@bwMode (§5.1.2.1.22); grpSpPr@bwMode (§5.6.2.17); spPr@bwMode (§5.6.2.29); spPr@bwMode (§5.7.2.198); spPr@bwMode (§5.8.2.23); spPr@bwMode (§5.1.2.1.35); spPr@bwMode (§4.4.1.41); spPr@bwMode (§5.9.3.7); spPr@bwMode (§5.2.2.6)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_BlackWhiteMode">
   <restriction base="xsd:token">
      <enumeration value="clr"/>
      <enumeration value="auto"/>
      <enumeration value="gray"/>
      <enumeration value="ltGray"/>
      <enumeration value="invGray"/>
      <enumeration value="grayWhite"/>
      <enumeration value="blackGray"/>
      <enumeration value="blackWhite"/>
      <enumeration value="black"/>
      <enumeration value="white"/>
      <enumeration value="hidden"/>
   </restriction>
</simpleType>
```

5.1.12.11 ST_BlendMode (Blend Mode)

This simple type describes how to render effects one on top of another.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
darken (Darken)	Darken
lighten (Lighten)	Lighten
mult (Multiply)	Multiply
over (Overlay)	Overlay
screen (Screen)	Screen

Referenced By
blend@blend (§5.1.10.12); fillOverlay@blend (§5.1.10.29)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_BlendMode">
   <restriction base="xsd:token">
      <enumeration value="over"/>
      <enumeration value="mult"/>
      <enumeration value="screen"/>
      <enumeration value="darken"/>
      <enumeration value="lighten"/>
   </restriction>
</simpleType>
```

5.1.12.12 ST_BlipCompression (Blip Compression Type)

This type specifies the amount of compression that has been used for a particular binary large image or picture (blip).

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
email (Email Compression)	Compression size suitable for inclusion with email
hqprint (High Quality Printing Compression)	Compression size suitable for high quality printing
none (No Compression)	No compression was used
print (Printing Compression)	Compression size suitable for printing
screen (Screen Viewing Compression)	Compression size suitable for viewing on screen

Referenced By
blip@cstate (§5.1.10.13)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_BlipCompression">
   <restriction base="xsd:token">
      <enumeration value="email"/>
      <enumeration value="screen"/>
      <enumeration value="print"/>
      <enumeration value="hqprint"/>
      <enumeration value="none"/>
   </restriction>
</simpleType>
```

5.1.12.13 ST_ChartBuildStep (Chart Animation Build Step)

This simple type specifies an animation build step within a chart animation.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
allPts (All Points)	Animate all points within the chart for this animation build step
category (Category)	Animate a chart category for this animation build step
gridLegend (Grid and Legend)	Animate the chart grid and legend for this animation build step

Enumeration Value	Description
ptInCategory (Category Points)	Animate a point in a chart category for this animation build step
ptInSeries (Series Points)	Animate a point in a chart series for this animation build step
series (Series)	Animate a chart series for this animation build step

Referenced By
chart@bldStep (§5.1.2.1.3)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ChartBuildStep">
   <restriction base="xsd:token">
      <enumeration value="category"/>
      <enumeration value="ptInCategory"/>
      <enumeration value="series"/>
      <enumeration value="ptInSeries"/>
      <enumeration value="allPts"/>
      <enumeration value="gridLegend"/>
   </restriction>
</simpleType>
```

5.1.12.14 ST_ColorSchemeIndex (Theme Color Reference)

A reference to a color in the color scheme.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
accent1 (Accent 1)	Represents the accent 1 color.
accent2 (Accent 2)	Represents the accent 2 color.
accent3 (Accent 3)	Represents the accent 3 color.
accent4 (Accent 4)	Represents the accent 4 color.
accent5 (Accent 5)	Represents the accent 5 color.
accent6 (Accent 6)	Represents the accent 6 color.
dk1 (Dark 1)	Represents the first dark color.
dk2 (Dark 2)	Represents the second dark color.
folHlink (Followed Hyperlink)	Represents the followed hyperlink color.
hlink (Hyperlink)	Represents the hyperlink color.
lt1 (Light 1)	Represents the first light color.

Enumeration Value	Description
lt2 (Light 2)	Represents the second light color.

Referenced By
clrMap@accent1 (§4.4.1.6); clrMap@accent1 (§5.1.8.1); clrMap@accent2 (§4.4.1.6); clrMap@accent2 (§5.1.8.1); clrMap@accent3 (§4.4.1.6); clrMap@accent3 (§5.1.8.1); clrMap@accent4 (§4.4.1.6); clrMap@accent4 (§5.1.8.1); clrMap@accent5 (§4.4.1.6); clrMap@accent5 (§5.1.8.1); clrMap@accent6 (§4.4.1.6); clrMap@accent6 (§5.1.8.1); clrMap@bg1 (§4.4.1.6); clrMap@bg1 (§5.1.8.1); clrMap@bg2 (§4.4.1.6); clrMap@bg2 (§5.1.8.1); clrMap@folHlink (§4.4.1.6); clrMap@folHlink (§5.1.8.1); clrMap@hlink (§4.4.1.6); clrMap@hlink (§5.1.8.1); clrMap@tx1 (§4.4.1.6); clrMap@tx1 (§5.1.8.1); clrMap@tx2 (§4.4.1.6); clrMap@tx2 (§5.1.8.1); clrMapOvr@accent1 (§5.7.2.30); clrMapOvr@accent2 (§5.7.2.30); clrMapOvr@accent3 (§5.7.2.30); clrMapOvr@accent4 (§5.7.2.30); clrMapOvr@accent5 (§5.7.2.30); clrMapOvr@accent6 (§5.7.2.30); clrMapOvr@bg1 (§5.7.2.30); clrMapOvr@bg2 (§5.7.2.30); clrMapOvr@folHlink (§5.7.2.30); clrMapOvr@hlink (§5.7.2.30); clrMapOvr@tx1 (§5.7.2.30); clrMapOvr@tx2 (§5.7.2.30); overrideClrMapping@accent1 (§5.1.8.8); overrideClrMapping@accent2 (§5.1.8.8); overrideClrMapping@accent3 (§5.1.8.8); overrideClrMapping@accent4 (§5.1.8.8); overrideClrMapping@accent5 (§5.1.8.8); overrideClrMapping@accent6 (§5.1.8.8); overrideClrMapping@bg1 (§5.1.8.8); overrideClrMapping@bg2 (§5.1.8.8); overrideClrMapping@folHlink (§5.1.8.8); overrideClrMapping@hlink (§5.1.8.8); overrideClrMapping@tx1 (§5.1.8.8); overrideClrMapping@tx2 (§5.1.8.8)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ColorSchemeIndex">
   <restriction base="xsd:token">
      <enumeration value="dk1"/>
      <enumeration value="lt1"/>
      <enumeration value="dk2"/>
      <enumeration value="lt2"/>
      <enumeration value="accent1"/>
      <enumeration value="accent2"/>
      <enumeration value="accent3"/>
      <enumeration value="accent4"/>
      <enumeration value="accent5"/>
      <enumeration value="accent6"/>
      <enumeration value="hlink"/>
      <enumeration value="folHlink"/>
   </restriction>
</simpleType>
```

5.1.12.15 ST_CompoundLine (Compound Line Type)

This type specifies the compound line type that is to be used for lines with text such as underlines.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
dbl (Double Lines)	Double lines of equal width
sng (Single Line)	Single line: one normal width
thickThin (Thick Thin Double Lines)	Double lines: one thick, one thin
thinThick (Thin Thick Double Lines)	Double lines: one thin, one thick
tri (Thin Thick Thin Triple Lines)	Three lines: thin, thick, thin

Referenced By
ln@cmpd (§5.1.2.1.24); lnB@cmpd (§5.1.6.3); lnBlToTr@cmpd (§5.1.6.4); lnL@cmpd (§5.1.6.5); lnR@cmpd (§5.1.6.6); lnT@cmpd (§5.1.6.7); lnTlToBr@cmpd (§5.1.6.8); uLn@cmpd (§5.1.5.3.14)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_CompoundLine">
   <restriction base="xsd:token">
      <enumeration value="sng"/>
      <enumeration value="dbl"/>
      <enumeration value="thickThin"/>
      <enumeration value="thinThick"/>
      <enumeration value="tri"/>
   </restriction>
</simpleType>
```

5.1.12.16 ST_Coordinate (Coordinate)

This simple type represents a one dimensional position or length in EMUs. EMUs (English Metric Units) are a high precision coordinate space (914400 dpi / 360000 dpc).

This simple type's contents are a restriction of the XML Schema long datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to -27273042329600.
- This simple type has a maximum value of less than or equal to 27273042316900.

Referenced By
alphaOutset@rad (§5.1.10.7); anchor@x (§5.1.7.1); anchor@y (§5.1.7.1); anchor@z (§5.1.7.1); chOff@x (§5.1.9.2); chOff@y (§5.1.9.2); colOff (§5.6.2.11); effectExtent@b (§5.5.2.6); effectExtent@l (§5.5.2.6); effectExtent@r (§5.5.2.6); effectExtent@t (§5.5.2.6); flatTx@z (§5.1.7.8); gridCol@w (§5.1.6.2); lineTo@x (§5.5.2.9); lineTo@y (§5.5.2.9); norm@dx (§5.1.7.10); norm@dy (§5.1.7.10); norm@dz (§5.1.7.10); off@x (§5.1.9.4); off@y (§5.1.9.4); origin@x (§4.3.2.9); origin@y (§4.3.2.9); pos@x (§4.5.5); pos@x (§5.6.2.25); pos@y (§4.5.5); pos@y (§5.6.2.25); rowOff (§5.6.2.27); simplePos@x (§5.5.2.13); simplePos@y (§5.5.2.13); sp3d@z (§5.1.7.12); sp3d@z (§5.9.5.6); ST_AdjCoordinate (§5.1.12.2); start@x (§5.5.2.14); start@y (§5.5.2.14); tile@tx (§5.1.10.58); tile@ty (§5.1.10.58); tr@h (§5.1.6.16); up@dx (§5.1.7.13); up@dy (§5.1.7.13); up@dz (§5.1.7.13); xfrm@tx (§5.1.10.61); xfrm@ty (§5.1.10.61)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Coordinate">
   <restriction base="xsd:long">
      <minInclusive value="-27273042329600"/>
      <maxInclusive value="27273042316900"/>
   </restriction>
</simpleType>
```

5.1.12.17 ST_Coordinate32 (Coordinate Point)

This type specifies a coordinate within the document. This may be used for measurements or spacing with the maximum size requirement being a 32 bit integer.

NOTE: The units of measurement used here are EMUs (English Metric Units).

This simple type's contents are a restriction of the XML Schema int datatype.

Referenced By
bodyPr@bIns (§5.1.5.1.1); bodyPr@lIns (§5.1.5.1.1); bodyPr@rIns (§5.1.5.1.1); bodyPr@tIns (§5.1.5.1.1); defPPr@defTabSz (§5.1.5.2.2); guide@pos (§4.3.2.4); lvl1pPr@defTabSz (§5.1.5.4.13); lvl2pPr@defTabSz (§5.1.5.4.14); lvl3pPr@defTabSz (§5.1.5.4.15); lvl4pPr@defTabSz (§5.1.5.4.16); lvl5pPr@defTabSz (§5.1.5.4.17); lvl6pPr@defTabSz (§5.1.5.4.18); lvl7pPr@defTabSz (§5.1.5.4.19); lvl8pPr@defTabSz (§5.1.5.4.20); lvl9pPr@defTabSz (§5.1.5.4.21); pPr@defTabSz (§5.1.5.2.7); ST_LineWidth (§5.1.12.35); ST_PositiveCoordinate32 (§5.1.12.43); ST_TextIndent (§5.1.12.70); ST_TextMargin (§5.1.12.73); tab@pos (§5.1.5.2.12); tcPr@marB (§5.1.6.15); tcPr@marL (§5.1.6.15); tcPr@marR (§5.1.6.15); tcPr@marT (§5.1.6.15)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Coordinate32">
   <restriction base="xsd:int"/>
</simpleType>
```

5.1.12.18 ST_DgmBuildStep (Diagram Animation Build Steps)

This simple type specifies an animation build step within a diagram animation.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
bg (Background)	Animate the diagram background for this animation build step
sp (Shape)	Animate a diagram shape for this animation build step

Referenced By
dgm@bldStep (§5.1.2.1.12)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DgmBuildStep">
   <restriction base="xsd:token">
      <enumeration value="sp"/>
      <enumeration value="bg"/>
   </restriction>
</simpleType>
```

5.1.12.19 ST_DrawingElementId (Drawing Element ID)

This simple type specifies a unique integer identifier for each drawing element.

This simple type's contents are a restriction of the XML Schema unsignedInt datatype.

Referenced By
cNvPr@id (§5.2.2.3); cNvPr@id (§5.8.2.7); cNvPr@id (§4.4.1.12); cNvPr@id (§5.6.2.8); cNvPr@id (§5.1.2.1.8); docPr@id (§5.5.2.5); endCxn@id (§5.1.2.1.13); stCxn@id (§5.1.2.1.36)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DrawingElementId">
   <restriction base="xsd:unsignedInt"/>
</simpleType>
```

5.1.12.20 ST_EffectContainerType (Effect Container Type)

This simple type determines the relationship between effects in a container, either sibling or tree.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
sib (Sibling)	Each effect is separately applied to the parent object. [*Example*: If the parent element contains an outer shadow and a reflection, the resulting effect will be a shadow around the parent object and a reflection of the object. The reflection will not have a shadow. *end example*]
tree (Tree)	Each effect is applied to the result of the previous effect. [*Example*: If the parent element contains an outer shadow followed by a glow, the shadow will first be applied to the parent object. Then, the glow will be applied to the shadow (rather than the original object). The resulting effect would be a glowing shadow. *end example*]

Referenced By
cont@type (§5.1.10.20); effectDag@type (§5.1.10.25)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_EffectContainerType">
   <restriction base="xsd:token">
      <enumeration value="sib"/>
      <enumeration value="tree"/>
   </restriction>
</simpleType>
```

5.1.12.21 ST_FixedAngle (Fixed Angle)

This simple type represents a fixed range angle in 60000ths of a degree. Range from (-90, 90 degrees).

This simple type's contents are a restriction of the ST_Angle simple type (§5.1.12.3).

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than -5400000.
- This simple type has a maximum value of less than 5400000.

Referenced By
outerShdw@kx (§5.1.10.45); outerShdw@ky (§5.1.10.45); reflection@kx (§5.1.10.50); reflection@ky (§5.1.10.50); xfrm@kx (§5.1.10.61); xfrm@ky (§5.1.10.61)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FixedAngle">
   <restriction base="ST_Angle">
      <minExclusive value="-5400000"/>
      <maxExclusive value="5400000"/>
   </restriction>
</simpleType>
```

5.1.12.22 ST_FixedPercentage (Fixed Percentage)

This simple type represents a fixed percentage in 1000ths of a percent. Range from [-100%, 100%].

This simple type's contents are a restriction of the ST_Percentage simple type (§5.1.12.41).

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to -100000.
- This simple type has a maximum value of less than or equal to 100000.

Referenced By

Referenced By
alphaOff@val (§5.1.2.2.3); hsl@l (§4.6.46); hsl@lum (§5.1.10.39); hsl@s (§4.6.46); hsl@sat (§5.1.10.39); lum@bright (§5.1.10.42); lum@contrast (§5.1.10.42); rgb@b (§4.6.63); rgb@g (§4.6.63); rgb@r (§4.6.63); tint@amt (§5.1.10.60)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FixedPercentage">
   <restriction base="ST_Percentage">
      <minInclusive value="-100000"/>
      <maxInclusive value="100000"/>
   </restriction>
</simpleType>
```

5.1.12.23 ST_FontCollectionIndex (Font Collection Index)

This simple type represents one of the fonts associated with the style.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
major (Major Font)	The major font of the style's font scheme.
minor (Minor Font)	The minor font of the style's font scheme.
none (None)	No font reference.

Referenced By
fontRef@idx (§5.1.4.1.17)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FontCollectionIndex">
   <restriction base="xsd:token">
      <enumeration value="major"/>
      <enumeration value="minor"/>
      <enumeration value="none"/>
   </restriction>
</simpleType>
```

5.1.12.24 ST_FOVAngle (Field of View Angle)

Represents a positive angle in 60000ths of a degree. Range from [0, 180] degrees.

This simple type's contents are a restriction of the ST_Angle simple type (§5.1.12.3).

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 0.

- This simple type has a maximum value of less than or equal to 10800000.

Referenced By
camera@fov (§5.1.7.5)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FOVAngle">
   <restriction base="ST_Angle">
      <minInclusive value="0"/>
      <maxInclusive value="10800000"/>
   </restriction>
</simpleType>
```

5.1.12.25 ST_GeomGuideFormula (Geometry Guide Formula Properties)

This simple type specifies a geometry guide formula.

This simple type's contents are a restriction of the XML Schema string datatype.

Referenced By
gd@fmla (§5.1.11.11)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_GeomGuideFormula">
   <restriction base="xsd:string"/>
</simpleType>
```

5.1.12.26 ST_GeomGuideName (Geometry Guide Name Properties)

This simple type specifies a geometry guide name.

This simple type's contents are a restriction of the XML Schema token datatype.

Referenced By
ahPolar@gdRefAng (§5.1.11.2); ahPolar@gdRefR (§5.1.11.2); ahXY@gdRefX (§5.1.11.3); ahXY@gdRefY (§5.1.11.3); gd@name (§5.1.11.11); ST_AdjAngle (§5.1.12.1); ST_AdjCoordinate (§5.1.12.2)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_GeomGuideName">
   <restriction base="xsd:token"/>
</simpleType>
```

5.1.12.27 ST_Guid (GUID Method)

This type specifies a 128 bit GUID

This simple type's contents are a restriction of the XML Schema token datatype.

This simple type also specifies the following restrictions:

- This simple type's contents must match the following regular expression pattern: \{[0-9A-F]{8}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{12}\}.

Referenced By
dgm@id (§5.1.2.1.12); fld@id (§5.1.5.2.4); ST_ModelId (§5.9.7.42); tableStyle@styleId (§5.1.6.9); tableStyleId (§5.1.6.10); tblStyle@styleId (§5.1.4.2.26); tblStyleLst@def (§5.1.4.2.27)

5.1.12.28 ST_HexBinary3 (Hex Binary of Length 3)

This simple type specifies a hex representation of a binary number whose length is 3 characters.

This simple type's contents are a restriction of the XML Schema hexBinary datatype.

This simple type also specifies the following restrictions:

- This simple type's contents must have a length of exactly 3 characters.

Referenced By
srgbClr@val (§5.1.2.2.32); sysClr@lastClr (§5.1.2.2.33)

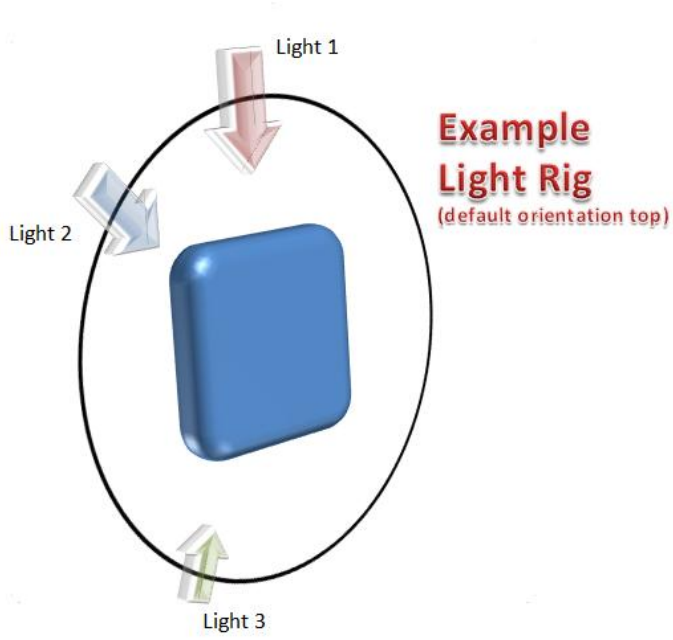
The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_HexBinary3">
   <restriction base="xsd:hexBinary">
      <length value="3"/>
   </restriction>
</simpleType>
```

5.1.12.29 ST_LightRigDirection (Light Rig Direction)

Represents the direction from which the light rig is positioned relative to the scene. The light rig, itself, can be made up of multiple lights in any orientation around a given shape. This simple type defines the orientation of the light rig as a whole, and not the individual lights within the rig. This means that because the direction of the light rig is left, that does not guarantee the light will be coming from the left side of the shape, but rather the orientation of the rig as a whole is rotated to the left.

[*Example*: Consider the following example as a visual representation of a light rig oriented from the top of the shape in the center:



In this example we see that the light rig defines three lights (all in a single plane as represented by the black circular line). The lights defined in this representation can all have different intensities, which means, for this example, Light 3 and Light 2 look to have a more intense effect (or could even be a different color) than Light 1. One can image rotating this rig to the so that Light 1 is to the right of the shape when the light rig direction is defined to be right. *end example*]

The following properties were used to define the shape used in the image examples below:

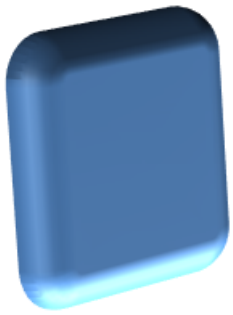
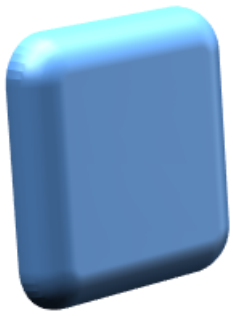
- Rounded rectangle shape
- Three Point light rig type
- Circle bevel type
- Plastic material type
- Camera type defined by the `orthographicFront` preset
- Bevel width and height each equal to `190500`

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
b (Bottom)	[*Example*: Consider the following example of a light direction from the bottom:

Enumeration Value	Description
	 *end example*]
bl (Bottom Left)	[*Example*: Consider the following example of a light direction from the bottom left:  *end example*]
br (Bottom Right)	[*Example*: Consider the following example of a light direction from the bottom right:  *end example*]
l (Left)	[*Example*: Consider the following example of a light direction from the left:

Enumeration Value	Description
	 *end example*]
r (Right)	[*Example*: Consider the following example of a light direction from the right:  *end example*]
t (Top)	[*Example*: Consider the following example of a light direction from the top:  *end example*]
tl (Top Left)	[*Example*: Consider the following example of a light

Enumeration Value	Description
	direction from the top left:  *end example*]
tr (Top Right)	[*Example*: Consider the following example of a light direction from the top right:  *end example*]

Referenced By
lightRig@dir (§5.1.7.9)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_LightRigDirection">
   <restriction base="xsd:token">
      <enumeration value="tl"/>
      <enumeration value="t"/>
      <enumeration value="tr"/>
      <enumeration value="l"/>
      <enumeration value="r"/>
      <enumeration value="bl"/>
      <enumeration value="b"/>
      <enumeration value="br"/>
   </restriction>
</simpleType>
```

5.1.12.30 ST_LightRigType (Light Rig Type)

Represents a preset light right that can be applied to a shape. The light rig represents a group of lights oriented in a specific way relative to a 3D scene. The following properties were used to define the shape used in the image examples below:

- Rounded rectangle shape
- Circle bevel type
- Warm Matte material type
- Camera type defined by the `perspectiveContrastingRightFacing` preset
- Bevel width and height each equal to `190500`

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

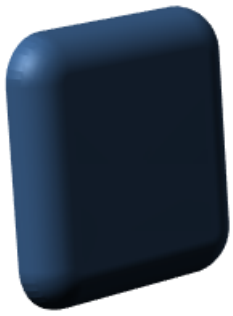
Enumeration Value	Description
balanced (Light Rig Enum (Balanced))	Balanced
brightRoom (Bright Room)	[*Example*: Consider the following example of the `brightRoom` light rig applied to a basic shape: 

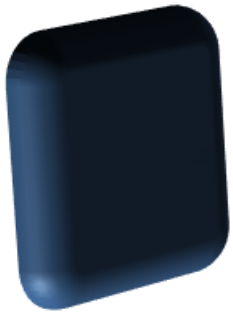
Enumeration Value	Description
	end example]
chilly (Chilly)	[*Example*: Consider the following example of the `chilly` light rig applied to a basic shape:  *end example*]
contrasting (Contrasting)	[*Example*: Consider the following example of the `contrasting` light rig applied to a basic shape:  *end example*]
flat (Flat)	[*Example*: Consider the following example of the `flat` light rig applied to a basic shape: 

Enumeration Value	Description
	end example]
flood (Flood)	[*Example*: Consider the following example of the `flood` light rig applied to a basic shape:  *end example*]
freezing (Freezing)	[*Example*: Consider the following example of the `freezing` light rig applied to a basic shape:  *end example*]
glow (Glow)	[*Example*: Consider the following example of the `glow` light rig applied to a basic shape:

Enumeration Value	Description
	 *end example*]
harsh (Harsh)	[*Example*: Consider the following example of the `harsh` light rig applied to a basic shape:  *end example*]
legacyFlat1 (Legacy Flat 1)	[*Example*: Consider the following example of the `legacyFlat1` light rig applied to a basic shape:  *end example*]
legacyFlat2 (Legacy Flat 2)	[*Example*: Consider the following example of the `legacyFlat2` light rig applied to a basic shape:

Enumeration Value	Description
	 *end example*]
legacyFlat3 (Legacy Flat 3)	[*Example*: Consider the following example of the `legacyFlat3` light rig applied to a basic shape:  *end example*]
legacyFlat4 (Legacy Flat 4)	[*Example*: Consider the following example of the `legacyFlat4` light rig applied to a basic shape:  *end example*]
legacyHarsh1 (Legacy Harsh 1)	[*Example*: Consider the following example of the `legacyHarsh1` light rig applied to a basic shape:

Enumeration Value	Description
	 *end example*]
legacyHarsh2 (Legacy Harsh 2)	[*Example*: Consider the following example of the `legacyHarsh2` light rig applied to a basic shape:  *end example*]
legacyHarsh3 (Legacy Harsh 3)	[*Example*: Consider the following example of the `legacyHarsh3` light rig applied to a basic shape:  *end example*]
legacyHarsh4 (Legacy Harsh 4)	[*Example*: Consider the following example of the

Enumeration Value	Description
	`legacyHarsh4` light rig applied to a basic shape:  *end example*]
legacyNormal1 (Legacy Normal 1)	[*Example*: Consider the following example of the `legacyNormal1` light rig applied to a basic shape:  *end example*]
legacyNormal2 (Legacy Normal 2)	[*Example*: Consider the following example of the `legacyNormal2` light rig applied to a basic shape:  *end example*]

Enumeration Value	Description
legacyNormal3 (Legacy Normal 3)	[*Example*: Consider the following example of the `legacyNormal3` light rig applied to a basic shape:  *end example*]
legacyNormal4 (Legacy Normal 4)	[*Example*: Consider the following example of the `legacyNormal4` light rig applied to a basic shape:  *end example*]
morning (Morning)	[*Example*: Consider the following example of the `morning` light rig applied to a basic shape:  *end example*]

Enumeration Value	Description
soft (Soft)	[*Example*: Consider the following example of the `soft` light rig applied to a basic shape:  *end example*]
sunrise (Sunrise)	[*Example*: Consider the following example of the `sunrise` light rig applied to a basic shape:  *end example*]
sunset (Sunset)	[*Example*: Consider the following example of the `sunset` light rig applied to a basic shape:  *end example*]

Enumeration Value	Description
threePt (Three Point)	[*Example*: Consider the following example of the `threePt` light rig applied to a basic shape:  *end example*]
twoPt (Two Point)	[*Example*: Consider the following example of the `twoPt` light rig applied to a basic shape:  *end example*]

Referenced By
lightRig@rig (§5.1.7.9)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_LightRigType">
   <restriction base="xsd:token">
      <enumeration value="legacyFlat1"/>
      <enumeration value="legacyFlat2"/>
      <enumeration value="legacyFlat3"/>
      <enumeration value="legacyFlat4"/>
      <enumeration value="legacyNormal1"/>
      <enumeration value="legacyNormal2"/>
      <enumeration value="legacyNormal3"/>
      <enumeration value="legacyNormal4"/>
      <enumeration value="legacyHarsh1"/>
      <enumeration value="legacyHarsh2"/>
      <enumeration value="legacyHarsh3"/>
      <enumeration value="legacyHarsh4"/>
      <enumeration value="threePt"/>
      <enumeration value="balanced"/>
      <enumeration value="soft"/>
      <enumeration value="harsh"/>
      <enumeration value="flood"/>
      <enumeration value="contrasting"/>
      <enumeration value="morning"/>
      <enumeration value="sunrise"/>
      <enumeration value="sunset"/>
      <enumeration value="chilly"/>
      <enumeration value="freezing"/>
      <enumeration value="flat"/>
      <enumeration value="twoPt"/>
      <enumeration value="glow"/>
      <enumeration value="brightRoom"/>
   </restriction>
</simpleType>
```

5.1.12.31 ST_LineCap (End Line Cap)

This type specifies how to cap the ends of lines. This also affects the ends of line segments for dashed lines.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
flat (Flat Line Cap)	Line ends at end point.
rnd (Round Line Cap)	Rounded ends. Semi-circle protrudes by half line width.
sq (Square Line Cap)	Square protrudes by half line width.

Referenced By

Referenced By
ln@cap (§5.1.2.1.24); lnB@cap (§5.1.6.3); lnBlToTr@cap (§5.1.6.4); lnL@cap (§5.1.6.5); lnR@cap (§5.1.6.6); lnT@cap (§5.1.6.7); lnTlToBr@cap (§5.1.6.8); uLn@cap (§5.1.5.3.14)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_LineCap">
   <restriction base="xsd:token">
      <enumeration value="rnd"/>
      <enumeration value="sq"/>
      <enumeration value="flat"/>
   </restriction>
</simpleType>
```

5.1.12.32 ST_LineEndLength (Line End Length)

This simple type represents the length of the line end decoration (e.g., arrowhead) relative to the width of the line itself.

[*Example:* See the example images below. These samples have an arrow line end type and medium line end width. *end example*]

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
lg (Large)	Large
med (Medium)	Medium
sm (Small)	Small

Referenced By
headEnd@len (§5.1.10.38); tailEnd@len (§5.1.10.57)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_LineEndLength">
   <restriction base="xsd:token">
      <enumeration value="sm"/>
      <enumeration value="med"/>
      <enumeration value="lg"/>
   </restriction>
</simpleType>
```

5.1.12.33 ST_LineEndType (Line End Type)

This simple type represents the shape decoration that appears at the ends of lines. For example, one choice is an arrow head.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
arrow (Arrow Head)	 Line arrow head
diamond (Diamond)	 Diamond
none (None)	No end
oval (Oval)	 Oval
stealth (Stealth Arrow)	 Stealth arrow head
triangle (Triangle Arrow Head)	 Triangle arrow head

Referenced By
headEnd@type (§5.1.10.38); tailEnd@type (§5.1.10.57)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_LineEndType">
   <restriction base="xsd:token">
      <enumeration value="none"/>
      <enumeration value="triangle"/>
      <enumeration value="stealth"/>
      <enumeration value="diamond"/>
      <enumeration value="oval"/>
      <enumeration value="arrow"/>
   </restriction>
</simpleType>
```

5.1.12.34 ST_LineEndWidth (Line End Width)

This simple type represents the width of the line end decoration (e.g., arrowhead) relative to the width of the line itself.

[*Example:* See the example images below. These samples have an arrow line end type and medium line end length. *end example*]

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description

Enumeration Value	Description
lg (Large)	→ Large
med (Medium)	→ Medium
sm (Small)	→ Small

Referenced By
headEnd@w (§5.1.10.38); tailEnd@w (§5.1.10.57)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_LineEndWidth">
   <restriction base="xsd:token">
      <enumeration value="sm"/>
      <enumeration value="med"/>
      <enumeration value="lg"/>
   </restriction>
</simpleType>
```

5.1.12.35 ST_LineWidth (Line Width)

This type specifies the width of a line in EMUs. 1 pt = 12700 EMUs.

This simple type's contents are a restriction of the ST_Coordinate32 simple type (§5.1.12.17).

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 0.
- This simple type has a maximum value of less than or equal to 20116800.

Referenced By
ln@w (§5.1.2.1.24); lnB@w (§5.1.6.3); lnBlToTr@w (§5.1.6.4); lnL@w (§5.1.6.5); lnR@w (§5.1.6.6); lnT@w (§5.1.6.7); lnTlToBr@w (§5.1.6.8); uLn@w (§5.1.5.3.14)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_LineWidth">
   <restriction base="ST_Coordinate32">
      <minInclusive value="0"/>
      <maxInclusive value="20116800"/>
   </restriction>
</simpleType>
```

5.1.12.36 ST_OnOffStyleType (On/Off Style Type)

This simple type represents whether a style property should be applied.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
def (Default)	Follow parent settings. For a themed property, follow the theme settings. For an unthemed property, follow the parent setting in the property inheritance chain.
off (Off)	Property is off.
on (On)	Property is on.

Referenced By
tcTxStyle@b (§5.1.4.2.30); tcTxStyle@i (§5.1.4.2.30)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_OnOffStyleType">
   <restriction base="xsd:token">
      <enumeration value="on"/>
      <enumeration value="off"/>
      <enumeration value="def"/>
   </restriction>
</simpleType>
```

5.1.12.37 ST_Panose (Panose Type)

This type specifies the Panose setting for this font so that generating applications using this Office Open XML Standard may determine the closest font type if necessary.

This simple type's contents are a restriction of the XML Schema hexBinary datatype.

This simple type also specifies the following restrictions:

- This simple type's contents must have a length of exactly 10 characters.

Referenced By
buFont@panose (§5.1.5.4.6); cs@panose (§5.1.5.3.1); ea@panose (§5.1.5.3.3); font@panose (§4.3.1.10); latin@panose (§5.1.5.3.7); sym@panose (§5.1.5.3.10)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Panose">
   <restriction base="xsd:hexBinary">
      <length value="10"/>
   </restriction>
</simpleType>
```

5.1.12.38 ST_PathFillMode (Path Fill Mode)

This simple type specifies the manner in which a path should be filled. The lightening and darkening of a path allow for certain parts of the shape to be colored lighter of darker depending on user preference.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
darken (Darken Path Fill)	This specifies that the corresponding path should have a darker shaded color applied to it's fill.
darkenLess (Darken Path Fill Less)	This specifies that the corresponding path should have a slightly darker shaded color applied to it's fill.
lighten (Lighten Path Fill)	This specifies that the corresponding path should have a lightly shaded color applied to it's fill.
lightenLess (Lighten Path Fill Less)	This specifies that the corresponding path should have a slightly lighter shaded color applied to it's fill.
none (No Path Fill)	This specifies that the corresponding path should have no fill.
norm (Normal Path Fill)	This specifies that the corresponding path should have a normally shaded color applied to it's fill.

Referenced By
path@fill (§5.1.11.15)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PathFillMode">
   <restriction base="xsd:token">
      <enumeration value="none"/>
      <enumeration value="norm"/>
      <enumeration value="lighten"/>
      <enumeration value="lightenLess"/>
      <enumeration value="darken"/>
      <enumeration value="darkenLess"/>
   </restriction>
</simpleType>
```

5.1.12.39 ST_PathShadeType (Path Shade Type)

This simple type describes the shape of path to follow for a path gradient shade.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
circle (Circle)	Gradient follows a circular path
rect (Rectangle)	Gradient follows a rectangular path
shape (Shape)	Gradient follows the shape

Referenced By
path@path (§5.1.10.46)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PathShadeType">
   <restriction base="xsd:token">
      <enumeration value="shape"/>
      <enumeration value="circle"/>
      <enumeration value="rect"/>
   </restriction>
</simpleType>
```

5.1.12.40 ST_PenAlignment (Alignment Type)

This type specifies the Pen Alignment type for use within a text body.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
ctr (Center Alignment)	Center pen (line drawn at center of path stroke).
in (Inset Alignment)	Inset pen (the pen is aligned on the inside of the edge of the path).

Referenced By
ln@algn (§5.1.2.1.24); lnB@algn (§5.1.6.3); lnBlToTr@algn (§5.1.6.4); lnL@algn (§5.1.6.5); lnR@algn (§5.1.6.6); lnT@algn (§5.1.6.7); lnTlToBr@algn (§5.1.6.8); uLn@algn (§5.1.5.3.14)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PenAlignment">
   <restriction base="xsd:token">
      <enumeration value="ctr"/>
      <enumeration value="in"/>
   </restriction>
</simpleType>
```

5.1.12.41 ST_Percentage (Percentage)

This simple type represents a percentage in 1000ths of a percent, e.g., a value of 1 represents 0.001% == 0.00001; a value of 100000 is equal to 100%. Percentages have no intrinsic units, but are used to scale other values with units.

This simple type's contents are a restriction of the XML Schema int datatype.

Referenced By
blue@val (§5.1.2.2.4); blueMod@val (§5.1.2.2.5); blueOff@val (§5.1.2.2.6); by@x (§4.6.20); by@y (§4.6.20); cTn@spd (§4.6.33); defRPr@baseline (§5.1.5.3.2); endParaRPr@baseline (§5.1.5.2.3); fillRect@b (§5.1.10.30); fillRect@l (§5.1.10.30); fillRect@r (§5.1.10.30); fillRect@t (§5.1.10.30); fillToRect@b (§5.1.10.31); fillToRect@l (§5.1.10.31); fillToRect@r (§5.1.10.31); fillToRect@t (§5.1.10.31); from@x (§4.6.43); from@y (§4.6.43); green@val (§5.1.2.2.10); greenMod@val (§5.1.2.2.11); greenOff@val (§5.1.2.2.12); hslClr@lum (§5.1.2.2.13); hslClr@sat (§5.1.2.2.13); lum@val (§5.1.2.2.19); lumMod@val (§5.1.2.2.20); lumOff@val (§5.1.2.2.21); outerShdw@sx (§5.1.10.45); outerShdw@sy (§5.1.10.45); presentation@serverZoom (§4.3.1.24); rCtr@x (§4.6.62); rCtr@y (§4.6.62); red@val (§5.1.2.2.23); redMod@val (§5.1.2.2.24); redOff@val (§5.1.2.2.25); reflection@sx (§5.1.10.50); reflection@sy (§5.1.10.50); relOff@tx (§5.1.10.51); relOff@ty (§5.1.10.51); rPr@baseline (§5.1.5.3.9); sat@val (§5.1.2.2.26); satMod@val (§5.1.2.2.27); satOff@val (§5.1.2.2.28); scrgbClr@b (§5.1.2.2.30); scrgbClr@g (§5.1.2.2.30); scrgbClr@r (§5.1.2.2.30); srcRect@b (§5.1.10.55); srcRect@l (§5.1.10.55); srcRect@r (§5.1.10.55); srcRect@t (§5.1.10.55); ST_FixedPercentage (§5.1.12.22); ST_PositiveFixedPercentage (§5.1.12.45); ST_PositivePercentage (§5.1.12.46); ST_TextBulletSizePercent (§5.1.12.62); ST_TextFontScalePercent (§5.1.12.67); ST_TextSpacingPercent (§5.1.12.77); tile@sx (§5.1.10.58); tile@sy (§5.1.10.58); tileRect@b (§5.1.10.59); tileRect@l (§5.1.10.59); tileRect@r (§5.1.10.59); tileRect@t (§5.1.10.59); to@x (§4.6.88); to@y (§4.6.88); xfrm@sx (§5.1.10.61); xfrm@sy (§5.1.10.61)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Percentage">
   <restriction base="xsd:int"/>
</simpleType>
```

5.1.12.42 ST_PositiveCoordinate (Positive Coordinate)

This simple type represents a positive position or length in EMUs.

This simple type's contents are a restriction of the XML Schema long datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 0.
- This simple type has a maximum value of less than or equal to 27273042316900.

Referenced By
bevel@h (§5.1.4.2.5); bevel@w (§5.1.4.2.5); bevelB@h (§5.1.7.3); bevelB@w (§5.1.7.3); bevelT@h (§5.1.7.4); bevelT@w (§5.1.7.4); blur@rad (§5.1.10.15); chExt@cx (§5.1.9.1); chExt@cy (§5.1.9.1); ext@cx (§5.1.9.3); ext@cx (§5.6.2.13); ext@cx (§5.8.2.10); ext@cy (§5.1.9.3); ext@cy (§5.6.2.13); ext@cy (§5.8.2.10); extent@cx (§5.5.2.7); extent@cy (§5.5.2.7); glow@rad (§5.1.10.32); gridSpacing@cx (§4.3.2.3); gridSpacing@cy

Referenced By
(§4.3.2.3); innerShdw@blurRad (§5.1.10.40); innerShdw@dist (§5.1.10.40); notesSz@cx (§4.3.1.20); notesSz@cy (§4.3.1.20); outerShdw@blurRad (§5.1.10.45); outerShdw@dist (§5.1.10.45); path@h (§5.1.11.15); path@w (§5.1.11.15); prstShdw@dist (§5.1.10.49); reflection@blurRad (§5.1.10.50); reflection@dist (§5.1.10.50); softEdge@rad (§5.1.10.53); sp3d@contourW (§5.1.7.12); sp3d@contourW (§5.9.5.6); sp3d@extrusionH (§5.1.7.12); sp3d@extrusionH (§5.9.5.6)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PositiveCoordinate">
   <restriction base="xsd:long">
      <minInclusive value="0"/>
      <maxInclusive value="27273042316900"/>
   </restriction>
</simpleType>
```

5.1.12.43 ST_PositiveCoordinate32 (Positive Coordinate Point)

This type specifies the a positive coordinate point that has a maximum size of 32 bits.

NOTE: The units of measurement used here are EMUs (English Metric Units).

This simple type's contents are a restriction of the ST_Coordinate32 simple type (§5.1.12.17).

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 0.

Referenced By
bodyPr@spcCol (§5.1.5.1.1); control@imgH (§4.4.2.1); control@imgW (§4.4.2.1); oleObj@imgH (§4.4.2.4); oleObj@imgW (§4.4.2.4); ST_SlideSizeCoordinate (§4.8.21)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PositiveCoordinate32">
   <restriction base="ST_Coordinate32">
      <minInclusive value="0"/>
   </restriction>
</simpleType>
```

5.1.12.44 ST_PositiveFixedAngle (Positive Fixed Angle)

This simple type represents a positive angle in 60000ths of a degree. Range from [0, 360 degrees).

This simple type's contents are a restriction of the ST_Angle simple type (§5.1.12.3).

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 0.
- This simple type has a maximum value of less than 21600000.

Referenced By
hsl@hue (§5.1.10.39); hslClr@hue (§5.1.2.2.13); hue@val (§5.1.2.2.14); innerShdw@dir (§5.1.10.40); lin@ang (§5.1.10.41); outerShdw@dir (§5.1.10.45); prstShdw@dir (§5.1.10.49); reflection@dir (§5.1.10.50); reflection@fadeDir (§5.1.10.50); rot@lat (§5.1.7.11); rot@lon (§5.1.7.11); rot@rev (§5.1.7.11); tint@hue (§5.1.10.60)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PositiveFixedAngle">
   <restriction base="ST_Angle">
      <minInclusive value="0"/>
      <maxExclusive value="21600000"/>
   </restriction>
</simpleType>
```

5.1.12.45 ST_PositiveFixedPercentage (Positive Fixed Percentage)

This simple type represents a positive fixed percentage in 1000ths of a percent. Range from [0%, 100%].

This simple type's contents are a restriction of the ST_Percentage simple type (§5.1.12.41).

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 0.
- This simple type has a maximum value of less than or equal to 100000.

Referenced By
alpha@val (§5.1.2.2.1); alphaBiLevel@thresh (§5.1.10.1); alphaRepl@a (§5.1.10.8); biLevel@thresh (§5.1.10.11); cMediaNode@vol (§4.6.29); cTn@accel (§4.6.33); cTn@decel (§4.6.33); gs@pos (§5.1.10.36); reflection@endA (§5.1.10.50); reflection@endPos (§5.1.10.50); reflection@stA (§5.1.10.50); reflection@stPos (§5.1.10.50); restoredLeft@sz (§4.3.2.11); restoredTop@sz (§4.3.2.12); shade@val (§5.1.2.2.31); ST_TLTimeAnimateValueTime (§4.8.43); tint@val (§5.1.2.2.34)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PositiveFixedPercentage">
   <restriction base="ST_Percentage">
      <minInclusive value="0"/>
      <maxInclusive value="100000"/>
   </restriction>
</simpleType>
```

5.1.12.46 ST_PositivePercentage (Positive Percentage)

This simple type represents a positive percentage in 1000ths of a percent. Range from [0%, inf).

This simple type's contents are a restriction of the ST_Percentage simple type (§5.1.12.41).

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 0.

Referenced By
alphaMod@val (§5.1.2.2.2); alphaModFix@amt (§5.1.10.6); camera@zoom (§5.1.7.5); ds@d (§5.1.10.22); ds@sp (§5.1.10.22); hueMod@val (§5.1.2.2.15); miter@lim (§5.1.10.43); tmPct@val (§4.6.83)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PositivePercentage">
   <restriction base="ST_Percentage">
      <minInclusive value="0"/>
   </restriction>
</simpleType>
```

5.1.12.47 ST_PresetCameraType (Preset Camera Type)

These enumeration values represent different algorithmic methods for setting all camera properties, including position. The following example images below are all based off the following shape:

In this image, we can see the shape has a camera pointing directly at the front face.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
isometricBottomDown (Isometric Bottom Down)	[*Example*: Consider the following example of the camera preset type:  *end example*]
isometricBottomUp (Isometric Bottom Up)	[*Example*: Consider the following example of the

Enumeration Value	Description
	camera preset type:  *end example*]
isometricLeftDown (Isometric Left Down)	[*Example*: Consider the following example of the camera preset type:  *end example*]
isometricLeftUp (Isometric Left Up)	[*Example*: Consider the following example of the camera preset type:  *end example*]

Enumeration Value	Description
isometricOffAxis1Left (Isometric Off Axis 1 Left)	[*Example*: Consider the following example of the camera preset type:  *end example*]
isometricOffAxis1Right (Isometric Off Axis 1 Right)	[*Example*: Consider the following example of the camera preset type:  *end example*]
isometricOffAxis1Top (Isometric Off Axis 1 Top)	[*Example*: Consider the following example of the camera preset type:  *end example*]
isometricOffAxis2Left (Isometric Off Axis 2 Left)	[*Example*: Consider the following example of the

Enumeration Value	Description
	camera preset type:  *end example*]
isometricOffAxis2Right (Isometric Off Axis 2 Right)	[*Example*: Consider the following example of the camera preset type:  *end example*]
isometricOffAxis2Top (Isometric Off Axis 2 Top)	[*Example*: Consider the following example of the camera preset type:  *end example*]
isometricOffAxis3Bottom (Isometric Off Axis 3 Bottom)	[*Example*: Consider the following example of the camera preset type:

Enumeration Value	Description
	 *end example*]
isometricOffAxis3Left (Isometric Off Axis 3 Left)	[*Example*: Consider the following example of the camera preset type:  *end example*]
isometricOffAxis3Right (Isometric Off Axis 3 Right)	[*Example*: Consider the following example of the camera preset type:  *end example*]
isometricOffAxis4Bottom (Isometric Off Axis 4 Bottom)	[*Example*: Consider the following example of the camera preset type:

Enumeration Value	Description
	 *end example*]
isometricOffAxis4Left (Isometric Off Axis 4 Left)	[*Example*: Consider the following example of the camera preset type:  *end example*]
isometricOffAxis4Right (Isometric Off Axis 4 Right)	[*Example*: Consider the following example of the camera preset type:  *end example*]
isometricRightDown (Isometric Right Down)	[*Example*: Consider the following example of the camera preset type:

Enumeration Value	Description
	 *end example*]
isometricRightUp (Isometric Right Up)	[*Example*: Consider the following example of the camera preset type:  *end example*]
isometricTopDown (Isometric Top Down)	[*Example*: Consider the following example of the camera preset type:  *end example*]
isometricTopUp (Isometric Top Up)	[*Example*: Consider the following example of the

Enumeration Value	Description
	camera preset type:  *end example*]
legacyObliqueBottom (Legacy Oblique Bottom)	[*Example*: Consider the following example of the camera preset type:  *end example*]
legacyObliqueBottomLeft (Legacy Oblique Bottom Left)	[*Example*: Consider the following example of the camera preset type:  *end example*]
legacyObliqueBottomRight (Legacy Oblique Bottom Right)	[*Example*: Consider the following example of the camera preset type:

Enumeration Value	Description
	 *end example*]
legacyObliqueFront (Legacy Oblique Front)	[*Example*: Consider the following example of the camera preset type:  *end example*]
legacyObliqueLeft (Legacy Oblique Left)	[*Example*: Consider the following example of the camera preset type:  *end example*]
legacyObliqueRight (Legacy Oblique Right)	[*Example*: Consider the following example of the camera preset type:

Enumeration Value	Description
	 *end example*]
legacyObliqueTop (Legacy Oblique Top)	[*Example*: Consider the following example of the camera preset type:  *end example*]
legacyObliqueTopLeft (Legacy Oblique Top Left)	[*Example*: Consider the following example of the camera preset type:  *end example*]
legacyObliqueTopRight (Legacy Oblique Top Right)	[*Example*: Consider the following example of the camera preset type:

Enumeration Value	Description
	 *end example*]
legacyPerspectiveBottom (Legacy Perspective Bottom)	[*Example*: Consider the following example of the camera preset type:  *end example*]
legacyPerspectiveBottomLeft (Legacy Perspective Bottom Left)	[*Example*: Consider the following example of the camera preset type:  *end example*]
legacyPerspectiveBottomRight (Legacy Perspective Bottom Right)	[*Example*: Consider the following example of the camera preset type:

Enumeration Value	Description
	 *end example*]
legacyPerspectiveFront (Legacy Perspective Front)	[*Example*: Consider the following example of the camera preset type:  *end example*]
legacyPerspectiveLeft (Legacy Perspective Left)	[*Example*: Consider the following example of the camera preset type:  *end example*]
legacyPerspectiveRight (Legacy Perspective Right)	[*Example*: Consider the following example of the camera preset type:

Enumeration Value	Description
	 *end example*]
legacyPerspectiveTop (Legacy Perspective Top)	[*Example*: Consider the following example of the camera preset type:  *end example*]
legacyPerspectiveTopLeft (Legacy Perspective Top Left)	[*Example*: Consider the following example of the camera preset type:  *end example*]
legacyPerspectiveTopRight (Legacy Perspective Top Right)	[*Example*: Consider the following example of the camera preset type:

Enumeration Value	Description
	 *end example*]
obliqueBottom (Oblique Bottom)	[*Example*: Consider the following example of the camera preset type:  *end example*]
obliqueBottomLeft (Oblique Bottom Left)	[*Example*: Consider the following example of the camera preset type:  *end example*]
obliqueBottomRight (Oblique Bottom Right)	[*Example*: Consider the following example of the camera preset type:

Enumeration Value	Description
	 *end example*]
obliqueLeft (Oblique Left)	[*Example*: Consider the following example of the camera preset type:  *end example*]
obliqueRight (Oblique Right)	[*Example*: Consider the following example of the camera preset type:  *end example*]
obliqueTop (Oblique Top)	[*Example*: Consider the following example of the camera preset type:

Enumeration Value	Description
	 *end example*]
obliqueTopLeft (Oblique Top Left)	[*Example*: Consider the following example of the camera preset type:  *end example*]
obliqueTopRight (Oblique Top Right)	[*Example*: Consider the following example of the camera preset type:  *end example*]
orthographicFront (Orthographic Front)	[*Example*: Consider the following example of the camera preset type:

Enumeration Value	Description
	 *end example*]
perspectiveAbove (Orthographic Above)	[*Example*: Consider the following example of the camera preset type:  *end example*]
perspectiveAboveLeftFacing (Perspective Above Left Facing)	[*Example*: Consider the following example of the camera preset type:  *end example*]
perspectiveAboveRightFacing (Perspective Above Right Facing)	[*Example*: Consider the following example of the camera preset type:

Enumeration Value	Description
	 *end example*]
perspectiveBelow (Perspective Below)	[*Example*: Consider the following example of the camera preset type:  *end example*]
perspectiveContrastingLeftFacing (Perspective Contrasting Left Facing)	[*Example*: Consider the following example of the camera preset type:  *end example*]
perspectiveContrastingRightFacing (Perspective Contrasting Right Facing)	[*Example*: Consider the following example of the camera preset type:

Enumeration Value	Description
	 *end example*]
perspectiveFront (Perspective Front)	[*Example*: Consider the following example of the camera preset type:  *end example*]
perspectiveHeroicExtremeLeftFacing (Perspective Heroic Extreme Left Facing)	[*Example*: Consider the following example of the camera preset type:  *end example*]
perspectiveHeroicExtremeRightFacing (Perspective Heroic Extreme Right Facing)	[*Example*: Consider the following example of the camera preset type:

Enumeration Value	Description
	 *end example*]
perspectiveHeroicLeftFacing (Perspective Heroic Left Facing)	[*Example*: Consider the following example of the camera preset type:  *end example*]
perspectiveHeroicRightFacing (Perspective Heroic Right Facing)	[*Example*: Consider the following example of the camera preset type:  *end example*]
perspectiveLeft (Perspective Left)	[*Example*: Consider the following example of the camera preset type:

Enumeration Value	Description
	 *end example*]
perspectiveRelaxed (Perspective Relaxed)	[*Example*: Consider the following example of the camera preset type:  *end example*]
perspectiveRelaxedModerately (Perspective Relaxed Moderately)	[*Example*: Consider the following example of the camera preset type:  *end example*]
perspectiveRight (Perspective Right)	[*Example*: Consider the following example of the camera preset type:

Enumeration Value	Description
	 *end example*]

Referenced By
camera@prst (§5.1.7.5)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PresetCameraType">
   <restriction base="xsd:token">
      <enumeration value="legacyObliqueTopLeft"/>
      <enumeration value="legacyObliqueTop"/>
      <enumeration value="legacyObliqueTopRight"/>
      <enumeration value="legacyObliqueLeft"/>
      <enumeration value="legacyObliqueFront"/>
      <enumeration value="legacyObliqueRight"/>
      <enumeration value="legacyObliqueBottomLeft"/>
      <enumeration value="legacyObliqueBottom"/>
      <enumeration value="legacyObliqueBottomRight"/>
      <enumeration value="legacyPerspectiveTopLeft"/>
      <enumeration value="legacyPerspectiveTop"/>
      <enumeration value="legacyPerspectiveTopRight"/>
      <enumeration value="legacyPerspectiveLeft"/>
      <enumeration value="legacyPerspectiveFront"/>
      <enumeration value="legacyPerspectiveRight"/>
      <enumeration value="legacyPerspectiveBottomLeft"/>
      <enumeration value="legacyPerspectiveBottom"/>
      <enumeration value="legacyPerspectiveBottomRight"/>
      <enumeration value="orthographicFront"/>
      <enumeration value="isometricTopUp"/>
      <enumeration value="isometricTopDown"/>
      <enumeration value="isometricBottomUp"/>
      <enumeration value="isometricBottomDown"/>
      <enumeration value="isometricLeftUp"/>
      <enumeration value="isometricLeftDown"/>
      <enumeration value="isometricRightUp"/>
      <enumeration value="isometricRightDown"/>
      <enumeration value="isometricOffAxis1Left"/>
      <enumeration value="isometricOffAxis1Right"/>
      <enumeration value="isometricOffAxis1Top"/>
      <enumeration value="isometricOffAxis2Left"/>
      <enumeration value="isometricOffAxis2Right"/>
      <enumeration value="isometricOffAxis2Top"/>
      <enumeration value="isometricOffAxis3Left"/>
      <enumeration value="isometricOffAxis3Right"/>
      <enumeration value="isometricOffAxis3Bottom"/>
      <enumeration value="isometricOffAxis4Left"/>
      <enumeration value="isometricOffAxis4Right"/>
      <enumeration value="isometricOffAxis4Bottom"/>
      <enumeration value="obliqueTopLeft"/>
      <enumeration value="obliqueTop"/>
      <enumeration value="obliqueTopRight"/>
      <enumeration value="obliqueLeft"/>
      <enumeration value="obliqueRight"/>
      <enumeration value="obliqueBottomLeft"/>
      <enumeration value="obliqueBottom"/>
      <enumeration value="obliqueBottomRight"/>
      <enumeration value="perspectiveFront"/>
      <enumeration value="perspectiveLeft"/>
      <enumeration value="perspectiveRight"/>
```

```
        <enumeration value="perspectiveAbove"/>
        <enumeration value="perspectiveBelow"/>
        <enumeration value="perspectiveAboveLeftFacing"/>
        <enumeration value="perspectiveAboveRightFacing"/>
        <enumeration value="perspectiveContrastingLeftFacing"/>
        <enumeration value="perspectiveContrastingRightFacing"/>
        <enumeration value="perspectiveHeroicLeftFacing"/>
        <enumeration value="perspectiveHeroicRightFacing"/>
        <enumeration value="perspectiveHeroicExtremeLeftFacing"/>
        <enumeration value="perspectiveHeroicExtremeRightFacing"/>
        <enumeration value="perspectiveRelaxed"/>
        <enumeration value="perspectiveRelaxedModerately"/>
    </restriction>
</simpleType>
```

5.1.12.48 ST_PresetColorVal (Preset Color Value)

This simple type represents a preset color value.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
aliceBlue (Alice Blue Preset Color)	Specifies a color with RGB value (240,248,255)
antiqueWhite (Antique White Preset Color)	Specifies a color with RGB value (250,235,215)
aqua (Aqua Preset Color)	Specifies a color with RGB value (0,255,255)
aquamarine (Aquamarine Preset Color)	Specifies a color with RGB value (127,255,212)
azure (Azure Preset Color)	Specifies a color with RGB value (240,255,255)
beige (Beige Preset Color)	Specifies a color with RGB value (245,245,220)
bisque (Bisque Preset Color)	Specifies a color with RGB value (255,228,196)
black (Black Preset Color)	Specifies a color with RGB value (0,0,0)
blanchedAlmond (Blanched Almond Preset Color)	Specifies a color with RGB value (255,235,205)
blue (Blue Preset Color)	Specifies a color with RGB value (0,0,255)
blueViolet (Blue Violet Preset Color)	Specifies a color with RGB value (138,43,226)
brown (Brown Preset Color)	Specifies a color with RGB value (165,42,42)
burlyWood (Burly Wood Preset Color)	Specifies a color with RGB value (222,184,135)
cadetBlue (Cadet Blue Preset Color)	Specifies a color with RGB value (95,158,160)
chartreuse (Chartreuse Preset Color)	Specifies a color with RGB value (127,255,0)
chocolate (Chocolate Preset Color)	Specifies a color with RGB value (210,105,30)
coral (Coral Preset Color)	Specifies a color with RGB value (255,127,80)
cornflowerBlue (Cornflower Blue Preset Color)	Specifies a color with RGB value (100,149,237)
cornsilk (Cornsilk Preset Color)	Specifies a color with RGB value (255,248,220)

Enumeration Value	Description
crimson (Crimson Preset Color)	Specifies a color with RGB value (220,20,60)
cyan (Cyan Preset Color)	Specifies a color with RGB value (0,255,255)
deepPink (Deep Pink Preset Color)	Specifies a color with RGB value (255,20,147)
deepSkyBlue (Deep Sky Blue Preset Color)	Specifies a color with RGB value (0,191,255)
dimGray (Dim Gray Preset Color)	Specifies a color with RGB value (105,105,105)
dkBlue (Dark Blue Preset Color)	Specifies a color with RGB value (0,0,139)
dkCyan (Dark Cyan Preset Color)	Specifies a color with RGB value (0,139,139)
dkGoldenrod (Dark Goldenrod Preset Color)	Specifies a color with RGB value (184,134,11)
dkGray (Dark Gray Preset Color)	Specifies a color with RGB value (169,169,169)
dkGreen (Dark Green Preset Color)	Specifies a color with RGB value (0,100,0)
dkKhaki (Dark Khaki Preset Color)	Specifies a color with RGB value (189,183,107)
dkMagenta (Dark Magenta Preset Color)	Specifies a color with RGB value (139,0,139)
dkOliveGreen (Dark Olive Green Preset Color)	Specifies a color with RGB value (85,107,47)
dkOrange (Dark Orange Preset Color)	Specifies a color with RGB value (255,140,0)
dkOrchid (Dark Orchid Preset Color)	Specifies a color with RGB value (153,50,204)
dkRed (Dark Red Preset Color)	Specifies a color with RGB value (139,0,0)
dkSalmon (Dark Salmon Preset Color)	Specifies a color with RGB value (233,150,122)
dkSeaGreen (Dark Sea Green Preset Color)	Specifies a color with RGB value (143,188,139)
dkSlateBlue (Dark Slate Blue Preset Color)	Specifies a color with RGB value (72,61,139)
dkSlateGray (Dark Slate Gray Preset Color)	Specifies a color with RGB value (47,79,79)
dkTurquoise (Dark Turquoise Preset Color)	Specifies a color with RGB value (0,206,209)
dkViolet (Dark Violet Preset Color)	Specifies a color with RGB value (148,0,211)
dodgerBlue (Dodger Blue Preset Color)	Specifies a color with RGB value (30,144,255)
firebrick (Firebrick Preset Color)	Specifies a color with RGB value (178,34,34)
floralWhite (Floral White Preset Color)	Specifies a color with RGB value (255,250,240)
forestGreen (Forest Green Preset Color)	Specifies a color with RGB value (34,139,34)
fuchsia (Fuchsia Preset Color)	Specifies a color with RGB value (255,0,255)
gainsboro (Gainsboro Preset Color)	Specifies a color with RGB value (220,220,220)
ghostWhite (Ghost White Preset Color)	Specifies a color with RGB value (248,248,255)
gold (Gold Preset Color)	Specifies a color with RGB value (255,215,0)
goldenrod (Goldenrod Preset Color)	Specifies a color with RGB value (218,165,32)
gray (Gray Preset Color)	Specifies a color with RGB value (128,128,128)
green (Green Preset Color)	Specifies a color with RGB value (0,128,0)
greenYellow (Green Yellow Preset Color)	Specifies a color with RGB value (173,255,47)

Enumeration Value	Description
honeydew (Honeydew Preset Color)	Specifies a color with RGB value (240,255,240)
hotPink (Hot Pink Preset Color)	Specifies a color with RGB value (255,105,180)
indianRed (Indian Red Preset Color)	Specifies a color with RGB value (205,92,92)
indigo (Indigo Preset Color)	Specifies a color with RGB value (75,0,130)
ivory (Ivory Preset Color)	Specifies a color with RGB value (255,255,240)
khaki (Khaki Preset Color)	Specifies a color with RGB value (240,230,140)
lavender (Lavender Preset Color)	Specifies a color with RGB value (230,230,250)
lavenderBlush (Lavender Blush Preset Color)	Specifies a color with RGB value (255,240,245)
lawnGreen (Lawn Green Preset Color)	Specifies a color with RGB value (124,252,0)
lemonChiffon (Lemon Chiffon Preset Color)	Specifies a color with RGB value (255,250,205)
lime (Lime Preset Color)	Specifies a color with RGB value (0,255,0)
limeGreen (Lime Green Preset Color)	Specifies a color with RGB value (50,205,50)
linen (Linen Preset Color)	Specifies a color with RGB value (250,240,230)
ltBlue (Light Blue Preset Color)	Specifies a color with RGB value (173,216,230)
ltCoral (Light Coral Preset Color)	Specifies a color with RGB value (240,128,128)
ltCyan (Light Cyan Preset Color)	Specifies a color with RGB value (224,255,255)
ltGoldenrodYellow (Light Goldenrod Yellow Preset Color)	Specifies a color with RGB value (250,250,120)
ltGray (Light Gray Preset Color)	Specifies a color with RGB value (211,211,211)
ltGreen (Light Green Preset Color)	Specifies a color with RGB value (144,238,144)
ltPink (Light Pink Preset Color)	Specifies a color with RGB value (255,182,193)
ltSalmon (Light Salmon Preset Color)	Specifies a color with RGB value (255,160,122)
ltSeaGreen (Light Sea Green Preset Color)	Specifies a color with RGB value (32,178,170)
ltSkyBlue (Light Sky Blue Preset Color)	Specifies a color with RGB value (135,206,250)
ltSlateGray (Light Slate Gray Preset Color)	Specifies a color with RGB value (119,136,153)
ltSteelBlue (Light Steel Blue Preset Color)	Specifies a color with RGB value (176,196,222)
ltYellow (Light Yellow Preset Color)	Specifies a color with RGB value (255,255,224)
magenta (Magenta Preset Color)	Specifies a color with RGB value (255,0,255)
maroon (Maroon Preset Color)	Specifies a color with RGB value (128,0,0)
medAquamarine (Medium Aquamarine Preset Color)	Specifies a color with RGB value (102,205,170)
medBlue (Medium Blue Preset Color)	Specifies a color with RGB value (0,0,205)
medOrchid (Medium Orchid Preset Color)	Specifies a color with RGB value (186,85,211)
medPurple (Medium Purple Preset Color)	Specifies a color with RGB value (147,112,219)
medSeaGreen (Medium Sea Green Preset Color)	Specifies a color with RGB value (60,179,113)

Enumeration Value	Description
medSlateBlue (Medium Slate Blue Preset Color)	Specifies a color with RGB value (123,104,238)
medSpringGreen (Medium Spring Green Preset Color)	Specifies a color with RGB value (0,250,154)
medTurquoise (Medium Turquoise Preset Color)	Specifies a color with RGB value (72,209,204)
medVioletRed (Medium Violet Red Preset Color)	Specifies a color with RGB value (199,21,133)
midnightBlue (Midnight Blue Preset Color)	Specifies a color with RGB value (25,25,112)
mintCream (Mint Cream Preset Color)	Specifies a color with RGB value (245,255,250)
mistyRose (Misty Rose Preset Color)	Specifies a color with RGB value (255,228,225)
moccasin (Moccasin Preset Color)	Specifies a color with RGB value (255,228,181)
navajoWhite (Navajo White Preset Color)	Specifies a color with RGB value (255,222,173)
navy (Navy Preset Color)	Specifies a color with RGB value (0,0,128)
oldLace (Old Lace Preset Color)	Specifies a color with RGB value (253,245,230)
olive (Olive Preset Color)	Specifies a color with RGB value (128,128,0)
oliveDrab (Olive Drab Preset Color)	Specifies a color with RGB value (107,142,35)
orange (Orange Preset Color)	Specifies a color with RGB value (255,165,0)
orangeRed (Orange Red Preset Color)	Specifies a color with RGB value (255,69,0)
orchid (Orchid Preset Color)	Specifies a color with RGB value (218,112,214)
paleGoldenrod (Pale Goldenrod Preset Color)	Specifies a color with RGB value (238,232,170)
paleGreen (Pale Green Preset Color)	Specifies a color with RGB value (152,251,152)
paleTurquoise (Pale Turquoise Preset Color)	Specifies a color with RGB value (175,238,238)
paleVioletRed (Pale Violet Red Preset Color)	Specifies a color with RGB value (219,112,147)
papayaWhip (Papaya Whip Preset Color)	Specifies a color with RGB value (255,239,213)
peachPuff (Peach Puff Preset Color)	Specifies a color with RGB value (255,218,185)
peru (Peru Preset Color)	Specifies a color with RGB value (205,133,63)
pink (Pink Preset Color)	Specifies a color with RGB value (255,192,203)
plum (Plum Preset Color)	Specifies a color with RGB value (221,160,221)
powderBlue (Powder Blue Preset Color)	Specifies a color with RGB value (176,224,230)
purple (Purple Preset Color)	Specifies a color with RGB value (128,0,128)
red (Red Preset Color)	Specifies a color with RGB value (255,0,0)
rosyBrown (Rosy Brown Preset Color)	Specifies a color with RGB value (188,143,143)
royalBlue (Royal Blue Preset Color)	Specifies a color with RGB value (65,105,225)
saddleBrown (Saddle Brown Preset Color)	Specifies a color with RGB value (139,69,19)
salmon (Salmon Preset Color)	Specifies a color with RGB value (250,128,114)
sandyBrown (Sandy Brown Preset Color)	Specifies a color with RGB value (244,164,96)
seaGreen (Sea Green Preset Color)	Specifies a color with RGB value (46,139,87)

Enumeration Value	Description
seaShell (Sea Shell Preset Color)	Specifies a color with RGB value (255,245,238)
sienna (Sienna Preset Color)	Specifies a color with RGB value (160,82,45)
silver (Silver Preset Color)	Specifies a color with RGB value (192,192,192)
skyBlue (Sky Blue Preset Color)	Specifies a color with RGB value (135,206,235)
slateBlue (Slate Blue Preset Color)	Specifies a color with RGB value (106,90,205)
slateGray (Slate Gray Preset Color)	Specifies a color with RGB value (112,128,144)
snow (Snow Preset Color)	Specifies a color with RGB value (255,250,250)
springGreen (Spring Green Preset Color)	Specifies a color with RGB value (0,255,127)
steelBlue (Steel Blue Preset Color)	Specifies a color with RGB value (70,130,180)
tan (Tan Preset Color)	Specifies a color with RGB value (210,180,140)
teal (Teal Preset Color)	Specifies a color with RGB value (0,128,128)
thistle (Thistle Preset Color)	Specifies a color with RGB value (216,191,216)
tomato (Tomato Preset Color)	Specifies a color with RGB value (255,99,71)
turquoise (Turquoise Preset Color)	Specifies a color with RGB value (64,224,208)
violet (Violet Preset Color)	Specifies a color with RGB value (238,130,238)
wheat (Wheat Preset Color)	Specifies a color with RGB value (245,222,179)
white (White Preset Color)	Specifies a color with RGB value (255,255,255)
whiteSmoke (White Smoke Preset Color)	Specifies a color with RGB value (245,245,245)
yellow (Yellow Preset Color)	Specifies a color with RGB value (255,255,0)
yellowGreen (Yellow Green Preset Color)	Specifies a color with RGB value (154,205,50)

Referenced By
prstClr@val (§5.1.2.2.22)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PresetColorVal">
   <restriction base="xsd:token">
      <enumeration value="aliceBlue"/>
      <enumeration value="antiqueWhite"/>
      <enumeration value="aqua"/>
      <enumeration value="aquamarine"/>
      <enumeration value="azure"/>
      <enumeration value="beige"/>
      <enumeration value="bisque"/>
      <enumeration value="black"/>
      <enumeration value="blanchedAlmond"/>
      <enumeration value="blue"/>
      <enumeration value="blueViolet"/>
      <enumeration value="brown"/>
      <enumeration value="burlyWood"/>
      <enumeration value="cadetBlue"/>
      <enumeration value="chartreuse"/>
      <enumeration value="chocolate"/>
      <enumeration value="coral"/>
      <enumeration value="cornflowerBlue"/>
      <enumeration value="cornsilk"/>
      <enumeration value="crimson"/>
      <enumeration value="cyan"/>
      <enumeration value="dkBlue"/>
      <enumeration value="dkCyan"/>
      <enumeration value="dkGoldenrod"/>
      <enumeration value="dkGray"/>
      <enumeration value="dkGreen"/>
      <enumeration value="dkKhaki"/>
      <enumeration value="dkMagenta"/>
      <enumeration value="dkOliveGreen"/>
      <enumeration value="dkOrange"/>
      <enumeration value="dkOrchid"/>
      <enumeration value="dkRed"/>
      <enumeration value="dkSalmon"/>
      <enumeration value="dkSeaGreen"/>
      <enumeration value="dkSlateBlue"/>
      <enumeration value="dkSlateGray"/>
      <enumeration value="dkTurquoise"/>
      <enumeration value="dkViolet"/>
      <enumeration value="deepPink"/>
      <enumeration value="deepSkyBlue"/>
      <enumeration value="dimGray"/>
      <enumeration value="dodgerBlue"/>
      <enumeration value="firebrick"/>
      <enumeration value="floralWhite"/>
      <enumeration value="forestGreen"/>
      <enumeration value="fuchsia"/>
      <enumeration value="gainsboro"/>
      <enumeration value="ghostWhite"/>
      <enumeration value="gold"/>
      <enumeration value="goldenrod"/>
```

```
        <enumeration value="gray"/>
        <enumeration value="green"/>
        <enumeration value="greenYellow"/>
        <enumeration value="honeydew"/>
        <enumeration value="hotPink"/>
        <enumeration value="indianRed"/>
        <enumeration value="indigo"/>
        <enumeration value="ivory"/>
        <enumeration value="khaki"/>
        <enumeration value="lavender"/>
        <enumeration value="lavenderBlush"/>
        <enumeration value="lawnGreen"/>
        <enumeration value="lemonChiffon"/>
        <enumeration value="ltBlue"/>
        <enumeration value="ltCoral"/>
        <enumeration value="ltCyan"/>
        <enumeration value="ltGoldenrodYellow"/>
        <enumeration value="ltGray"/>
        <enumeration value="ltGreen"/>
        <enumeration value="ltPink"/>
        <enumeration value="ltSalmon"/>
        <enumeration value="ltSeaGreen"/>
        <enumeration value="ltSkyBlue"/>
        <enumeration value="ltSlateGray"/>
        <enumeration value="ltSteelBlue"/>
        <enumeration value="ltYellow"/>
        <enumeration value="lime"/>
        <enumeration value="limeGreen"/>
        <enumeration value="linen"/>
        <enumeration value="magenta"/>
        <enumeration value="maroon"/>
        <enumeration value="medAquamarine"/>
        <enumeration value="medBlue"/>
        <enumeration value="medOrchid"/>
        <enumeration value="medPurple"/>
        <enumeration value="medSeaGreen"/>
        <enumeration value="medSlateBlue"/>
        <enumeration value="medSpringGreen"/>
        <enumeration value="medTurquoise"/>
        <enumeration value="medVioletRed"/>
        <enumeration value="midnightBlue"/>
        <enumeration value="mintCream"/>
        <enumeration value="mistyRose"/>
        <enumeration value="moccasin"/>
        <enumeration value="navajoWhite"/>
        <enumeration value="navy"/>
        <enumeration value="oldLace"/>
        <enumeration value="olive"/>
        <enumeration value="oliveDrab"/>
        <enumeration value="orange"/>
        <enumeration value="orangeRed"/>
        <enumeration value="orchid"/>
        <enumeration value="paleGoldenrod"/>
```

```
        <enumeration value="paleGreen"/>
        <enumeration value="paleTurquoise"/>
        <enumeration value="paleVioletRed"/>
        <enumeration value="papayaWhip"/>
        <enumeration value="peachPuff"/>
        <enumeration value="peru"/>
        <enumeration value="pink"/>
        <enumeration value="plum"/>
        <enumeration value="powderBlue"/>
        <enumeration value="purple"/>
        <enumeration value="red"/>
        <enumeration value="rosyBrown"/>
        <enumeration value="royalBlue"/>
        <enumeration value="saddleBrown"/>
        <enumeration value="salmon"/>
        <enumeration value="sandyBrown"/>
        <enumeration value="seaGreen"/>
        <enumeration value="seaShell"/>
        <enumeration value="sienna"/>
        <enumeration value="silver"/>
        <enumeration value="skyBlue"/>
        <enumeration value="slateBlue"/>
        <enumeration value="slateGray"/>
        <enumeration value="snow"/>
        <enumeration value="springGreen"/>
        <enumeration value="steelBlue"/>
        <enumeration value="tan"/>
        <enumeration value="teal"/>
        <enumeration value="thistle"/>
        <enumeration value="tomato"/>
        <enumeration value="turquoise"/>
        <enumeration value="violet"/>
        <enumeration value="wheat"/>
        <enumeration value="white"/>
        <enumeration value="whiteSmoke"/>
        <enumeration value="yellow"/>
        <enumeration value="yellowGreen"/>
    </restriction>
</simpleType>
```

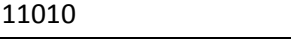
5.1.12.49 ST_PresetLineDashVal (Preset Line Dash Value)

This simple type represents preset line dash values. The description for each style shows an illustration of the line style. Each style also contains a precise binary representation of the repeating dash style. Each 1 corresponds to a line segment of the same length as the line width, and each 0 corresponds to a space of the same length as the line width.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description

Enumeration Value	Description
dash (Dash)	 1111000
dashDot (Dash Dot)	 11110001000
dot (Dot)	 1000
lgDash (Large Dash)	 11111111000
lgDashDot (Large Dash Dot)	 111111110001000
lgDashDotDot (Large Dash Dot Dot)	 1111111100010001000
solid (Solid)	 1
sysDash (System Dash)	 1110
sysDashDot (System Dash Dot)	 111010
sysDashDotDot (System Dash Dot Dot)	 11101010
sysDot (System Dot)	 10

Referenced By
prstDash@val (§5.1.10.48)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PresetLineDashVal">
   <restriction base="xsd:token">
      <enumeration value="solid"/>
      <enumeration value="dot"/>
      <enumeration value="dash"/>
      <enumeration value="lgDash"/>
      <enumeration value="dashDot"/>
      <enumeration value="lgDashDot"/>
      <enumeration value="lgDashDotDot"/>
      <enumeration value="sysDash"/>
      <enumeration value="sysDot"/>
      <enumeration value="sysDashDot"/>
      <enumeration value="sysDashDotDot"/>
   </restriction>
</simpleType>
```

5.1.12.50 ST_PresetMaterialType (Preset Material Type)

Describes surface appearance of a shape. The material type combines with lighting characteristics to create the final look and feel of a shape. The following properties were used to define the shape used in the image examples below:

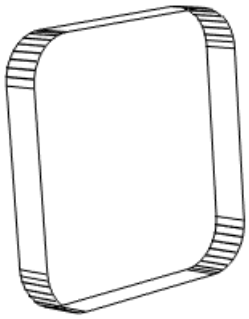
- Rounded rectangle shape
- Circle bevel type
- Three Point light rig type
- Camera type defined by the `perspectiveContrastingRightFacing` preset
- Bevel width and height each equal to `190500`

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
clear (Clear)	[*Example*: Consider the following example of the `clear` material type:  *end example*]

Enumeration Value	Description
dkEdge (Dark Edge)	[*Example*: Consider the following example of the `dkEdge` material type:  *end example*]
flat (Flat)	[*Example*: Consider the following example of the `flat` material type:  *end example*]
legacyMatte (Legacy Matte)	[*Example*: Consider the following example of the `legacyMatte` material type: 

Enumeration Value	Description
	end example]
legacyMetal (Legacy Metal)	[*Example*: Consider the following example of the `legacyMetal` material type:  *end example*]
legacyPlastic (Legacy Plastic)	[*Example*: Consider the following example of the `legacyPlastic` material type:  *end example*]
legacyWireframe (Legacy Wireframe)	[*Example*: Consider the following example of the `legacyWireframe` material type: 

Enumeration Value	Description
	end example]
matte (Matte)	[*Example*: Consider the following example of the `matte` material type:  *end example*]
metal (Metal)	[*Example*: Consider the following example of the `warmMatte` material type:  *end example*]
plastic (Plastic)	[*Example*: Consider the following example of the `warmMatte` material type:

Enumeration Value	Description
	 *end example*]
powder (Powder)	[*Example*: Consider the following example of the `warmMatte` material type:  *end example*]
softEdge (Soft Edge)	[*Example*: Consider the following example of the `warmMatte` material type:  *end example*]
softmetal (Soft Metal)	[*Example*: Consider the following example of the `softmetal` material type:

Enumeration Value	Description
	 *end example*]
translucentPowder (Translucent Powder)	[*Example*: Consider the following example of the `warmMatte` material type:  *end example*]
warmMatte (Warm Matte)	[*Example*: Consider the following example of the `warmMatte` material type:  *end example*]

Referenced By
cell3D@prstMaterial (§5.1.6.1); sp3d@prstMaterial (§5.1.7.12); sp3d@prstMaterial (§5.9.5.6)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PresetMaterialType">
   <restriction base="xsd:token">
      <enumeration value="legacyMatte"/>
      <enumeration value="legacyPlastic"/>
      <enumeration value="legacyMetal"/>
      <enumeration value="legacyWireframe"/>
      <enumeration value="matte"/>
      <enumeration value="plastic"/>
      <enumeration value="metal"/>
      <enumeration value="warmMatte"/>
      <enumeration value="translucentPowder"/>
      <enumeration value="powder"/>
      <enumeration value="dkEdge"/>
      <enumeration value="softEdge"/>
      <enumeration value="clear"/>
      <enumeration value="flat"/>
      <enumeration value="softmetal"/>
   </restriction>
</simpleType>
```

5.1.12.51 ST_PresetPatternVal (Preset Pattern Value)

This simple type indicates a preset type of pattern fill. The description of each value contains an illustration of the fill type.

[*Note*: These presets correspond to members of the HatchStyle enumeration in the Microsoft .NET Framework. A reference for this type may be found at http://msdn2.microsoft.com/en-us/library/system.drawing.drawing2d.hatchstyle.aspx. *end note*]

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
cross (Cross)	
dashDnDiag (Dashed Downward Diagonal)	
dashHorz (Dashed Horizontal)	
dashUpDiag (Dashed Upward DIagonal)	
dashVert (Dashed Vertical)	

Enumeration Value	Description
diagBrick (Diagonal Brick)	
diagCross (Diagonal Cross)	
divot (Divot)	
dkDnDiag (Dark Downward Diagonal)	
dkHorz (Dark Horizontal)	
dkUpDiag (Dark Upward Diagonal)	
dkVert (Dark Vertical)	
dnDiag (Downward Diagonal)	
dotDmnd (Dotted Diamond)	
dotGrid (Dotted Grid)	
horz (Horizontal)	
horzBrick (Horizontal Brick)	
lgCheck (Large Checker Board)	
lgConfetti (Large Confetti)	
lgGrid (Large Grid)	
ltDnDiag (Light Downward Diagonal)	
ltHorz (Light Horizontal)	
ltUpDiag (Light Upward Diagonal)	
ltVert (Light Vertical)	
narHorz (Narrow Horizontal)	
narVert (Narrow Vertical)	
openDmnd (Open Diamond)	

Enumeration Value	Description
pct10 (10%)	
pct20 (20%)	
pct25 (25%)	
pct30 (30%)	
pct40 (40%)	
pct5 (5%)	
pct50 (50%)	
pct60 (60%)	
pct70 (70%)	
pct75 (75%)	
pct80 (80%)	
pct90 (90%)	
plaid (Plaid)	
shingle (Shingle)	
smCheck (Small Checker Board)	
smConfetti (Small Confetti)	
smGrid (Small Grid)	
solidDmnd (Solid Diamond)	
sphere (Sphere)	
trellis (Trellis)	
upDiag (Upward Diagonal)	
vert (Vertical)	

Enumeration Value	Description
wave (Wave)	
wdDnDiag (Wide Downward Diagonal)	
wdUpDiag (Wide Upward Diagonal)	
weave (Weave)	
zigZag (Zig Zag)	

Referenced By
pattFill@prst (§5.1.10.47)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PresetPatternVal">
   <restriction base="xsd:token">
      <enumeration value="pct5"/>
      <enumeration value="pct10"/>
      <enumeration value="pct20"/>
      <enumeration value="pct25"/>
      <enumeration value="pct30"/>
      <enumeration value="pct40"/>
      <enumeration value="pct50"/>
      <enumeration value="pct60"/>
      <enumeration value="pct70"/>
      <enumeration value="pct75"/>
      <enumeration value="pct80"/>
      <enumeration value="pct90"/>
      <enumeration value="horz"/>
      <enumeration value="vert"/>
      <enumeration value="ltHorz"/>
      <enumeration value="ltVert"/>
      <enumeration value="dkHorz"/>
      <enumeration value="dkVert"/>
      <enumeration value="narHorz"/>
      <enumeration value="narVert"/>
      <enumeration value="dashHorz"/>
      <enumeration value="dashVert"/>
      <enumeration value="cross"/>
      <enumeration value="dnDiag"/>
      <enumeration value="upDiag"/>
      <enumeration value="ltDnDiag"/>
      <enumeration value="ltUpDiag"/>
      <enumeration value="dkDnDiag"/>
      <enumeration value="dkUpDiag"/>
      <enumeration value="wdDnDiag"/>
      <enumeration value="wdUpDiag"/>
      <enumeration value="dashDnDiag"/>
      <enumeration value="dashUpDiag"/>
      <enumeration value="diagCross"/>
      <enumeration value="smCheck"/>
      <enumeration value="lgCheck"/>
      <enumeration value="smGrid"/>
      <enumeration value="lgGrid"/>
      <enumeration value="dotGrid"/>
      <enumeration value="smConfetti"/>
      <enumeration value="lgConfetti"/>
      <enumeration value="horzBrick"/>
      <enumeration value="diagBrick"/>
      <enumeration value="solidDmnd"/>
      <enumeration value="openDmnd"/>
      <enumeration value="dotDmnd"/>
      <enumeration value="plaid"/>
      <enumeration value="sphere"/>
      <enumeration value="weave"/>
      <enumeration value="divot"/>
```

```
        <enumeration value="shingle"/>
        <enumeration value="wave"/>
        <enumeration value="trellis"/>
        <enumeration value="zigZag"/>
    </restriction>
</simpleType>
```

5.1.12.52 ST_PresetShadowVal (Preset Shadow Type)

This simple type indicates one of 20 preset shadow types. Each enumeration value description illustrates the type of shadow represented by the value. Each description contains the parameters to the outer shadow effect represented by the preset, in addition to those attributes common to all prstShdw effects.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
shdw1 (Top Left Drop Shadow)	 No additional attributes specified.
shdw10 (Top Left Large Drop Shadow)	 align = "br" sx = 125% sy = 125%
shdw11 (Back Left Long Perspective Shadow)	 align = "b" kx = 40.89° sy = 50%
shdw12 (Back Right Long Perspective Shadow)	 align = "b" kx = -40.89° sy = 50%
shdw13 (Top Left Double Drop Shadow)	 Equivalent to two outer shadow effects. Shadow 1: No additional attributes specified. Shadow 2: color = min(1, shadow 1's color (0 <= r, g, b <= 1) + 102/255), per r, g, b component dist = 2 * shadow 1's distance
shdw14 (Bottom Right Small Drop Shadow)	

Enumeration Value	Description
	No additional attributes specified.
shdw15 (Front Left Long Perspective Shadow)	 align = "b" kx = 40.89° sy = -50%
shdw16 (Front Right LongPerspective Shadow)	 align = "b" kx = -40.89° sy = -50%
shdw17 (3D Outer Box Shadow)	 Equivalent to two outer shadow effects. Shadow 1: No additional attributes specified. Shadow 2: color = min(1, shadow 1's color (0 <= r, g, b <= 1) + 102/255), per r, g, b component dir = shadow 1's direction + 180°
shdw18 (3D Inner Box Shadow)	 Equivalent to two outer shadow effects. Shadow 1: No additional attributes specified. Shadow 2: color = min(1, shadow 1's color (0 <= r, g, b <= 1) + 102/255), per r, g, b component dir = shadow 1's direction + 180°
shdw19 (Back Center Perspective Shadow)	 align = "b" sy = 50°
shdw2 (Top Right Drop Shadow)	 No additional attributes specified.
shdw20 (Front Bottom Shadow)	 align = "b"

Enumeration Value	Description
	sy = -100°
shdw3 (Back Left Perspective Shadow)	 align = "b" ky = 40.89° sy = 50%
shdw4 (Back Right Perspective Shadow)	 align = "b" kx = -40.89° sy = 50%
shdw5 (Bottom Left Drop Shadow)	 No additional attributes specified.
shdw6 (Bottom Right Drop Shadow)	 No additional attributes specified.
shdw7 (Front Left Perspective Shadow)	 align = "b" kx = 40.89° sy = -50%
shdw8 (Front Right Perspective Shadow)	 align = "b" kx = -40.89° sy = -50%
shdw9 (Top Left Small Drop Shadow)	 align = "tl" sx = 75% sy = 75%

Referenced By
prstShdw@prst (§5.1.10.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PresetShadowVal">
   <restriction base="xsd:token">
      <enumeration value="shdw1"/>
      <enumeration value="shdw2"/>
      <enumeration value="shdw3"/>
      <enumeration value="shdw4"/>
      <enumeration value="shdw5"/>
      <enumeration value="shdw6"/>
      <enumeration value="shdw7"/>
      <enumeration value="shdw8"/>
      <enumeration value="shdw9"/>
      <enumeration value="shdw10"/>
      <enumeration value="shdw11"/>
      <enumeration value="shdw12"/>
      <enumeration value="shdw13"/>
      <enumeration value="shdw14"/>
      <enumeration value="shdw15"/>
      <enumeration value="shdw16"/>
      <enumeration value="shdw17"/>
      <enumeration value="shdw18"/>
      <enumeration value="shdw19"/>
      <enumeration value="shdw20"/>
   </restriction>
</simpleType>
```

5.1.12.53 ST_RectAlignment (Rectangle Alignments)

This simple type describes how to position two rectangles relative to each other.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
b (Rectangle Alignment Enum (Bottom))	Bottom
bl (Rectangle Alignment Enum (Bottom Left))	Bottom Left
br (Rectangle Alignment Enum (Bottom Right))	Bottom Right
ctr (Rectangle Alignment Enum (Center))	Center
l (Rectangle Alignment Enum (Left))	Left
r (Rectangle Alignment Enum (Right))	Right
t (Rectangle Alignment Enum (Top))	Top
tl (Rectangle Alignment Enum (Top Left))	Top Left
tr (Rectangle Alignment Enum (Top Right))	Top Right

Referenced By

Referenced By
outerShdw@algn (§5.1.10.45); reflection@algn (§5.1.10.50); tile@algn (§5.1.10.58)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_RectAlignment">
   <restriction base="xsd:token">
      <enumeration value="tl"/>
      <enumeration value="t"/>
      <enumeration value="tr"/>
      <enumeration value="l"/>
      <enumeration value="ctr"/>
      <enumeration value="r"/>
      <enumeration value="bl"/>
      <enumeration value="b"/>
      <enumeration value="br"/>
   </restriction>
</simpleType>
```

5.1.12.54 ST_SchemeColorVal (Scheme Color)

This simple type represents a scheme color value.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
accent1 (Accent Color 1)	Extra scheme color 1
accent2 (Accent Color 2)	Extra scheme color 2
accent3 (Accent Color 3)	Extra scheme color 3
accent4 (Accent Color 4)	Extra scheme color 4
accent5 (Accent Color 5)	Extra scheme color 5
accent6 (Accent Color 6)	Extra scheme color 6
bg1 (Background Color 1)	Semantic background color
bg2 (Background Color 2)	Semantic additional background color
dk1 (Dark Color 1)	Main dark color 1
dk2 (Dark Color 2)	Main dark color 2
folHlink (Followed Hyperlink Color)	Followed Hyperlink Color
hlink (Hyperlink Color)	Regular Hyperlink Color
lt1 (Light Color 1)	Main Light Color 1
lt2 (Light Color 2)	Main Light Color 2
phClr (Style Color)	A color used in theme definitions which means to use the color of the style.

Enumeration Value	Description
tx1 (Text Color 1)	Semantic text color
tx2 (Text Color 2)	Semantic additional text color

Referenced By
schemeClr@val (§5.1.2.2.29)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SchemeColorVal">
   <restriction base="xsd:token">
      <enumeration value="bg1"/>
      <enumeration value="tx1"/>
      <enumeration value="bg2"/>
      <enumeration value="tx2"/>
      <enumeration value="accent1"/>
      <enumeration value="accent2"/>
      <enumeration value="accent3"/>
      <enumeration value="accent4"/>
      <enumeration value="accent5"/>
      <enumeration value="accent6"/>
      <enumeration value="hlink"/>
      <enumeration value="folHlink"/>
      <enumeration value="phClr"/>
      <enumeration value="dk1"/>
      <enumeration value="lt1"/>
      <enumeration value="dk2"/>
      <enumeration value="lt2"/>
   </restriction>
</simpleType>
```

5.1.12.55 ST_ShapeID (Shape ID)

Specifies the shape ID for legacy shape identification purposes.

This simple type's contents are a restriction of the XML Schema token datatype.

Referenced By
bldDgm@spid (§4.6.12); bldGraphic@spid (§4.6.13); bldOleChart@spid (§4.6.15); bldP@spid (§4.6.16); control@spid (§4.4.2.1); inkTgt@spid (§4.6.47); legacyDrawing@spid (§5.3.2.1); oleObj@spid (§4.4.2.4); spTgt@spid (§4.6.72); subSp@spid (§4.6.77)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ShapeID">
   <restriction base="xsd:token"/>
</simpleType>
```

5.1.12.56 ST_ShapeType (Preset Shape Types)

This simple type specifies the preset shape geometry that is to be used for a shape. An enumeration of this type is used so that a custom geometry does not have to be specified but instead can be constructed automatically by the generating application. For each enumeration listed there is also the corresponding DrawingML code that would be used to construct this shape were it a custom geometry. Within the construction code for each of these preset shapes there are predefined guides that the generating application must maintain for calculation purposes at all times. The necessary guides should have the following values.

3/4 of a Circle ('3cd4') - Constant value of "16200000.0"

The units here are in 60,000ths of a degree. This is equivalent to 270 degrees.

3/8 of a Circle ('3cd8') - Constant value of "8100000.0"

The units here are in 60,000ths of a degree. This is equivalent to 135 degrees.

5/8 of a Circle ('5cd8') - Constant value of "13500000.0"

The units here are in 60,000ths of a degree. This is equivalent to 225 degrees.

7/8 of a Circle ('7cd8') - Constant value of "18900000.0"

The units here are in 60,000ths of a degree. This is equivalent to 315 degrees.

Shape Bottom Edge ('b') - Constant value of "h"

This is the bottom edge of the shape and since the top edge of the shape is considered the 0 point, the bottom edge is thus the shape height.

1/2 of a Circle ('cd2') - Constant value of "10800000.0"

The units here are in 60,000ths of a degree. This is equivalent to 180 degrees.

1/4 of a Circle ('cd4') - Constant value of "5400000.0"

The units here are in 60,000ths of a degree. This is equivalent to 90 degrees.

1/8 of a Circle ('cd8') - Constant value of "2700000.0"

The units here are in 60,000ths of a degree. This is equivalent to 45 degrees.

Shape Height ('h')

This is the variable height of the shape defined in the shape properties. This value is received from the shape transform listed within the <spPr> element.

Horizontal Center ('hc') - Calculated value of "*/ w 1.0 2.0"

This is the horizontal center of the shape which is just the width divided by 2.

1/2 of Shape Height ('hd2') - Calculated value of "*/ h 1.0 2.0"

This is 1/2 the shape height.

1/4 of Shape Height ('hd4') - Calculated value of "*/ h 1.0 4.0"

This is 1/4 the shape height.

1/5 of Shape Height ('hd5') - Calculated value of "*/ h 1.0 5.0"

This is 1/5 the shape height.

1/6 of Shape Height ('hd6') - Calculated value of "*/ h 1.0 6.0"

This is 1/6 the shape height.

1/8 of Shape Height ('hd8') - Calculated value of "*/ h 1.0 8.0"

This is 1/8 the shape height.

Shape Left Edge ('l') - Constant value of "0"

This is the left edge of the shape and the left edge of the shape is considered the horizontal 0 point.

Longest Side of Shape ('ls') - Calculated value of "max w h"

This is the longest side of the shape. This value is either the width or the height depending on which is greater.

Shape Right Edge ('r') - Constant value of "w"

This is the right edge of the shape and since the left edge of the shape is considered the 0 point, the right edge is thus the shape width.

Shortest Side of Shape ('ss') - Calculated value of "min w h"

This is the shortest side of the shape. This value is either the width or the height depending on which is smaller.

1/2 Shortest Side of Shape ('ssd2') - Calculated value of "*/ ss 1.0 2.0"

This is 1/2 the shortest side of the shape.

1/4 Shortest Side of Shape ('ssd4') - Calculated value of "*/ ss 1.0 4.0"

This is 1/4 the shortest side of the shape.

1/6 Shortest Side of Shape ('ssd6') - Calculated value of "*/ ss 1.0 6.0"

This is 1/6 the shortest side of the shape.

1/8 Shortest Side of Shape ('ssd8') - Calculated value of "*/ ss 1.0 8.0"

This is 1/8 the shortest side of the shape.

Shape Top Edge ('t') - Constant value of "0"

This is the top edge of the shape and the top edge of the shape is considered the vertical 0 point.

Vertical Center of Shape ('vc') - Calculated value of "*/ h 1.0 2.0"

This is the vertical center of the shape which is just the height divided by 2.

Shape Width ('w')

This is the variable width of the shape defined in the shape properties. This value is received from the shape transform listed within the <spPr> element.

1/2 of Shape Width ('wd2') - Calculated value of "*/ w 1.0 2.0"

This is 1/2 the shape width.

1/4 of Shape Width ('wd4') - Calculated value of "*/ w 1.0 4.0"

This is 1/4 the shape width.

1/5 of Shape Width ('wd5') - Calculated value of "*/ w 1.0 5.0"

This is 1/5 the shape width.

1/6 of Shape Width ('wd6') - Calculated value of "*/ w 1.0 6.0"

This is 1/6 the shape width.

1/8 of Shape Width ('wd8') - Calculated value of "*/ w 1.0 8.0"

This is 1/8 the shape width.

1/10 of Shape Width ('wd10') - Calculated value of "*/ w 1.0 10.0"

This is 1/10 the shape width.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
accentBorderCallout1 (Callout 1 with Border and Accent Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the accentBorderCallout1 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
accentBorderCallout2 (Callout 2 with Border and Accent Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the accentBorderCallout2 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
accentBorderCallout3 (Callout 3 with Border and Accent Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the accentBorderCallout3 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are

Enumeration Value	Description
	described in further detail above. *end note*] 
accentCallout1 (Callout 1 Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the accentCallout1 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
accentCallout2 (Callout 2 Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the accentCallout2 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
accentCallout3 (Callout 3 Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the accentCallout3 element in the preset shape geometries electronic addenda of Annex D. The

Enumeration Value	Description
	constants used in that markup are guides that are described in further detail above. *end note*] 
actionButtonBackPrevious (Back or Previous Button Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the actionButtonBackPrevious element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
actionButtonBeginning (Beginning Button Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the actionButtonBeginning element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
actionButtonBlank (Blank Button Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below.

Enumeration Value	Description
	[*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the actionButtonBlank element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
actionButtonDocument (Document Button Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the actionButtonDocument element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
actionButtonEnd (End Button Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the actionButtonEnd element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]

Enumeration Value	Description
	
actionButtonForwardNext (Forward or Next Button Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the actionButtonForwardNext element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
actionButtonHelp (Help Button Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the actionButtonHelp element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
actionButtonHome (Home Button Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used

Enumeration Value	Description
	to generate this preset shape definition is contained in the actionButtonHome element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
actionButtonInformation (Information Button Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the actionButtonInformation element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
actionButtonMovie (Movie Button Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the actionButtonMovie element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]

Enumeration Value	Description
	
actionButtonReturn (Return Button Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the actionButtonReturn element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
actionButtonSound (Sound Button Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the actionButtonSound element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
arc (Curved Arc Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used

Enumeration Value	Description
	to generate this preset shape definition is contained in the arc element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
bentArrow (Bent Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the bentArrow element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
bentConnector2 (Bent Connector 2 Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the bentConnector2 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
bentConnector3 (Bent Connector 3 Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used

Enumeration Value	Description
	to generate this preset shape definition is contained in the bentConnector3 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
bentConnector4 (Bent Connector 4 Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the bentConnector4 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
bentConnector5 (Bent Connector 5 Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the bentConnector5 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
bentUpArrow (Bent Up Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below.

Enumeration Value	Description
	[*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the bentUpArrow element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
bevel (Bevel Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the bevel element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
blockArc (Block Arc Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the blockArc element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
borderCallout1 (Callout 1 with Border Shape)	Specifies a preset shape geometry. This geometry shall

Enumeration Value	Description
	be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the borderCallout1 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
borderCallout2 (Callout 2 with Border Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the borderCallout2 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
borderCallout3 (Callout 3 with Border Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the borderCallout3 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 

Enumeration Value	Description
bracePair (Brace Pair Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the bracePair element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
bracketPair (Bracket Pair Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the bracketPair element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
callout1 (Callout 1 Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the callout1 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]

Enumeration Value	Description
	
callout2 (Callout 2 Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the callout2 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
callout3 (Callout 3 Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the callout3 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
can (Can Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the can element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]

Enumeration Value	Description
	
chartPlus (Chart Plus Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the chartPlus element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
chartStar (Chart Star Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the chartStar element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
chartX (Chart X Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the chartX element in the preset shape geometries

Enumeration Value	Description
	electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
chevron (Chevron Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the chevron element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
chord (Chord Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the chord element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
circularArrow (Circular Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in

Enumeration Value	Description
	the circularArrow element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
cloud (Cloud Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the cloud element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
cloudCallout (Callout Cloud Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the cloudCallout element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
corner (Corner Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in

Enumeration Value	Description
	the corner element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
cornerTabs (Corner Tabs Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the cornerTabs element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
cube (Cube Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the cube element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
curvedConnector2 (Curved Connector 2 Shape)	Specifies a preset shape geometry. This geometry shall

Enumeration Value	Description
	be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the curvedConnector2 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
curvedConnector3 (Curved Connector 3 Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the curvedConnector3 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
curvedConnector4 (Curved Connector 4 Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the curvedConnector4 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 

Enumeration Value	Description
curvedConnector5 (Curved Connector 5 Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the curvedConnector5 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
curvedDownArrow (Curved Down Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the curvedDownArrow element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
curvedLeftArrow (Curved Left Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the curvedLeftArrow element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]

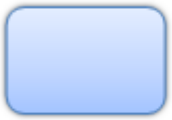
Enumeration Value	Description
	
curvedRightArrow (Curved Right Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the curvedRightArrow element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
curvedUpArrow (Curved Up Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the curvedUpArrow element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
decagon (Decagon Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used

Enumeration Value	Description
	to generate this preset shape definition is contained in the decagon element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
diagStripe (Diagonal Stripe Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the diagStripe element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
diamond (Diamond Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the diamond element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 

Enumeration Value	Description
dodecagon (Dodecagon Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the dodecagon element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
donut (Donut Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the donut element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
doubleWave (Double Wave Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the doubleWave element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]

Enumeration Value	Description
	
downArrow (Down Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the downArrow element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
downArrowCallout (Callout Down Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the downArrowCallout element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
ellipse (Ellipse Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the ellipse element in the preset shape geometries

Enumeration Value	Description
	electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
ellipseRibbon (Ellipse Ribbon Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the ellipseRibbon element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
ellipseRibbon2 (Ellipse Ribbon 2 Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the ellipseRibbon2 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartAlternateProcess (Alternate Process Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below.

Enumeration Value	Description
	[*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the flowChartAlternateProcess element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartCollate (Collate Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the flowChartCollate element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartConnector (Connector Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the flowChartConnector element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartDecision (Decision Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below.

Enumeration Value	Description
	[*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the flowChartDecision element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartDelay (Delay Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the flowChartDelay element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartDisplay (Display Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the flowChartDisplay element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartDocument (Document Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used

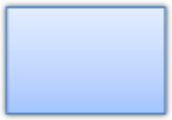
Enumeration Value	Description
	to generate this preset shape definition is contained in the flowChartDocument element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartExtract (Extract Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the flowChartExtract element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartInputOutput (Input Output Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the flowChartInputOutput element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartInternalStorage (Internal Storage Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used

Enumeration Value	Description
	to generate this preset shape definition is contained in the flowChartInternalStorage element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartMagneticDisk (Magnetic Disk Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the flowChartMegneticDisk element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartMagneticDrum (Magnetic Drum Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the flowChartMagneticDrum element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartMagneticTape (Magnetic Tape Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used

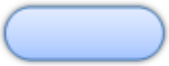
Enumeration Value	Description
	to generate this preset shape definition is contained in the flowChartMagneticTape element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartManualInput (Manual Input Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the flowChartManualInput element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartManualOperation (Manual Operation Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the flowChartManualOperation element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartMerge (Merge Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in

Enumeration Value	Description
	the flowChartMerge element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartMultidocument (Multi-Document Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the flowChartMultidocument element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartOfflineStorage (Offline Storage Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the flowChartOfflineStorage element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartOffpageConnector (Off-Page Connector Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used

Enumeration Value	Description
	to generate this preset shape definition is contained in the flowChartOffpageConnector element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartOnlineStorage (Online Storage Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the flowChartOnlineStorage element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartOr (Or Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the flowChartOr element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartPredefinedProcess (Predefined Process Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used

Enumeration Value	Description
	to generate this preset shape definition is contained in the flowChartPredefinedProcess element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartPreparation (Preparation Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the flowChartPreparation element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartProcess (Process Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the flowChartProcess element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartPunchedCard (Punched Card Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used

Enumeration Value	Description
	to generate this preset shape definition is contained in the flowChartPunchedCard element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartPunchedTape (Punched Tape Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the flowChartPunchedTape element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartSort (Sort Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the flowChartSort element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartSummingJunction (Summing Junction Flow	Specifies a preset shape geometry. This geometry shall

Enumeration Value	Description
Shape)	be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the flowChartSummingJunction element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
flowChartTerminator (Terminator Flow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the flowChartTerminator element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
foldedCorner (Folded Corner Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the foldedCorner element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
frame (Frame Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below.

Enumeration Value	Description
	[*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the frame element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
funnel (Funnel Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the funnel element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
gear6 (Gear 6 Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the gear6 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
gear9 (Gear 9 Shape)	Specifies a preset shape geometry. This geometry shall

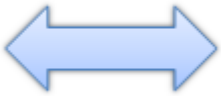
Enumeration Value	Description
	be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the gear9 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
halfFrame (Half Frame Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the halfFrame element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
heart (Heart Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the heart element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]

Enumeration Value	Description
	
heptagon (Heptagon Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the heptagon element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
hexagon (Hexagon Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the hexagon element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
homePlate (Home Plate Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in

Enumeration Value	Description
	the homePlate element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
horizontalScroll (Horizontal Scroll Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the horizontalScroll element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
irregularSeal1 (Irregular Seal 1 Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the irregularSeal1 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
irregularSeal2 (Irregular Seal 2 Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below.

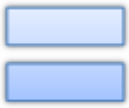
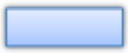
Enumeration Value	Description
	[*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the irregularSeal2 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
leftArrow (Left Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the leftArrow element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
leftArrowCallout (Callout Left Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the leftArrowCallout element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
leftBrace (Left Brace Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below.

Enumeration Value	Description
	[*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the leftBrace element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
leftBracket (Left Bracket Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the leftBracket element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
leftCircularArrow (Left Circular Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the leftCircularArrow element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
leftRightArrow (Left Right Arrow Shape)	Specifies a preset shape geometry. This geometry shall

Enumeration Value	Description
	be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the leftRightArrow element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
leftRightArrowCallout (Callout Left Right Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the leftRightArrowCallout element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
leftRightCircularArrow (Left Right Circular Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the leftRightCircularArrow element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
leftRightRibbon (Left Right Ribbon Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used

Enumeration Value	Description
	to generate this preset shape definition is contained in the leftRightRibbon element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
leftRightUpArrow (Left Right Up Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the leftRightUpArrow element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
leftUpArrow (Left Up Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the leftUpArrow element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
lightningBolt (Lightning Bolt Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below.

Enumeration Value	Description
	[*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the lightningBolt element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
line (Line Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the line element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
lineInv (Line Inverse Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the lineInv element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 

Enumeration Value	Description
mathDivide (Divide Math Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the mathDivide element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
mathEqual (Equal Math Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the mathEqual element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
mathMinus (Minus Math Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the mathMinus element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
mathMultiply (Multiply Math Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below.

Enumeration Value	Description
	[*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the mathMultiply element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
mathNotEqual (Not Equal Math Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the mathNotEqual element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
mathPlus (Plus Math Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the mathPlus element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
moon (Moon Shape)	Specifies a preset shape geometry. This geometry shall

Enumeration Value	Description
	be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the moon element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
nonIsoscelesTrapezoid (Non-Isosceles Trapezoid Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the nonIsocelesTrapezoid element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
noSmoking (No Smoking Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the noSmoking element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]

Enumeration Value	Description
	
notchedRightArrow (Notched Right Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the notchedRightArrow element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
octagon (Octagon Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the octagon element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
parallelogram (Parallelogram Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the parallelogram element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are

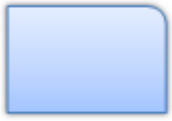
Enumeration Value	Description
	described in further detail above. *end note*] 
pentagon (Pentagon Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the pentagon element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
pie (Pie Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the pie element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
pieWedge (Pie Wedge Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below.

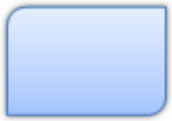
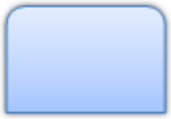
Enumeration Value	Description
	[*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the pieWedge element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
plaque (Plaque Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the plaque element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
plaqueTabs (Plaque Tabs Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the plaqueTabs element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
plus (Plus Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below.

Enumeration Value	Description
	[*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the plus element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
quadArrow (Quad-Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the quadArrow element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
quadArrowCallout (Callout Quad-Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the quadArrowCallout element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]

Enumeration Value	Description
	
rect (Rectangle Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the rect element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
ribbon (Ribbon Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the ribbon element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
ribbon2 (Ribbon 2 Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the ribbon2 element in the preset shape geometries

Enumeration Value	Description
	electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
rightArrow (Right Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the rightArrow element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
rightArrowCallout (Callout Right Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the rightArrowCallout element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
rightBrace (Right Brace Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in

Enumeration Value	Description
	the rightBrace element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
rightBracket (Right Bracket Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the rightBracket element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
round1Rect (One Round Corner Rectangle Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the round1Rect element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
round2DiagRect (Two Diagonal Round Corner Rectangle Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below.

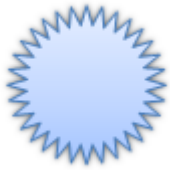
Enumeration Value	Description
	[*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the round2DiagRect element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
round2SameRect (Two Same-side Round Corner Rectangle Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the round2SameRect element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
roundRect (Round Corner Rectangle Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the roundRect element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
rtTriangle (Right Triangle Shape)	Specifies a preset shape geometry. This geometry shall

Enumeration Value	Description
	be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the rtTriangle element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
smileyFace (Smiley Face Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the smileyFace element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
snip1Rect (One Snip Corner Rectangle Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the snip1Rect element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]

Enumeration Value	Description
	
snip2DiagRect (Two Diagonal Snip Corner Rectangle Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the snip2DiagRect element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
snip2SameRect (Two Same-side Snip Corner Rectangle Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the snip2SameRect element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
snipRoundRect (One Snip One Round Corner Rectangle Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the snipRoundRect element in the preset shape geometries electronic addenda of Annex D. The

Enumeration Value	Description
	constants used in that markup are guides that are described in further detail above. *end note*] 
squareTabs (Square Tabs Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the squareTabs element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
star10 (Ten Pointed Star Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the star10 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
star12 (Twelve Pointed Star Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in

Enumeration Value	Description
	the star12 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
star16 (Sixteen Pointed Star Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the star16 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
star24 (Twenty Four Pointed Star Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the star24 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
star32 (Thirty Two Pointed Star Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below.

Enumeration Value	Description
	[*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the star32 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
star4 (Four Pointed Star Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the star4 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
star5 (Five Pointed Star Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the star5 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]

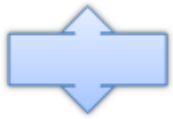
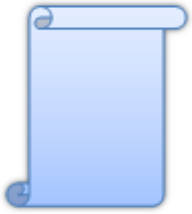
Enumeration Value	Description
	
star6 (Six Pointed Star Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the star6 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
star7 (Seven Pointed Star Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the star7 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
star8 (Eight Pointed Star Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the star8 element in the preset shape geometries electronic addenda of Annex D. The constants used in

Enumeration Value	Description
	that markup are guides that are described in further detail above. *end note*] 
straightConnector1 (Straight Connector 1 Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the straightConnector1 element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
stripedRightArrow (Striped Right Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the stripedRightArrow element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
sun (Sun Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in

Enumeration Value	Description
	the sun element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
swooshArrow (Swoosh Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the swooshArrow element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
teardrop (Teardrop Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the teardrop element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
trapezoid (Trapezoid Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below.

Enumeration Value	Description
	[*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the trapezoid element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
triangle (Triangle Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the triangle element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
upArrow (Up Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the upArrow element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]

Enumeration Value	Description
	
upArrowCallout (Callout Up Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the upArrowCallout element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
upDownArrow (Up Down Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the upDownArrow element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
upDownArrowCallout (Callout Up Down Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in

Enumeration Value	Description
	the upDownArrowCallout element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
uturnArrow (U-Turn Arrow Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the uturnArrow element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
verticalScroll (Vertical Scroll Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the verticalScroll element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
wave (Wave Shape)	Specifies a preset shape geometry. This geometry shall

Enumeration Value	Description
	be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the wave element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
wedgeEllipseCallout (Callout Wedge Ellipse Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the wedgeEllipseCallout element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 
wedgeRectCallout (Callout Wedge Rectangle Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the wedgeRectCallout element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]

Enumeration Value	Description
	
wedgeRoundRectCallout (Callout Wedge Round Rectangle Shape)	Specifies a preset shape geometry. This geometry shall be designed to match the normative image below. [*Note*: An example of DrawingML which may be used to generate this preset shape definition is contained in the wedgeRoundRectCallout element in the preset shape geometries electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*] 

Referenced By
prstGeom@prst (§5.1.11.18); ST_LayoutShapeType (§5.9.7.40)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ShapeType">
   <restriction base="xsd:token">
      <enumeration value="line"/>
      <enumeration value="lineInv"/>
      <enumeration value="triangle"/>
      <enumeration value="rtTriangle"/>
      <enumeration value="rect"/>
      <enumeration value="diamond"/>
      <enumeration value="parallelogram"/>
      <enumeration value="trapezoid"/>
      <enumeration value="nonIsoscelesTrapezoid"/>
      <enumeration value="pentagon"/>
      <enumeration value="hexagon"/>
      <enumeration value="heptagon"/>
      <enumeration value="octagon"/>
      <enumeration value="decagon"/>
      <enumeration value="dodecagon"/>
      <enumeration value="star4"/>
      <enumeration value="star5"/>
      <enumeration value="star6"/>
      <enumeration value="star7"/>
      <enumeration value="star8"/>
      <enumeration value="star10"/>
      <enumeration value="star12"/>
      <enumeration value="star16"/>
      <enumeration value="star24"/>
      <enumeration value="star32"/>
      <enumeration value="roundRect"/>
      <enumeration value="round1Rect"/>
      <enumeration value="round2SameRect"/>
      <enumeration value="round2DiagRect"/>
      <enumeration value="snipRoundRect"/>
      <enumeration value="snip1Rect"/>
      <enumeration value="snip2SameRect"/>
      <enumeration value="snip2DiagRect"/>
      <enumeration value="plaque"/>
      <enumeration value="ellipse"/>
      <enumeration value="teardrop"/>
      <enumeration value="homePlate"/>
      <enumeration value="chevron"/>
      <enumeration value="pieWedge"/>
      <enumeration value="pie"/>
      <enumeration value="blockArc"/>
      <enumeration value="donut"/>
      <enumeration value="noSmoking"/>
      <enumeration value="rightArrow"/>
      <enumeration value="leftArrow"/>
      <enumeration value="upArrow"/>
      <enumeration value="downArrow"/>
      <enumeration value="stripedRightArrow"/>
      <enumeration value="notchedRightArrow"/>
      <enumeration value="bentUpArrow"/>
```

```
        <enumeration value="leftRightArrow"/>
        <enumeration value="upDownArrow"/>
        <enumeration value="leftUpArrow"/>
        <enumeration value="leftRightUpArrow"/>
        <enumeration value="quadArrow"/>
        <enumeration value="leftArrowCallout"/>
        <enumeration value="rightArrowCallout"/>
        <enumeration value="upArrowCallout"/>
        <enumeration value="downArrowCallout"/>
        <enumeration value="leftRightArrowCallout"/>
        <enumeration value="upDownArrowCallout"/>
        <enumeration value="quadArrowCallout"/>
        <enumeration value="bentArrow"/>
        <enumeration value="uturnArrow"/>
        <enumeration value="circularArrow"/>
        <enumeration value="leftCircularArrow"/>
        <enumeration value="leftRightCircularArrow"/>
        <enumeration value="curvedRightArrow"/>
        <enumeration value="curvedLeftArrow"/>
        <enumeration value="curvedUpArrow"/>
        <enumeration value="curvedDownArrow"/>
        <enumeration value="swooshArrow"/>
        <enumeration value="cube"/>
        <enumeration value="can"/>
        <enumeration value="lightningBolt"/>
        <enumeration value="heart"/>
        <enumeration value="sun"/>
        <enumeration value="moon"/>
        <enumeration value="smileyFace"/>
        <enumeration value="irregularSeal1"/>
        <enumeration value="irregularSeal2"/>
        <enumeration value="foldedCorner"/>
        <enumeration value="bevel"/>
        <enumeration value="frame"/>
        <enumeration value="halfFrame"/>
        <enumeration value="corner"/>
        <enumeration value="diagStripe"/>
        <enumeration value="chord"/>
        <enumeration value="arc"/>
        <enumeration value="leftBracket"/>
        <enumeration value="rightBracket"/>
        <enumeration value="leftBrace"/>
        <enumeration value="rightBrace"/>
        <enumeration value="bracketPair"/>
        <enumeration value="bracePair"/>
        <enumeration value="straightConnector1"/>
        <enumeration value="bentConnector2"/>
        <enumeration value="bentConnector3"/>
        <enumeration value="bentConnector4"/>
        <enumeration value="bentConnector5"/>
        <enumeration value="curvedConnector2"/>
        <enumeration value="curvedConnector3"/>
        <enumeration value="curvedConnector4"/>
```

```
        <enumeration value="curvedConnector5"/>
        <enumeration value="callout1"/>
        <enumeration value="callout2"/>
        <enumeration value="callout3"/>
        <enumeration value="accentCallout1"/>
        <enumeration value="accentCallout2"/>
        <enumeration value="accentCallout3"/>
        <enumeration value="borderCallout1"/>
        <enumeration value="borderCallout2"/>
        <enumeration value="borderCallout3"/>
        <enumeration value="accentBorderCallout1"/>
        <enumeration value="accentBorderCallout2"/>
        <enumeration value="accentBorderCallout3"/>
        <enumeration value="wedgeRectCallout"/>
        <enumeration value="wedgeRoundRectCallout"/>
        <enumeration value="wedgeEllipseCallout"/>
        <enumeration value="cloudCallout"/>
        <enumeration value="cloud"/>
        <enumeration value="ribbon"/>
        <enumeration value="ribbon2"/>
        <enumeration value="ellipseRibbon"/>
        <enumeration value="ellipseRibbon2"/>
        <enumeration value="leftRightRibbon"/>
        <enumeration value="verticalScroll"/>
        <enumeration value="horizontalScroll"/>
        <enumeration value="wave"/>
        <enumeration value="doubleWave"/>
        <enumeration value="plus"/>
        <enumeration value="flowChartProcess"/>
        <enumeration value="flowChartDecision"/>
        <enumeration value="flowChartInputOutput"/>
        <enumeration value="flowChartPredefinedProcess"/>
        <enumeration value="flowChartInternalStorage"/>
        <enumeration value="flowChartDocument"/>
        <enumeration value="flowChartMultidocument"/>
        <enumeration value="flowChartTerminator"/>
        <enumeration value="flowChartPreparation"/>
        <enumeration value="flowChartManualInput"/>
        <enumeration value="flowChartManualOperation"/>
        <enumeration value="flowChartConnector"/>
        <enumeration value="flowChartPunchedCard"/>
        <enumeration value="flowChartPunchedTape"/>
        <enumeration value="flowChartSummingJunction"/>
        <enumeration value="flowChartOr"/>
        <enumeration value="flowChartCollate"/>
        <enumeration value="flowChartSort"/>
        <enumeration value="flowChartExtract"/>
        <enumeration value="flowChartMerge"/>
        <enumeration value="flowChartOfflineStorage"/>
        <enumeration value="flowChartOnlineStorage"/>
        <enumeration value="flowChartMagneticTape"/>
        <enumeration value="flowChartMagneticDisk"/>
        <enumeration value="flowChartMagneticDrum"/>
```

```
        <enumeration value="flowChartDisplay"/>
        <enumeration value="flowChartDelay"/>
        <enumeration value="flowChartAlternateProcess"/>
        <enumeration value="flowChartOffpageConnector"/>
        <enumeration value="actionButtonBlank"/>
        <enumeration value="actionButtonHome"/>
        <enumeration value="actionButtonHelp"/>
        <enumeration value="actionButtonInformation"/>
        <enumeration value="actionButtonForwardNext"/>
        <enumeration value="actionButtonBackPrevious"/>
        <enumeration value="actionButtonEnd"/>
        <enumeration value="actionButtonBeginning"/>
        <enumeration value="actionButtonReturn"/>
        <enumeration value="actionButtonDocument"/>
        <enumeration value="actionButtonSound"/>
        <enumeration value="actionButtonMovie"/>
        <enumeration value="gear6"/>
        <enumeration value="gear9"/>
        <enumeration value="funnel"/>
        <enumeration value="mathPlus"/>
        <enumeration value="mathMinus"/>
        <enumeration value="mathMultiply"/>
        <enumeration value="mathDivide"/>
        <enumeration value="mathEqual"/>
        <enumeration value="mathNotEqual"/>
        <enumeration value="cornerTabs"/>
        <enumeration value="squareTabs"/>
        <enumeration value="plaqueTabs"/>
        <enumeration value="chartX"/>
        <enumeration value="chartStar"/>
        <enumeration value="chartPlus"/>
    </restriction>
</simpleType>
```

5.1.12.57 ST_StyleMatrixColumnIndex (Style Matrix Column Index)

This simple type specifies an index into one of the lists in the style matrix specified by the fmtScheme element (bgFillStyleLst, effectStyleLst, fillStyleLst, or lnStyleLst).

This simple type's contents are a restriction of the XML Schema unsignedInt datatype.

Referenced By
bgRef@idx (§4.4.1.3); effectRef@idx (§5.1.4.2.8); fillRef@idx (§5.1.4.2.10); lnRef@idx (§5.1.4.2.19)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_StyleMatrixColumnIndex">
    <restriction base="xsd:unsignedInt"/>
</simpleType>
```

5.1.12.58 ST_SystemColorVal (System Color Value)

This simple type specifies a system color value. This color is based upon the value that this color currently has within the system on which the document is being viewed.

Applications shall use the lastClr attribute to determine the absolute value of the last color used if system colors are not supported.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
3dDkShadow (3D Dark System Color)	Specifies a Dark shadow color for three-dimensional display elements.
3dLight (3D Light System Color)	Specifies a Light color for three-dimensional display elements (for edges facing the light source).
activeBorder (Active Border System Color)	Specifies an Active Window Border Color.
activeCaption (Active Caption System Color)	Specifies the active window title bar color. In particular the left side color in the color gradient of an active window's title bar if the gradient effect is enabled.
appWorkspace (Application Workspace System Color)	Specifies the Background color of multiple document interface (MDI) applications.
background (Background System Color)	Specifies the desktop background color.
btnFace (Button Face System Color)	Specifies the face color for three-dimensional display elements and for dialog box backgrounds.
btnHighlight (Button Highlight System Color)	Specifies the highlight color for three-dimensional display elements (for edges facing the light source).
btnShadow (Button Shadow System Color)	Specifies the shadow color for three-dimensional display elements (for edges facing away from the light source).
btnText (Button Text System Color)	Specifies the color of text on push buttons.
captionText (Caption Text System Color)	Specifies the color of text in the caption, size box, and scroll bar arrow box.
gradientActiveCaption (Gradient Active Caption System Color)	Specifies the right side color in the color gradient of an active window's title bar.
gradientInactiveCaption (Gradient Inactive Caption System Color)	Specifies the right side color in the color gradient of an inactive window's title bar.
grayText (Gray Text System Color)	Specifies a grayed (disabled) text. This color is set to 0 if the current display driver does not support a solid gray color.
highlight (Highlight System Color)	Specifies the color of Item(s) selected in a control.

Enumeration Value	Description
highlightText (Highlight Text System Color)	Specifies the text color of item(s) selected in a control.
hotLight (Hot Light System Color)	Specifies the color for a hyperlink or hot-tracked item.
inactiveBorder (Inactive Border System Color)	Specifies the color of the Inactive window border.
inactiveCaption (Inactive Caption System Color)	Specifies the color of the Inactive window caption. Specifies the left side color in the color gradient of an inactive window's title bar if the gradient effect is enabled.
inactiveCaptionText (Inactive Caption Text System Color)	Specifies the color of text in an inactive caption.
infoBk (Info Back System Color)	Specifies the background color for tooltip controls.
infoText (Info Text System Color)	Specifies the text color for tooltip controls.
menu (Menu System Color)	Specifies the menu background color.
menuBar (Menu Bar System Color)	Specifies the background color for the menu bar when menus appear as flat menus.
menuHighlight (Menu Highlight System Color)	Specifies the color used to highlight menu items when the menu appears as a flat menu.
menuText (Menu Text System Color)	Specifies the color of Text in menus.
scrollBar (Scroll Bar System Color)	Specifies the scroll bar gray area color.
window (Window System Color)	Specifies window background color.
windowFrame (Window Frame System Color)	Specifies the window frame color.
windowText (Window Text System Color)	Specifies the color of text in windows.

Referenced By
sysClr@val (§5.1.2.2.33)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SystemColorVal">
   <restriction base="xsd:token">
      <enumeration value="scrollBar"/>
      <enumeration value="background"/>
      <enumeration value="activeCaption"/>
      <enumeration value="inactiveCaption"/>
      <enumeration value="menu"/>
      <enumeration value="window"/>
      <enumeration value="windowFrame"/>
      <enumeration value="menuText"/>
      <enumeration value="windowText"/>
      <enumeration value="captionText"/>
      <enumeration value="activeBorder"/>
      <enumeration value="inactiveBorder"/>
      <enumeration value="appWorkspace"/>
      <enumeration value="highlight"/>
      <enumeration value="highlightText"/>
      <enumeration value="btnFace"/>
      <enumeration value="btnShadow"/>
      <enumeration value="grayText"/>
      <enumeration value="btnText"/>
      <enumeration value="inactiveCaptionText"/>
      <enumeration value="btnHighlight"/>
      <enumeration value="3dDkShadow"/>
      <enumeration value="3dLight"/>
      <enumeration value="infoText"/>
      <enumeration value="infoBk"/>
      <enumeration value="hotLight"/>
      <enumeration value="gradientActiveCaption"/>
      <enumeration value="gradientInactiveCaption"/>
      <enumeration value="menuHighlight"/>
      <enumeration value="menuBar"/>
   </restriction>
</simpleType>
```

5.1.12.59 ST_TextAlignType (Text Alignment Types)

This type specifies the text alignment types

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
ctr (Text Alignment Enum (Center))	Align text in the center.
dist (Text Alignment Enum (Distributed))	Distributes the text words across an entire text line.
just (Text Alignment Enum (Justified))	Align text so that it is justified across the whole line. It is smart in the sense that it will not justify sentences which are short.

Enumeration Value	Description
justLow (Text Alignment Enum (Justified Low))	Aligns the text with an adjusted kashida length for Arabic text.
l (Text Alignment Enum (Left))	Align text to the left margin.
r (Text Alignment Enum (Right))	Align text to the right margin.
thaiDist (Text Alignment Enum (Thai Distributed))	Distributes Thai text specially, because each character is treated as a word.

Referenced By
defPPr@algn (§5.1.5.2.2); lvl1pPr@algn (§5.1.5.4.13); lvl2pPr@algn (§5.1.5.4.14); lvl3pPr@algn (§5.1.5.4.15); lvl4pPr@algn (§5.1.5.4.16); lvl5pPr@algn (§5.1.5.4.17); lvl6pPr@algn (§5.1.5.4.18); lvl7pPr@algn (§5.1.5.4.19); lvl8pPr@algn (§5.1.5.4.20); lvl9pPr@algn (§5.1.5.4.21); pPr@algn (§5.1.5.2.7)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextAlignType">
   <restriction base="xsd:token">
      <enumeration value="l"/>
      <enumeration value="ctr"/>
      <enumeration value="r"/>
      <enumeration value="just"/>
      <enumeration value="justLow"/>
      <enumeration value="dist"/>
      <enumeration value="thaiDist"/>
   </restriction>
</simpleType>
```

5.1.12.60 ST_TextAnchoringType (Text Anchoring Types)

This type specifies a list of available anchoring types for text.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
b (Text Anchor Enum (Bottom))	Anchor the text at the bottom of the bounding rectangle.
ctr (Text Anchor Enum (Center))	Anchor the text at the middle of the bounding rectangle.
dist (Text Anchor Enum (Distributed))	Anchor the text so that it is distributed vertically. When text is horizontal, this spaces out the actual lines of text and is almost always identical in behavior to anchorJustified (special case: if only 1 line, then anchored in middle). When text is vertical, then it distributes the letters vertically. This is different than

Enumeration Value	Description
	anchorJustified, because it always forces distribution of the words, even if there are only one or two words in a line.
just (Text Anchor Enum (Justified))	Anchor the text so that it is justified vertically. When text is horizontal, this spaces out the actual lines of text and is almost always identical in behavior to 'distrib' (special case: if only 1 line, then anchored at top). When text is vertical, then it justifies the letters vertically. This is different than anchorDistributed, because in some cases such as very little text in a line, it will not justify.
t (Text Anchoring Type Enum (Top))	Anchor the text at the top of the bounding rectangle.

Referenced By
bodyPr@anchor (§5.1.5.1.1); tcPr@anchor (§5.1.6.15)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextAnchoringType">
   <restriction base="xsd:token">
      <enumeration value="t"/>
      <enumeration value="ctr"/>
      <enumeration value="b"/>
      <enumeration value="just"/>
      <enumeration value="dist"/>
   </restriction>
</simpleType>
```

5.1.12.61 ST_TextAutonumberScheme (Text Auto-number Schemes)

This type specifies a list of automatic numbering schemes.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
alphaLcParenBoth (Autonumber Enum (alphaLcParenBoth))	(a), (b), (c), …
alphaLcParenR (Autonumbering Enum (alphaLcParenR))	a), b), c), …
alphaLcPeriod (Autonumbering Enum (alphaLcPeriod))	a., b., c., …
alphaUcParenBoth (Autonumbering Enum (	(A), (B), (C), …

Enumeration Value	Description
alphaUcParenBoth))	
alphaUcParenR (Autonumbering Enum (alphaUcParenR))	A), B), C), …
alphaUcPeriod (Autonumbering Enum (alphaUcPeriod))	A., B., C., …
arabic1Minus (Autonumbering Enum (arabic1Minus))	Bidi Arabic 1 (AraAlpha) with ANSI minus symbol
arabic2Minus (Autonumbering Enum (arabic2Minus))	Bidi Arabic 2 (AraAbjad) with ANSI minus symbol
arabicDbPeriod (Autonumbering Enum (arabicDbPeriod))	Dbl-byte Arabic numbers w/ double-byte period
arabicDbPlain (Autonumbering Enum (arabicDbPlain))	Dbl-byte Arabic numbers
arabicParenBoth (Autonumbering Enum (arabicParenBoth))	(1), (2), (3), …
arabicParenR (Autonumbering Enum (arabicParenR))	1), 2), 3), …
arabicPeriod (Autonumbering Enum (arabicPeriod))	1., 2., 3., …
arabicPlain (Autonumbering Enum (arabicPlain))	1, 2, 3, …
circleNumDbPlain (Autonumbering Enum (circleNumDbPlain))	Dbl-byte circle numbers (1-10 circle[0x2460-], 11- arabic numbers)
circleNumWdBlackPlain (Autonumbering Enum (circleNumWdBlackPlain))	Wingdings black circle numbers
circleNumWdWhitePlain (Autonumbering Enum (circleNumWdWhitePlain))	Wingdings white circle numbers (0-10 circle[0x0080-], 11- arabic numbers)
ea1ChsPeriod (Autonumbering Enum (ea1ChsPeriod))	EA: Simplified Chinese w/ single-byte period
ea1ChsPlain (Autonumbering Enum (ea1ChsPlain))	EA: Simplified Chinese (TypeA 1-99, TypeC 100-)
ea1ChtPeriod (Autonumbering Enum (ea1ChtPeriod))	EA: Traditional Chinese w/ single-byte period
ea1ChtPlain (Autonumbering Enum (ea1ChtPlain))	EA: Traditional Chinese (TypeA 1-19, TypeC 20-)
ea1JpnChsDbPeriod (Autonumbering Enum (ea1JpnChsDbPeriod))	EA: Japanese w/ double-byte period
ea1JpnKorPeriod (Autonumbering Enum (ea1JpnKorPeriod))	EA: Japanese/Korean w/ single-byte period
ea1JpnKorPlain (Autonumbering Enum (ea1JpnKorPlain))	EA: Japanese/Korean (TypeC 1-)
hebrew2Minus (Autonumbering Enum (hebrew2Minus))	Bidi Hebrew 2 with ANSI minus symbol

Enumeration Value	Description
hindiAlpha1Period (Autonumbering Enum (hindiAlpha1Period))	Hindi alphabet period - consonants
hindiAlphaPeriod (Autonumbering Enum (hindiAlphaPeriod))	Hindi alphabet period - vowels
hindiNumParenR (Autonumbering Enum (hindiNumParenR))	Hindi numerical parentheses - right
hindiNumPeriod (Autonumbering Enum (hindiNumPeriod))	Hindi numerical period
romanLcParenBoth (Autonumbering Enum (romanLcParenBoth))	(i), (ii), (iii), …
romanLcParenR (Autonumbering Enum (romanLcParenR))	i), ii), iii), …
romanLcPeriod (Autonumbering Enum (romanLcPeriod))	i., ii., iii., …
romanUcParenBoth (Autonumbering Enum (romanUcParenBoth))	(I), (II), (III), …
romanUcParenR (Autonumbering Enum (romanUcParenR))	I), II), III), …
romanUcPeriod (Autonumbering Enum (romanUcPeriod))	I., II., III., …
thaiAlphaParenBoth (Autonumbering Enum (thaiAlphaParenBoth))	Thai alphabet parentheses - both
thaiAlphaParenR (Autonumbering Enum (thaiAlphaParenR))	Thai alphabet parentheses - right
thaiAlphaPeriod (Autonumbering Enum (thaiAlphaPeriod))	Thai alphabet period
thaiNumParenBoth (Autonumbering Enum (thaiNumParenBoth))	Thai numerical parentheses - both
thaiNumParenR (Autonumbering Enum (thaiNumParenR))	Thai numerical parentheses - right
thaiNumPeriod (Autonumbering Enum (thaiNumPeriod))	Thai numerical period

Referenced By
buAutoNum@type (§5.1.5.4.1)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextAutonumberScheme">
   <restriction base="xsd:token">
      <enumeration value="alphaLcParenBoth"/>
      <enumeration value="alphaUcParenBoth"/>
      <enumeration value="alphaLcParenR"/>
      <enumeration value="alphaUcParenR"/>
      <enumeration value="alphaLcPeriod"/>
      <enumeration value="alphaUcPeriod"/>
      <enumeration value="arabicParenBoth"/>
      <enumeration value="arabicParenR"/>
      <enumeration value="arabicPeriod"/>
      <enumeration value="arabicPlain"/>
      <enumeration value="romanLcParenBoth"/>
      <enumeration value="romanUcParenBoth"/>
      <enumeration value="romanLcParenR"/>
      <enumeration value="romanUcParenR"/>
      <enumeration value="romanLcPeriod"/>
      <enumeration value="romanUcPeriod"/>
      <enumeration value="circleNumDbPlain"/>
      <enumeration value="circleNumWdBlackPlain"/>
      <enumeration value="circleNumWdWhitePlain"/>
      <enumeration value="arabicDbPeriod"/>
      <enumeration value="arabicDbPlain"/>
      <enumeration value="ea1ChsPeriod"/>
      <enumeration value="ea1ChsPlain"/>
      <enumeration value="ea1ChtPeriod"/>
      <enumeration value="ea1ChtPlain"/>
      <enumeration value="ea1JpnChsDbPeriod"/>
      <enumeration value="ea1JpnKorPlain"/>
      <enumeration value="ea1JpnKorPeriod"/>
      <enumeration value="arabic1Minus"/>
      <enumeration value="arabic2Minus"/>
      <enumeration value="hebrew2Minus"/>
      <enumeration value="thaiAlphaPeriod"/>
      <enumeration value="thaiAlphaParenR"/>
      <enumeration value="thaiAlphaParenBoth"/>
      <enumeration value="thaiNumPeriod"/>
      <enumeration value="thaiNumParenR"/>
      <enumeration value="thaiNumParenBoth"/>
      <enumeration value="hindiAlphaPeriod"/>
      <enumeration value="hindiNumPeriod"/>
      <enumeration value="hindiNumParenR"/>
      <enumeration value="hindiAlpha1Period"/>
   </restriction>
</simpleType>
```

5.1.12.62 ST_TextBulletSizePercent (Bullet Size Percentage)

This type specifies the range that the bullet percent can be. A bullet percent is the size of the bullet with respect to the text that should follow it. 25000 = 25 %400000 = 400 %

This simple type's contents are a restriction of the ST_Percentage simple type (§5.1.12.41).

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 25000.
- This simple type has a maximum value of less than or equal to 400000.

Referenced By
buSzPct@val (§5.1.5.4.9)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextBulletSizePercent">
   <restriction base="ST_Percentage">
      <minInclusive value="25000"/>
      <maxInclusive value="400000"/>
   </restriction>
</simpleType>
```

5.1.12.63 ST_TextBulletStartAtNum (Start Bullet At Number)

This type specifies the range that the start at number for a bullet's auto-numbering sequence can begin at. When the numbering is alphabetical, then the numbers map to the appropriate letter. 1->a, 2->b, etc. If the numbers go above 26, then the numbers begin to double up. For example, 27->aa and 53->aaa.

This simple type's contents are a restriction of the XML Schema int datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 1.
- This simple type has a maximum value of less than or equal to 32767.

Referenced By
buAutoNum@startAt (§5.1.5.4.1)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextBulletStartAtNum">
   <restriction base="xsd:int">
      <minInclusive value="1"/>
      <maxInclusive value="32767"/>
   </restriction>
</simpleType>
```

5.1.12.64 ST_TextCapsType (Text Cap Types)

This type specifies the cap types of the text.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
all (Text Caps Enum (All))	Apply all caps on the text. All lower case letters are converted to upper case even though they are stored differently in the backing store.
none (Text Caps Enum (None))	The reason we cannot implicitly have noCaps be the scenario where capitalization is not specified is because not being specified implies deriving from a particular style and the user might want to override that and make some text not have a capitalization scheme even though the style says otherwise.
small (Text Caps Enum (Small))	Apply small caps to the text. All letters are converted to lower case.

Referenced By
defRPr@cap (§5.1.5.3.2); endParaRPr@cap (§5.1.5.2.3); rPr@cap (§5.1.5.3.9)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextCapsType">
   <restriction base="xsd:token">
      <enumeration value="none"/>
      <enumeration value="small"/>
      <enumeration value="all"/>
   </restriction>
</simpleType>
```

5.1.12.65 ST_TextColumnCount (Text Column Count)

This type specifies the number of columns.

This simple type's contents are a restriction of the XML Schema int datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 1.
- This simple type has a maximum value of less than or equal to 16.

Referenced By
bodyPr@numCol (§5.1.5.1.1)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextColumnCount">
   <restriction base="xsd:int">
      <minInclusive value="1"/>
      <maxInclusive value="16"/>
   </restriction>
</simpleType>
```

5.1.12.66 ST_TextFontAlignType (Font Alignment Types)

This type specifies the different types of font alignment.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
auto (Font Alignment Enum (Automatic))	When the text flow is horizontal or simple vertical same as fontBaseline but for other vertical modes same as fontCenter.
b (Font Alignment Enum (Bottom))	The letters are anchored to the very bottom of a single line. This is different than the bottom baseline because of letters such as "g," "q," "y," etc.
base (Font Alignment Enum (Baseline))	The letters are anchored to the bottom baseline of a single line.
ctr (Font Alignment Enum (Center))	The letters are anchored between the two baselines of a single line.
t (Font Alignment Enum (Top))	The letters are anchored to the top baseline of a single line.

Referenced By
defPPr@fontAlgn (§5.1.5.2.2); lvl1pPr@fontAlgn (§5.1.5.4.13); lvl2pPr@fontAlgn (§5.1.5.4.14); lvl3pPr@fontAlgn (§5.1.5.4.15); lvl4pPr@fontAlgn (§5.1.5.4.16); lvl5pPr@fontAlgn (§5.1.5.4.17); lvl6pPr@fontAlgn (§5.1.5.4.18); lvl7pPr@fontAlgn (§5.1.5.4.19); lvl8pPr@fontAlgn (§5.1.5.4.20); lvl9pPr@fontAlgn (§5.1.5.4.21); pPr@fontAlgn (§5.1.5.2.7)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextFontAlignType">
   <restriction base="xsd:token">
      <enumeration value="auto"/>
      <enumeration value="t"/>
      <enumeration value="ctr"/>
      <enumeration value="base"/>
      <enumeration value="b"/>
   </restriction>
</simpleType>
```

5.1.12.67 ST_TextFontScalePercent (Text Font Scale Percentage)

This type specifies the percentage range text can be scaled to in order to fit.

This simple type's contents are a restriction of the ST_Percentage simple type (§5.1.12.41).

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 1000.
- This simple type has a maximum value of less than or equal to 100000.

Referenced By
normAutofit@fontScale (§5.1.5.1.3)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextFontScalePercent">
   <restriction base="ST_Percentage">
      <minInclusive value="1000"/>
      <maxInclusive value="100000"/>
   </restriction>
</simpleType>
```

5.1.12.68 ST_TextFontSize (Text Font Size)

This type specifies the size of any text in hundredths of a point. Must be at least 1 point.

This simple type's contents are a restriction of the XML Schema int datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 100.
- This simple type has a maximum value of less than or equal to 400000.

Referenced By
buSzPts@val (§5.1.5.4.10); defRPr@sz (§5.1.5.3.2); endParaRPr@sz (§5.1.5.2.3); rPr@sz (§5.1.5.3.9)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextFontSize">
   <restriction base="xsd:int">
      <minInclusive value="100"/>
      <maxInclusive value="400000"/>
   </restriction>
</simpleType>
```

5.1.12.69 ST_TextHorzOverflowType (Text Horizontal Overflow Types)

This type specifies the text horizontal overflow types

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
clip (Text Horizontal Overflow Enum (Clip))	When a big character does not fit into a line, clip it at the proper horizontal overflow.
overflow (Text Horizontal Overflow Enum (Overflow	When a big character does not fit into a line, allow a

Enumeration Value	Description
))	horizontal overflow.

Referenced By
bodyPr@horzOverflow (§5.1.5.1.1); tcPr@horzOverflow (§5.1.6.15)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextHorzOverflowType">
    <restriction base="xsd:token">
        <enumeration value="overflow"/>
        <enumeration value="clip"/>
    </restriction>
</simpleType>
```

5.1.12.70 ST_TextIndent (Text Indentation)

This type specifies the text indentation amount to be used.

NOTE: The units of measurement used here are EMUs (English Metric Units).

This simple type's contents are a restriction of the ST_Coordinate32 simple type (§5.1.12.17).

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to -51206400.
- This simple type has a maximum value of less than or equal to 51206400.

Referenced By
defPPr@indent (§5.1.5.2.2); lvl1pPr@indent (§5.1.5.4.13); lvl2pPr@indent (§5.1.5.4.14); lvl3pPr@indent (§5.1.5.4.15); lvl4pPr@indent (§5.1.5.4.16); lvl5pPr@indent (§5.1.5.4.17); lvl6pPr@indent (§5.1.5.4.18); lvl7pPr@indent (§5.1.5.4.19); lvl8pPr@indent (§5.1.5.4.20); lvl9pPr@indent (§5.1.5.4.21); pPr@indent (§5.1.5.2.7)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextIndent">
    <restriction base="ST_Coordinate32">
        <minInclusive value="-51206400"/>
        <maxInclusive value="51206400"/>
    </restriction>
</simpleType>
```

5.1.12.71 ST_TextIndentLevelType (Text Indent Level Type)

This type specifies the indent level type. We support list level 0 to 8, and we use -1 and -2 for outline mode levels that should only exist in memory.

This simple type's contents are a restriction of the XML Schema int datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 0.
- This simple type has a maximum value of less than or equal to 8.

Referenced By
defPPr@lvl (§5.1.5.2.2); lvl1pPr@lvl (§5.1.5.4.13); lvl2pPr@lvl (§5.1.5.4.14); lvl3pPr@lvl (§5.1.5.4.15); lvl4pPr@lvl (§5.1.5.4.16); lvl5pPr@lvl (§5.1.5.4.17); lvl6pPr@lvl (§5.1.5.4.18); lvl7pPr@lvl (§5.1.5.4.19); lvl8pPr@lvl (§5.1.5.4.20); lvl9pPr@lvl (§5.1.5.4.21); pPr@lvl (§5.1.5.2.7)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextIndentLevelType">
   <restriction base="xsd:int">
      <minInclusive value="0"/>
      <maxInclusive value="8"/>
   </restriction>
</simpleType>
```

5.1.12.72 ST_TextLanguageID (Language ID)

This type specifies an ID representing a particular language. Example: "en-us"

This simple type's contents are a restriction of the XML Schema string datatype.

Referenced By
defRPr@altLang (§5.1.5.3.2); defRPr@lang (§5.1.5.3.2); endParaRPr@altLang (§5.1.5.2.3); endParaRPr@lang (§5.1.5.2.3); rPr@altLang (§5.1.5.3.9); rPr@lang (§5.1.5.3.9)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextLanguageID">
   <restriction base="xsd:string"/>
</simpleType>
```

5.1.12.73 ST_TextMargin (Text Margin)

This type specifies the margin that will be used and its corresponding size.

This simple type's contents are a restriction of the ST_Coordinate32 simple type (§5.1.12.17).

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 0.
- This simple type has a maximum value of less than or equal to 51206400.

Referenced By
defPPr@marL (§5.1.5.2.2); defPPr@marR (§5.1.5.2.2); lvl1pPr@marL (§5.1.5.4.13); lvl1pPr@marR (§5.1.5.4.13); lvl2pPr@marL (§5.1.5.4.14); lvl2pPr@marR (§5.1.5.4.14); lvl3pPr@marL (§5.1.5.4.15); lvl3pPr@marR (§5.1.5.4.15); lvl4pPr@marL (§5.1.5.4.16); lvl4pPr@marR (§5.1.5.4.16); lvl5pPr@marL

Referenced By
(§5.1.5.4.17); lvl5pPr@marR (§5.1.5.4.17); lvl6pPr@marL (§5.1.5.4.18); lvl6pPr@marR (§5.1.5.4.18); lvl7pPr@marL (§5.1.5.4.19); lvl7pPr@marR (§5.1.5.4.19); lvl8pPr@marL (§5.1.5.4.20); lvl8pPr@marR (§5.1.5.4.20); lvl9pPr@marL (§5.1.5.4.21); lvl9pPr@marR (§5.1.5.4.21); pPr@marL (§5.1.5.2.7); pPr@marR (§5.1.5.2.7)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextMargin">
   <restriction base="ST_Coordinate32">
      <minInclusive value="0"/>
      <maxInclusive value="51206400"/>
   </restriction>
</simpleType>
```

5.1.12.74 ST_TextNonNegativePoint (Text Non-Negative Point)

This type specifies a non-negative font size in hundredths of a point. This is restricted to the range [0, 400000].

This simple type's contents are a restriction of the XML Schema int datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 0.
- This simple type has a maximum value of less than or equal to 400000.

Referenced By
defRPr@kern (§5.1.5.3.2); endParaRPr@kern (§5.1.5.2.3); rPr@kern (§5.1.5.3.9)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextNonNegativePoint">
   <restriction base="xsd:int">
      <minInclusive value="0"/>
      <maxInclusive value="400000"/>
   </restriction>
</simpleType>
```

5.1.12.75 ST_TextPoint (Text Point)

This type specifies a font size in hundredths of a point. This is restricted to the range [-400000, 400000], i.e from -4000 pt to 4000 pt.

This simple type's contents are a restriction of the XML Schema int datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to -400000.
- This simple type has a maximum value of less than or equal to 400000.

Referenced By
defRPr@spc (§5.1.5.3.2); endParaRPr@spc (§5.1.5.2.3); rPr@spc (§5.1.5.3.9)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextPoint">
   <restriction base="xsd:int">
      <minInclusive value="-400000"/>
      <maxInclusive value="400000"/>
   </restriction>
</simpleType>
```

5.1.12.76 ST_TextShapeType (Preset Text Shape Types)

This simple type specifies the preset text shape geometry that is to be used for a shape. An enumeration of this type is used so that a custom geometry does not have to be specified but instead can be constructed automatically by the generating application. For each enumeration listed there is also the corresponding DrawingML code that would be used to construct this shape were it a custom geometry. Within the construction code for each of these preset text shapes there are predefined guides that the generating application must maintain for calculation purposes at all times. The necessary guides should have the following values.

3/4 of a Circle ('3cd4') - Constant value of "16200000.0"

The units here are in 60,000ths of a degree. This is equivalent to 270 degrees.

3/8 of a Circle ('3cd8') - Constant value of "8100000.0"

The units here are in 60,000ths of a degree. This is equivalent to 135 degrees.

5/8 of a Circle ('5cd8') - Constant value of "13500000.0"

The units here are in 60,000ths of a degree. This is equivalent to 225 degrees.

7/8 of a Circle ('7cd8') - Constant value of "18900000.0"

The units here are in 60,000ths of a degree. This is equivalent to 315 degrees.

Shape Bottom Edge ('b') - Constant value of "h"

This is the bottom edge of the shape and since the top edge of the shape is considered the 0 point, the bottom edge is thus the shape height.

1/2 of a Circle ('cd2') - Constant value of "10800000.0"

The units here are in 60,000ths of a degree. This is equivalent to 180 degrees.

1/4 of a Circle ('cd4') - Constant value of "5400000.0"

The units here are in 60,000ths of a degree. This is equivalent to 90 degrees.

1/8 of a Circle ('cd8') - Constant value of "2700000.0"

The units here are in 60,000ths of a degree. This is equivalent to 45 degrees.

Shape Height ('h')

This is the variable height of the shape defined in the shape properties. This value is received from the shape transform listed within the <spPr> element.

Horizontal Center ('hc') - Calculated value of "*/ w 1.0 2.0"

This is the horizontal center of the shape which is just the width divided by 2.

1/2 of Shape Height ('hd2') - Calculated value of "*/ h 1.0 2.0"

This is 1/2 the shape height.

1/4 of Shape Height ('hd4') - Calculated value of "*/ h 1.0 4.0"

This is 1/4 the shape height.

1/5 of Shape Height ('hd5') - Calculated value of "*/ h 1.0 5.0"

This is 1/5 the shape height.

1/6 of Shape Height ('hd6') - Calculated value of "*/ h 1.0 6.0"

This is 1/6 the shape height.

1/8 of Shape Height ('hd8') - Calculated value of "*/ h 1.0 8.0"

This is 1/8 the shape height.

Shape Left Edge ('l') - Constant value of "0"

This is the left edge of the shape and the left edge of the shape is considered the horizontal 0 point.

Longest Side of Shape ('ls') - Calculated value of "max w h"

This is the longest side of the shape. This value is either the width or the height depending on which is greater.

Shape Right Edge ('r') - Constant value of "w"

This is the right edge of the shape and since the left edge of the shape is considered the 0 point, the right edge is thus the shape width.

Shortest Side of Shape ('ss') - Calculated value of "min w h"

This is the shortest side of the shape. This value is either the width or the height depending on which is smaller.

1/2 Shortest Side of Shape ('ssd2') - Calculated value of "*/ ss 1.0 2.0"

This is 1/2 the shortest side of the shape.

1/4 Shortest Side of Shape ('ssd4') - Calculated value of "*/ ss 1.0 4.0"

This is 1/4 the shortest side of the shape.

1/6 Shortest Side of Shape ('ssd6') - Calculated value of "*/ ss 1.0 6.0"

This is 1/6 the shortest side of the shape.

1/8 Shortest Side of Shape ('ssd8') - Calculated value of "*/ ss 1.0 8.0"

This is 1/8 the shortest side of the shape.

Shape Top Edge ('t') - Constant value of "0"

This is the top edge of the shape and the top edge of the shape is considered the vertical 0 point.

Vertical Center of Shape ('vc') - Calculated value of "*/ h 1.0 2.0"

This is the vertical center of the shape which is just the height divided by 2.

Shape Width ('w')

This is the variable width of the shape defined in the shape properties. This value is received from the shape transform listed within the <spPr> element.

1/2 of Shape Width ('wd2') - Calculated value of "*/ w 1.0 2.0"

This is 1/2 the shape width.

1/4 of Shape Width ('wd4') - Calculated value of "*/ w 1.0 4.0"

This is 1/4 the shape width.

1/5 of Shape Width ('wd5') - Calculated value of "*/ w 1.0 5.0"

This is 1/5 the shape width.

1/6 of Shape Width ('wd6') - Calculated value of "*/ w 1.0 6.0"

This is 1/6 the shape width.

1/8 of Shape Width ('wd8') - Calculated value of "*/ w 1.0 8.0"

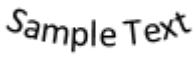
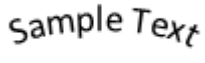
This is 1/8 the shape width.

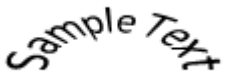
1/10 of Shape Width ('wd10') - Calculated value of "*/ w 1.0 10.0"

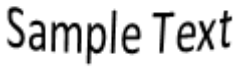
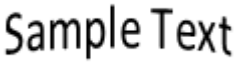
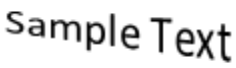
This is 1/10 the shape width.

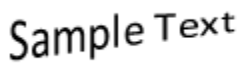
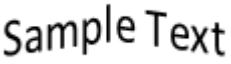
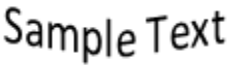
This simple type's contents are a restriction of the XML Schema token datatype.

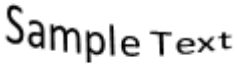
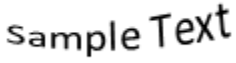
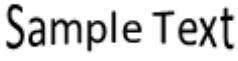
The following are possible enumeration values for this type:

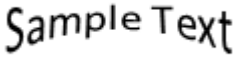
Enumeration Value	Description
textArchDown (Downward Arch Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textArchDown element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textArchDownPour (Downward Pour Arch Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textArchDownPour element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textArchUp (Upward Arch Text Shape)	

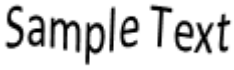
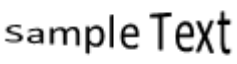
Enumeration Value	Description
	Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textArchUp element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textArchUpPour (Upward Pour Arch Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textArchUpPour element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textButton (Button Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textButton element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textButtonPour (Button Pour Text Shape)	 Specifies a text shape that shall match the normative shape shown above.

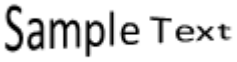
Enumeration Value	Description
	[*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textButtonPour element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textCanDown (Downward Can Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textCanDown element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textCanUp (Upward Can Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textCanUp element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textCascadeDown (Downward Cascade Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textCascadeDown element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]

Enumeration Value	Description
textCascadeUp (Upward Cascade Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textCascadeUp element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textChevron (Chevron Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textChevron element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textChevronInverted (Inverted Chevron Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textChevronInverted element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textCircle (Circle Text Shape)	Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textCircle element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]

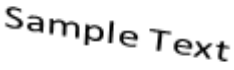
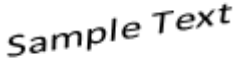
Enumeration Value	Description
textCirclePour (Circle Pour Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textCirclePour element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textCurveDown (Downward Curve Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textCurveDown element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textCurveUp (Upward Curve Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textCurveUp element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textDeflate (Deflate Text Shape)	 Specifies a text shape that shall match the normative

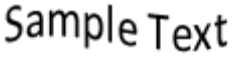
Enumeration Value	Description
	shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textDeflate element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textDeflateBottom (Bottom Deflate Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textDeflateBottom element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textDeflateInflate (Deflate-Inflate Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textDeflateInflate element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textDeflateInflateDeflate (Deflate-Inflate-Deflate Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the

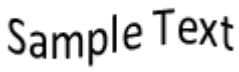
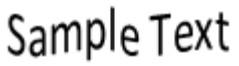
Enumeration Value	Description
	textDeflateInflateDeflate element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textDeflateTop (Top Deflate Text Shape)	Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textDeflateTop element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textDoubleWave1 (Double Wave 1 Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textDoubleWave1 element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textFadeDown (Downward Fade Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textFadeDown element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textFadeLeft (Left Fade Text Shape)	 Specifies a text shape that shall match the normative shape shown above.

Enumeration Value	Description
	[*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textFadeLeft element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textFadeRight (Right Fade Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textFadeRight element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textFadeUp (Upward Fade Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textFadeUp element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textInflate (Inflate Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textInflate element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]

Enumeration Value	Description
textInflateBottom (Bottom Inflate Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textInflateBottom element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textInflateTop (Top Inflate Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textInflateTop element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textNoShape (No Text Shape)	Specifies that the text will have no associated shape with it and thus the text should not be warped but instead be constrained by the normal text bounding box.
textPlain (Plain Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textPlain element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textRingInside (Inside Ring Text Shape)	

Enumeration Value	Description
	Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textRingInside element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textRingOutside (Outside Ring Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textRingOutside element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above
textSlantDown (Downward Slant Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textSlantDown element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textSlantUp (Upward Slant Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textSlantUp element in the preset text warp electronic addenda of Annex D. The constants used in that

Enumeration Value	Description
	markup are guides that are described in further detail above. *end note*]
textStop (Stop Sign Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textStop element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textTriangle (Triangle Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textTriangle element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textTriangleInverted (Inverted Triangle Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textTriangleInverted element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textWave1 (Wave 1 Text Shape)	 Specifies a text shape that shall match the normative shape shown above.

Enumeration Value	Description
	[*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textWave1 element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textWave2 (Wave 2 Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textWave2 element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]
textWave4 (Wave 4 Text Shape)	 Specifies a text shape that shall match the normative shape shown above. [*Note*: An example of DrawingML markup which may be used to achieve this effect is contained in the textWave4 element in the preset text warp electronic addenda of Annex D. The constants used in that markup are guides that are described in further detail above. *end note*]

Referenced By
prstTxWarp@prst (§5.1.11.19)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextShapeType">
   <restriction base="xsd:token">
      <enumeration value="textNoShape"/>
      <enumeration value="textPlain"/>
      <enumeration value="textStop"/>
      <enumeration value="textTriangle"/>
      <enumeration value="textTriangleInverted"/>
      <enumeration value="textChevron"/>
      <enumeration value="textChevronInverted"/>
      <enumeration value="textRingInside"/>
      <enumeration value="textRingOutside"/>
      <enumeration value="textArchUp"/>
      <enumeration value="textArchDown"/>
      <enumeration value="textCircle"/>
      <enumeration value="textButton"/>
      <enumeration value="textArchUpPour"/>
      <enumeration value="textArchDownPour"/>
      <enumeration value="textCirclePour"/>
      <enumeration value="textButtonPour"/>
      <enumeration value="textCurveUp"/>
      <enumeration value="textCurveDown"/>
      <enumeration value="textCanUp"/>
      <enumeration value="textCanDown"/>
      <enumeration value="textWave1"/>
      <enumeration value="textWave2"/>
      <enumeration value="textDoubleWave1"/>
      <enumeration value="textWave4"/>
      <enumeration value="textInflate"/>
      <enumeration value="textDeflate"/>
      <enumeration value="textInflateBottom"/>
      <enumeration value="textDeflateBottom"/>
      <enumeration value="textInflateTop"/>
      <enumeration value="textDeflateTop"/>
      <enumeration value="textDeflateInflate"/>
      <enumeration value="textDeflateInflateDeflate"/>
      <enumeration value="textFadeRight"/>
      <enumeration value="textFadeLeft"/>
      <enumeration value="textFadeUp"/>
      <enumeration value="textFadeDown"/>
      <enumeration value="textSlantUp"/>
      <enumeration value="textSlantDown"/>
      <enumeration value="textCascadeUp"/>
      <enumeration value="textCascadeDown"/>
   </restriction>
</simpleType>
```

5.1.12.77 ST_TextSpacingPercent (Text Spacing Percent)

This type specifies the range that the spacing percent, in terms of a line. Represented in the file format from 0-1000 line hundredths, maps to 0.00-10.00 lines.

This simple type's contents are a restriction of the ST_Percentage simple type (§5.1.12.41).

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 0.
- This simple type has a maximum value of less than or equal to 13200000.

Referenced By
normAutofit@lnSpcReduction (§5.1.5.1.3); spcPct@val (§5.1.5.2.10)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextSpacingPercent">
   <restriction base="ST_Percentage">
      <minInclusive value="0"/>
      <maxInclusive value="13200000"/>
   </restriction>
</simpleType>
```

5.1.12.78 ST_TextSpacingPoint (Text Spacing Point)

This type specifies the Text Spacing that will be used in terms of font point size.

This simple type's contents are a restriction of the XML Schema int datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 0.
- This simple type has a maximum value of less than or equal to 158400.

Referenced By
spcPts@val (§5.1.5.2.11)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextSpacingPoint">
   <restriction base="xsd:int">
      <minInclusive value="0"/>
      <maxInclusive value="158400"/>
   </restriction>
</simpleType>
```

5.1.12.79 ST_TextStrikeType (Text Strike Type)

This type specifies the strike type.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description

Enumeration Value	Description
dblStrike (Text Strike Enum (Double Strike))	A double strikethrough applied on the text
noStrike (Text Strike Enum (No Strike))	No strike is applied to the text
sngStrike (Text Strike Enum (Single Strike))	A single strikethrough is applied to the text

Referenced By
defRPr@strike (§5.1.5.3.2); endParaRPr@strike (§5.1.5.2.3); rPr@strike (§5.1.5.3.9)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextStrikeType">
   <restriction base="xsd:token">
      <enumeration value="noStrike"/>
      <enumeration value="sngStrike"/>
      <enumeration value="dblStrike"/>
   </restriction>
</simpleType>
```

5.1.12.80 ST_TextTabAlignType (Text Tab Alignment Types)

This type specifies the text tab alignment types.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
ctr (Text Tab Alignment Enum (Center))	The text at this tab stop is center aligned.
dec (Text Tab Alignment Enum (Decimal))	At this tab stop, the decimals are lined up. From a user's point of view, the text here behaves as right aligned until the decimal, and then as left aligned after the decimal.
l (Text Tab Alignment Enum (Left))	The text at this tab stop is left aligned.
r (Text Tab Alignment Enum (Right))	The text at this tab stop is right aligned.

Referenced By
tab@algn (§5.1.5.2.12)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextTabAlignType">
   <restriction base="xsd:token">
      <enumeration value="l"/>
      <enumeration value="ctr"/>
      <enumeration value="r"/>
      <enumeration value="dec"/>
   </restriction>
</simpleType>
```

5.1.12.81 ST_TextTypeface (Text Typeface)

This type specifies the way we represent a font typeface.

This simple type's contents are a restriction of the XML Schema string datatype.

Referenced By
buFont@typeface (§5.1.5.4.6); cs@typeface (§5.1.5.3.1); ea@typeface (§5.1.5.3.3); font@typeface (§4.3.1.10); font@typeface (§5.1.4.1.16); latin@typeface (§5.1.5.3.7); sym@typeface (§5.1.5.3.10)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextTypeface">
   <restriction base="xsd:string"/>
</simpleType>
```

5.1.12.82 ST_TextUnderlineType (Text Underline Types)

This type specifies the text underline types that will be used.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
dash (Text Underline Enum (Dashed))	Underline the text with a single, dashed line of normal thickness.
dashHeavy (Text Underline Enum (Heavy Dashed))	Underline the text with a single, dashed, thick line.
dashLong (Text Underline Enum (Long Dashed))	Underline the text with a single line consisting of long dashes of normal thickness.
dashLongHeavy (Text Underline Enum (Heavy Long Dashed))	Underline the text with a single line consisting of long, thick dashes.
dbl (Text Underline Enum (Double))	Underline the text with two lines of normal thickness.
dotDash (Text Underline Enum (Dot Dash))	Underline the text with a single line of normal thickness consisting of repeating dots and dashes.
dotDashHeavy (Text Underline Enum (Heavy Dot Dash))	Underline the text with a single, thick line consisting of repeating dots and dashes.

Enumeration Value	Description
dotDotDash (Text Underline Enum (Dot Dot Dash))	Underline the text with a single line of normal thickness consisting of repeating two dots and dashes.
dotDotDashHeavy (Text Underline Enum (Heavy Dot Dot Dash))	Underline the text with a single, thick line consisting of repeating two dots and dashes.
dotted (Text Underline Enum (Dotted))	Underline the text with a single, dotted line of normal thickness.
dottedHeavy (Text Underline Enum (Heavy Dotted))	Underline the text with a single, thick, dotted line.
heavy (Text Underline Enum (Heavy))	Underline the text with a single, thick line.
none (Text Underline Enum (None))	The reason we cannot implicitly have noUnderline be the scenario where underline is not specified is because not being specified implies deriving from a particular style and the user might want to override that and make some text not be underlined even though the style says otherwise.
sng (Text Underline Enum (Single))	Underline the text with a single line of normal thickness.
wavy (Text Underline Enum (Wavy))	Underline the text with a single wavy line of normal thickness.
wavyDbl (Text Underline Enum (Double Wavy))	Underline the text with two wavy lines of normal thickness.
wavyHeavy (Text Underline Enum (Heavy Wavy))	Underline the text with a single, thick wavy line.
words (Text Underline Enum (Words))	Underline just the words and not the spaces between them.

Referenced By
defRPr@u (§5.1.5.3.2); endParaRPr@u (§5.1.5.2.3); rPr@u (§5.1.5.3.9)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextUnderlineType">
   <restriction base="xsd:token">
      <enumeration value="none"/>
      <enumeration value="words"/>
      <enumeration value="sng"/>
      <enumeration value="dbl"/>
      <enumeration value="heavy"/>
      <enumeration value="dotted"/>
      <enumeration value="dottedHeavy"/>
      <enumeration value="dash"/>
      <enumeration value="dashHeavy"/>
      <enumeration value="dashLong"/>
      <enumeration value="dashLongHeavy"/>
      <enumeration value="dotDash"/>
      <enumeration value="dotDashHeavy"/>
      <enumeration value="dotDotDash"/>
      <enumeration value="dotDotDashHeavy"/>
      <enumeration value="wavy"/>
      <enumeration value="wavyHeavy"/>
      <enumeration value="wavyDbl"/>
   </restriction>
</simpleType>
```

5.1.12.83 ST_TextVerticalType (Vertical Text Types)

If there is vertical text, determines what type of vertical text is going to be used.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
eaVert (Vertical Text Type Enum (East Asian Vertical))	A special version of vertical text, where some fonts are displayed as if rotated by 90 degrees while some fonts (mostly East Asian) are displayed vertical.
horz (Vertical Text Type Enum (Horizontal))	Horizontal text. This should be default.
mongolianVert (Vertical Text Type Enum (Mongolian Vertical))	A special version of vertical text, where some fonts are displayed as if rotated by 90 degrees while some fonts (mostly East Asian) are displayed vertical. The difference between this and the eastAsianVertical is the text flows top down then LEFT RIGHT, instead of RIGHT LEFT
vert (Vertical Text Type Enum (Vertical))	Determines if all of the text is vertical orientation (each line is 90 degrees rotated clockwise, so it goes from top to bottom; each next line is to the left from the previous one).
vert270 (Vertical Text Type Enum (Vertical 270))	Determines if all of the text is vertical orientation

Enumeration Value	Description
	(each line is 270 degrees rotated clockwise, so it goes from bottom to top; each next line is to the right from the previous one).
wordArtVert (Vertical Text Type Enum (WordArt Vertical))	Determines if all of the text is vertical ("one letter on top of another").
wordArtVertRtl (Vertical WordArt Right to Left)	Specifies that vertical WordArt should be shown from right to left rather than left to right.

Referenced By
bodyPr@vert (§5.1.5.1.1); tcPr@vert (§5.1.6.15)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextVerticalType">
   <restriction base="xsd:token">
      <enumeration value="horz"/>
      <enumeration value="vert"/>
      <enumeration value="vert270"/>
      <enumeration value="wordArtVert"/>
      <enumeration value="eaVert"/>
      <enumeration value="mongolianVert"/>
      <enumeration value="wordArtVertRtl"/>
   </restriction>
</simpleType>
```

5.1.12.84 ST_TextVertOverflowType (Text Vertical Overflow)

This type specifies the text vertical overflow.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
clip (Text Overflow Enum (Clip))	Pay attention to top and bottom barriers. Provide no indication that there is text which is not visible.
ellipsis (Text Overflow Enum (Ellipsis))	Pay attention to top and bottom barriers. Use an ellipsis to denote that there is text which is not visible.
overflow (Text Overflow Enum (Overflow))	Overflow the text and pay no attention to top and bottom barriers.

Referenced By
bodyPr@vertOverflow (§5.1.5.1.1)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextVertOverflowType">
   <restriction base="xsd:token">
      <enumeration value="overflow"/>
      <enumeration value="ellipsis"/>
      <enumeration value="clip"/>
   </restriction>
</simpleType>
```

5.1.12.85 ST_TextWrappingType (Text Wrapping Types)

Text Wrapping Types

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
none (Text Wrapping Type Enum (None))	No wrapping will occur on this text body. Words will spill out without paying attention to the bounding rectangle boundaries.
square (Text Wrapping Type Enum (Square))	Determines whether we wrap words within the bounding rectangle.

Referenced By
bodyPr@wrap (§5.1.5.1.1)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextWrappingType">
   <restriction base="xsd:token">
      <enumeration value="none"/>
      <enumeration value="square"/>
   </restriction>
</simpleType>
```

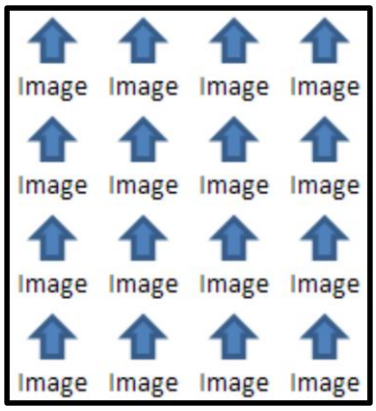
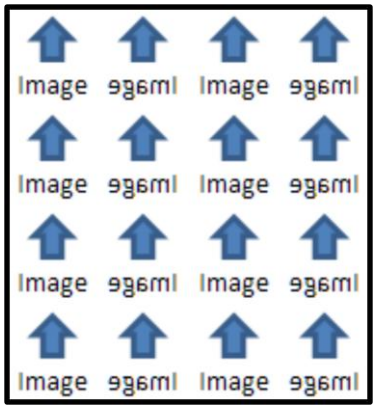
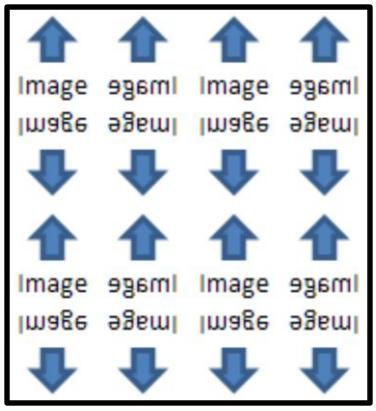
5.1.12.86 ST_TileFlipMode (Tile Flip Mode)

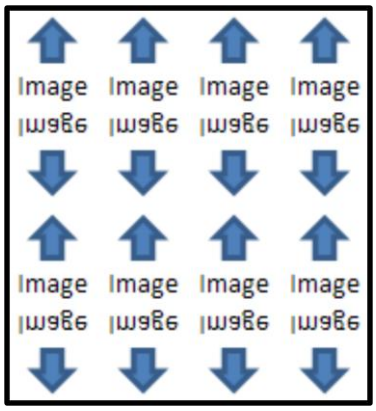
This simple type indicates whether/how to flip the contents of a tile region when using it to fill a larger fill region.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description

Enumeration Value	Description
none (None)	 Tiles are not flipped.
x (Horizontal)	 Tiles are flipped horizontally.
xy (Horizontal and Vertical)	 Tiles are flipped both horizontally and vertically.

Enumeration Value	Description
y (Vertical)	 Tiles are flipped vertically.

Referenced By
gradFill@flip (§5.1.10.33); tile@flip (§5.1.10.58)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TileFlipMode">
   <restriction base="xsd:token">
      <enumeration value="none"/>
      <enumeration value="x"/>
      <enumeration value="y"/>
      <enumeration value="xy"/>
   </restriction>
</simpleType>
```

5.2 DrawingML - Picture

These elements encompass the definition of pictures within the DrawingML framework. While pictures are in many ways very similar to shapes they have specific properties that are unique in order to optimize for picture-specific scenarios. Some of these properties include Fill behavior, Border behavior and Resize behavior.

5.2.1 Table of Contents

This subclause is informative.

End of informative text.

5.2.2 Elements

The following section defines the Picture portion of the DrawingML framework.

5.2.2.1 blipFill (Picture Fill)

This element specifies the type of picture fill that the picture object will have. Because a picture has a picture fill already by default, it is possible to have two fills specified for a picture object. An example of this is shown below.

[*Example*: Consider the picture below that has a blip fill applied to it. The image used to fill this picture object has transparent pixels instead of white pixels.

```
<p:pic>
  ..
  <p:blipFill>
    <a:blip r:embed="rId2"/>
    <a:stretch>
      <a:fillRect/>
    </a:stretch>
  </p:blipFill>
  ..
</p:pic>
```

The above picture object is shown as an example of this fill type. *end example*]

[*Example*: Consider now the same picture object but with an additional gradient fill applied within the shape properties portion of the picture.

```
<p:pic>
  ..
  <p:blipFill>
    <a:blip r:embed="rId2"/>
    <a:stretch>
      <a:fillRect/>
    </a:stretch>
  </p:blipFill>
  <p:spPr>
    <a:gradFill>
      <a:gsLst>
        <a:gs pos="0">
          <a:schemeClr val="tx2">
            <a:shade val="50000"/>
          </a:schemeClr>
        </a:gs>
        <a:gs pos="39999">
          <a:schemeClr val="tx2">
            <a:tint val="20000"/>
          </a:schemeClr>
        </a:gs>
        <a:gs pos="70000">
          <a:srgbClr val="C4D6EB"/>
        </a:gs>
        <a:gs pos="100000">
          <a:schemeClr val="bg1"/>
        </a:gs>
      </a:gsLst>
    </a:gradFill>
  </p:spPr>
  ..
</p:pic>
```

The above picture object is shown as an example of this double fill type. *end example*]

Parent Elements
pic (§5.2.2.5)

Child Elements	Subclause
blip (Blip)	§5.1.10.13
srcRect (Source Rectangle)	§5.1.10.55
stretch (Stretch)	§5.1.10.56
tile (Tile)	§5.1.10.58

Attributes	Description
dpi (DPI Setting) Namespace: .../drawingml/2006/main	Specifies the DPI (dots per inch) used to calculate the size of the blip. If not present or zero, the DPI in the blip is used. [*Note*: This attribute is primarily used to keep track of the picture quality within a document. There are different levels of quality needed for print than on-screen viewing and thus a need to track this information. *end note*] The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
rotWithShape (Rotate With Shape) Namespace: .../drawingml/2006/main	Specifies that the fill should rotate with the shape. That is, when the shape that has been filled with a picture and the containing shape (say a rectangle) is transformed with a rotation then the fill will be transformed with the same rotation. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BlipFillProperties">
   <sequence>
      <element name="blip" type="CT_Blip" minOccurs="0" maxOccurs="1"/>
      <element name="srcRect" type="CT_RelativeRect" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_FillModeProperties" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="dpi" type="xsd:unsignedInt" use="optional"/>
   <attribute name="rotWithShape" type="xsd:boolean" use="optional"/>
</complexType>
```

5.2.2.2 cNvPicPr (Non-Visual Picture Drawing Properties)

This element specifies the non-visual properties for the picture canvas. These properties are to be used by the generating application to determine how certain properties are to be changed for the picture object in question.

[*Example*: Consider the following DrawingML.

```
<p:pic>
  ..
  <p:nvPicPr>
    <p:cNvPr id="4" name="Lilly.jpg"/>
    <p:cNvPicPr>
      <a:picLocks noChangeAspect="1"/>
    </p:cNvPicPr>
    <p:nvPr/>
  </p:nvPicPr>
  ..
</p:pic>
```

end example]

Parent Elements
nvPicPr (§5.2.2.4)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
picLocks (Picture Locks)	§5.1.2.1.31

Attributes	Description
preferRelativeResize (Relative Resize Preferred)	Specifies if the user interface should show the resizing of the picture based on the picture's current size or its original size. If this attribute is set to true, then scaling will be relative to the original picture size as opposed to the current picture size.

Attributes	Description
Namespace: .../drawingml/2006/main	[*Example*: Consider the case where a picture has been resized within a document and is now 50% of the originally inserted picture size. Now if the user chooses to make a later adjustment to the size of this picture within the generating application, then the value of this attribute should be checked. If this attribute is set to true then a value of 50% will be shown. Similarly, if this attribute is set to false, then a value of 100% should be shown because the picture has not yet been resized from its current (smaller) size. *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualPictureProperties">
   <sequence>
      <element name="picLocks" type="CT_PictureLocking" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="preferRelativeResize" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.2.2.3 cNvPr (Non-Visual Drawing Properties)

This element specifies non-visual canvas properties. This allows for additional information that does not affect the appearance of the picture to be stored.

[*Example*: Consider the following DrawingML.

```
<p:pic>
  ..
  <p:nvPicPr>
    <p:cNvPr id="4" name="Lilly.jpg"/>
  </p:nvPicPr>
  ..
</p:pic>
```

end example]

Parent Elements
nvPicPr (§5.2.2.4)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
hlinkClick (Click Hyperlink)	§5.1.5.3.5
hlinkHover (Hyperlink for Hover)	§5.1.2.1.23

Attributes	Description
descr (Alternative Text for Object) Namespace: .../drawingml/2006/main	Specifies alternative text for the current DrawingML object, for use by assistive technologies or applications which will not display the current object. If this element is omitted, then no alternative text is present for the parent object. [*Example*: Consider a DrawingML object defined as follows: `<… descr="A picture of a bowl of fruit">` The descr attribute contains alternative text which may be used in place of the actual DrawingML object. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
hidden (Hidden) Namespace: .../drawingml/2006/main	Specifies whether this DrawingML object shall be displayed. When a DrawingML object is displayed within a document, that object may be hidden (i.e., present, but not visible). This attribute shall determine whether the object shall be rendered or made hidden. [*Note*: An application may have settings which allow this object to be viewed. *end note*] If this attribute is omitted, then the parent DrawingML object shall be displayed (i.e., not hidden). [*Example*: Consider an inline DrawingML object which shall be hidden within the document's content. This setting would be specified as follows: `<… hidden="true" />` The hidden attribute has a value of `true`, which specifies that the DrawingML object is hidden and not displayed when the document is displayed. *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
id (Unique Identifier) Namespace: .../drawingml/2006/main	Specifies a unique identifier for the current DrawingML object within the current document. This ID may be used to assist in uniquely identifying this object so that it can be referred to by other parts of the document. If multiple objects within the same document share the same id attribute value, then the document shall be considered non-conformant. [*Example*: Consider a DrawingML object defined as follows: `<… id="10" … >` The id attribute has a value of `10`, which is the unique identifier for this DrawingML object. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_DrawingElementId simple type (§5.1.12.19).
name (Name) Namespace: .../drawingml/2006/main	Specifies the name of the object. [*Note*: Typically, this will be used to store the original file name of a picture object. *end note*] [*Example*: Consider a DrawingML object defined as follows: `< … name="foo.jpg" >` The name attribute has a value of `foo.jpg`, which is the name of this DrawingML object. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualDrawingProps">
   <sequence>
      <element name="hlinkClick" type="CT_Hyperlink" minOccurs="0" maxOccurs="1"/>
      <element name="hlinkHover" type="CT_Hyperlink" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="id" type="ST_DrawingElementId" use="required"/>
   <attribute name="name" type="xsd:string" use="required"/>
   <attribute name="descr" type="xsd:string" use="optional" default=""/>
   <attribute name="hidden" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.2.2.4 nvPicPr (Non-Visual Picture Properties)

This element specifies the non visual properties for a picture. This allows for additional information that does not affect the appearance of the picture to be stored.

[*Example*: Consider the following DrawingML.

```
<p:pic>
  ..
  <p:nvPicPr>
    ..
  </p:nvPicPr>
  ..
</p:pic>
```

end example]

Parent Elements
pic (§5.2.2.5)

Child Elements	Subclause
cNvPicPr (Non-Visual Picture Drawing Properties)	§5.2.2.2
cNvPr (Non-Visual Drawing Properties)	§5.2.2.3

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PictureNonVisual">
   <sequence>
      <element name="cNvPr" type="a:CT_NonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
      <element name="cNvPicPr" type="a:CT_NonVisualPictureProperties" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.2.2.5 pic (Picture)

This element specifies the existence of a picture object within the document.

[*Example*: Consider the following DrawingML that specifies the existence of a picture within a document. This picture can have non-visual properties, a picture fill as well as shape properties attached to it.

```
<p:pic>
  <p:nvPicPr>
    <p:cNvPr id="4" name="lake.JPG" descr="Picture of a Lake" />
    <p:cNvPicPr>
      <a:picLocks noChangeAspect="1"/>
    </p:cNvPicPr>
    <p:nvPr/>
  </p:nvPicPr>
  <p:blipFill>
  ...
  </p:blipFill>
  <p:spPr>
  ...
  </p:spPr>
</p:pic>
```

end example]

Child Elements	Subclause
blipFill (Picture Fill)	§5.2.2.1
nvPicPr (Non-Visual Picture Properties)	§5.2.2.4
spPr (Shape Properties)	§5.2.2.6

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Picture">
   <sequence minOccurs="1" maxOccurs="1">
      <element name="nvPicPr" type="CT_PictureNonVisual" minOccurs="1" maxOccurs="1"/>
      <element name="blipFill" type="a:CT_BlipFillProperties" minOccurs="1" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.2.2.6 spPr (Shape Properties)

This element specifies the visual shape properties that can be applied to a picture. These are the same properties that are allowed to describe the visual properties of a shape but are used here to describe the visual appearance of a picture within a document. This allows for a picture to have both the properties of a shape as well as picture specific properties that are allowed under the pic element.

Parent Elements
pic (§5.2.2.5)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
custGeom (Custom Geometry)	§5.1.11.8
effectDag (Effect Container)	§5.1.10.25
effectLst (Effect Container)	§5.1.10.26
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35
ln (Outline)	§5.1.2.1.24
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
prstGeom (Preset geometry)	§5.1.11.18
scene3d (3D Scene Properties)	§5.1.4.1.26
solidFill (Solid Fill)	§5.1.10.54
sp3d (Apply 3D shape properties)	§5.1.7.12
xfrm (2D Transform for Individual Objects)	§5.1.9.6

Attributes	Description
bwMode (Black and White Mode)	Specifies that the picture should be rendered using only black and white coloring. That is the coloring information for the picture should be converted to either black or white

Attributes	Description
Namespace: .../drawingml/2006/main	when rendering the picture. No gray is to be used in rendering this image, only stark black and stark white. [*Note*: This does not mean that the picture itself that is stored within the file is necessarily a black and white picture. This attribute instead sets the rendering mode that the picture will have applied to when rendering. *end note*] The possible values for this attribute are defined by the ST_BlackWhiteMode simple type (§5.1.12.10).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShapeProperties">
   <sequence>
      <element name="xfrm" type="CT_Transform2D" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_Geometry" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_FillProperties" minOccurs="0" maxOccurs="1"/>
      <element name="ln" type="CT_LineProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_EffectProperties" minOccurs="0" maxOccurs="1"/>
      <element name="scene3d" type="CT_Scene3D" minOccurs="0" maxOccurs="1"/>
      <element name="sp3d" type="CT_Shape3D" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="bwMode" type="ST_BlackWhiteMode" use="optional"/>
</complexType>
```

5.3 DrawingML - Legacy Compatibility

Within the context of DrawingML, it must be possible (for considerations to legacy compatibility) to be able to include explicit references to specific shapes within VML Drawing parts.

[*Example*: A VML Drawing part is used to define ink on a PresentationML slide, but the resulting ink is referenced from the slide by its shape ID using the elements of this namespace. *end example*]

5.3.1 Table of Contents

This subclause is informative.

End of informative text.

5.3.2 Basics

Legacy Compatibility is part of the shape definitions and properties of the DrawingML framework.

5.3.2.1 legacyDrawing (Legacy Drawing Object)

This element specifies the shape ID for a legacy drawing object. These legacy drawing objects all have a shape ID associated with them that is unique across the entire document. In order to store these legacy shape IDs as well as new shape IDs this legacyDrawing element should be used.

Attributes	Description
spid (Shape ID)	Legacy Shape ID that is unique throughout the entire document. Legacy shape IDs should be assigned based on which portion of the document the drawing resides on. The assignment of these ids is broken down into clusters of 1024 values. The first cluster is 1-1024, the second 1025-2048 and so on. [*Example*: Within a word processing application the spid should be assigned based on the page that the drawing resides on. If the drawing resides on the second page then the assigned spid should be a value between 1025 and 2048. *end example*] [*Example*: Within a spreadsheet application the spid should be assigned based on the sheet that the drawing resides on. If the drawing resides on the second sheet then the assigned spid should be a value between 1025 and 2048. *end example*] [*Example*: Within a presentation application the spid should be assigned based on the slide that the drawing resides on. If the drawing resides on the second slide then the assigned spid should be a value between 1025 and 2048. *end example*] The possible values for this attribute are defined by the ST_ShapeID simple type (§5.1.12.55).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Compat">
   <attribute name="spid" type="a:ST_ShapeID" use="required"/>
</complexType>
```

5.4 DrawingML - Locked Canvas

Within a DrawingML object, a *locked canvas* allows DrawingML objects to be placed in a format where they can be viewed but not edited by the hosting application. This allows DrawingML objects not supported by an application to be included and viewed in applications where they cannot be edited.

5.4.1 Table of Contents

This subclause is informative.

End of informative text.

5.4.2 Basics

This section specifies a locked canvas within the basic DrawingML framework.

5.4.2.1 lockedCanvas (Locked Canvas Container)

The locked canvas element acts as a container for more advanced drawing objects. The notion of a locked canvas comes from the fact that the generating application opening the file cannot create this object and can thus not perform edits either. Thus the drawing object is locked from all UI adjustments that would normally take place.

Child Elements	Subclause
cxnSp (Connection Shape)	§5.1.2.1.10
extLst (Extension List)	§5.1.2.1.15
graphicFrame (Graphic Frame)	§5.1.2.1.18
grpSp (Group shape)	§5.1.2.1.20
grpSpPr (Visual Group Shape Properties)	§5.1.2.1.22
nvGrpSpPr (Non-Visual Properties for a Group Shape)	§5.1.2.1.27
pic (Picture)	§5.1.2.1.30
sp (Shape)	§5.1.2.1.33
txSp (Text Shape)	§5.1.2.1.41

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GvmlGroupShape">
   <sequence>
      <element name="nvGrpSpPr" type="CT_GvmlGroupShapeNonVisual" minOccurs="1" maxOccurs="1"/>
      <element name="grpSpPr" type="CT_GroupShapeProperties" minOccurs="1" maxOccurs="1"/>
      <choice minOccurs="0" maxOccurs="unbounded">
         <element name="txSp" type="CT_GvmlTextShape"/>
         <element name="sp" type="CT_GvmlShape"/>
         <element name="cxnSp" type="CT_GvmlConnector"/>
         <element name="pic" type="CT_GvmlPicture"/>
         <element name="graphicFrame" type="CT_GvmlGraphicalObjectFrame"/>
         <element name="grpSp" type="CT_GvmlGroupShape"/>
      </choice>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.5 DrawingML - WordprocessingML Drawing

Within a WordprocessingML document, it is possible to include graphical DrawingML objects:

- Pictures (§5.2)
- Locked Canvases (§5.4)
- Diagrams (§5.9)

- Charts (§5.7)

When these objects are present in a word processing document, it is necessary to include information which specifies how the objects shall be positioned relative to the paginated document. [*Example*: Whether the object is displayed in line with text. *end example*]

The WordprocessingML Drawing namespace acts in this capacity, specifying all information necessary to anchor and display DrawingML objects within a word processing document.

[*Example*: Consider a DrawingML picture which shall be displayed in the center of the printed page on which it appears, modifying the flow of text as necessary. This object would be specified as follows:

```
<w:r>
  <w:drawing>
    <wp:anchor relativeHeight="10" allowOverlap="true">
      <wp:positionH relativeFrom="margin">
        <wp:align>center</wp:align>
      </wp:positionH>
      <wp:positionV relativeFrom="margin">
        <wp:align>center</wp:align>
      </wp:positionV>
      <wp:extent cx="2441542" cy="1828800"/>
      <wp:wrapSquare wrapText="bothSides"/>
      <a:graphic>
        …
      </a:graphic>
    </wp:anchor>
  </w:drawing>
</w:r>
```

The anchor element (§5.5.2.3) specifies that this object is not positioned in line with text, and its child elements specify that the object is centered on the page horizontally and vertically (§5.5.2.10; §5.5.2.11), and that text can wrap around it in a square (§5.5.2.17). *end example*]

5.5.1 Table of Contents

This subclause is informative.

End of informative text.

5.5.2 Elements

The following elements define the contents of the WordprocessingML Drawing namespace:

5.5.2.1 align (Relative Horizontal Alignment)

This element specifies how a DrawingML object shall be horizontally aligned relative to the horizontal alignment base defined by the parent element. Once an alignment base is defined, this element shall determine how the DrawingML object shall be aligned relative to that location.

[*Example*: Consider a picture in a WordprocessingML document which has been aligned relative to the edge of the page - the left of the page horizontally, and the top of the page vertically. This alignment would be specified as follows:

```
<wp:anchor … >
  <wp:positionH relativeFrom="page">
    <wp:align>left</wp:align>
  </wp:positionH>
  …
</wp:anchor>
```

The align element with a value of `left` specifies that for the horizontal positioning defined by the parent element (in this case, positioning relative to the page), the picture shall be aligned to the left edge of the page. *end example*]

The possible values for this element are defined by the ST_AlignH simple type (§5.5.3.1).

Parent Elements
positionH (§5.5.2.10)

5.5.2.2 align (Relative Vertical Alignment)

This element specifies how a DrawingML object shall be vertically aligned relative to the vertical alignment base defined by the parent element. Once an alignment base is defined, this element shall determine how the DrawingML object shall be aligned relative to that location.

[*Example*: Consider a picture in a WordprocessingML document which has been aligned relative to the edge of the page - the left of the page horizontally, and the top of the page vertically. This alignment would be specified as follows:

```
<wp:anchor … >
  <wp:positionV relativeFrom="page">
    <wp:align>top</wp:align>
  </wp:positionH>
  …
</wp:anchor>
```

The align element with a value of `top` specifies that for the vertical positioning defined by the parent element (in this case, positioning relative to the page), the picture shall be aligned to the top edge of the page. *end example*]

The possible values for this element are defined by the ST_AlignV simple type (§5.5.3.2).

Parent Elements
positionV (§5.5.2.11)

5.5.2.3 anchor (Anchor for Floating DrawingML Object)

This element specifies that the DrawingML object located at this position in the document is a floating object. Within a WordprocessingML document, drawing objects can exist in two states:

- *Inline* - The drawing object is in line with the text, and affects the line height and layout of its line (like a character glyph of similar size).
- *Floating* - The drawing object is anchored within the text, but may be absolutely positioned in the document relative to the page.

When this element encapsulates the DrawingML object's information, then all child elements shall dictate the positioning of this object as a floating object on the page.

[*Example*: Consider a WordprocessingML document where the anchor for a floating DrawingML object shall be the first piece of run content within a paragraph. That paragraph's content would be specified as follows:

```
<w:p>
  <w:r>
    <w:drawing>
      <wp:anchor … >
        …
      </wp:anchor>
    </w:drawing>
  </w:r>
</w:p>
```

The anchor element, when present as the child element of the drawing element, specifies that this DrawingML object shall be positioned as a floating object based on the values of its child elements. *end example*]

Parent Elements
drawing (§2.3.3.9)

Child Elements	Subclause
cNvGraphicFramePr (Common DrawingML Non-Visual Properties)	§5.5.2.4
docPr (Drawing Object Non-Visual Properties)	§5.5.2.5
effectExtent (Object Extents Including Effects)	§5.5.2.6
extent (Drawing Object Size)	§5.5.2.7
graphic (Graphic Object)	§5.1.2.1.16
positionH (Horizontal Positioning)	§5.5.2.10
positionV (Vertical Positioning)	§5.5.2.11
simplePos (Simple Positioning Coordinates)	§5.5.2.13
wrapNone (No Text Wrapping)	§5.5.2.15
wrapSquare (Square Wrapping)	§5.5.2.17
wrapThrough (Through Wrapping)	§5.5.2.18
wrapTight (Tight Wrapping)	§5.5.2.19
wrapTopAndBottom (Top and Bottom Wrapping)	§5.5.2.20

Attributes	Description
allowOverlap	Specifies whether a DrawingML object which intersects another DrawingML object at

Attributes	Description

(Allow Objects to Overlap)	display time shall be allowed to overlap the contents of the other DrawingML object. If a DrawingML object cannot overlap other DrawingML object, it shall be repositioned when displayed to prevent this overlap as needed.
	
	If this element is omitted on a given DrawingML object, then overlap shall not be allowed between a DrawingML object which intersects another DrawingML object displayed at the same location.
	
	[*Example*: Consider a document with two DrawingML objects which are allowed to overlap each other. This would be specified as follows within each object's anchor markup:
	
	`<wp:anchor allowOverlap="true" … >`
	`  …`
	`</wp:anchor>`
	
	The allowOverlap attribute has a value of `true`, which specifies that this object shall be allowed to overlap other objects when it is displayed on the document. *end example*]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

behindDoc (Display Behind Document Text)	Specifies whether this floating DrawingML object shall be displayed behind the text of the document when the document is displayed. When a DrawingML object is displayed within a WordprocessingML document, that object may intersect with text in the document. This attribute shall determine whether the text or the object shall be rendered on top in case of overlapping.
	
	If this attribute is omitted, then the parent DrawingML object shall be displayed in front of the text content of the document in cases of overlapping.
	
	[*Example*: Consider a floating DrawingML object which shall be displayed above any text which it intersects within the document's content. This setting would be specified as follows:
	
	`<wp:anchor behindDoc="false" … >`
	`  …`
	`</wp:anchor>`
	
	The behindDoc attribute has a value of `false`, which specifies that the DrawingML object is displayed above the text of the document in z-order. *end example*]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

distB (Distance From Text on Bottom Edge)	Specifies the minimum distance which shall be maintained between the bottom edge of this drawing object and any subsequent text within the document when this graphical object is displayed within the document's contents. The distance shall be measured in EMUs (English Metric Units).

Attributes	Description

	If this object is an inline object (i.e. has a parent element of inline), then this value shall not have any effect when displaying the object in line with text, but may be maintained and used if the object is subsequently changed to floating. If the wrapping element [*Example*: wrapThrough or wrapSquare *end example*] present as a child element also has a distance from text, then this value shall be ignored.
	
	[*Example*: Consider a floating DrawingML object which shall have one-half of an inch of padding between its bottom edge and the nearest text. This setting would be specified as follows:
	
	`<wp:anchor distB="457200" … >`
	`  …`
	`</wp:anchor>`
	
	The distB attribute specifies that the padding distance shall be 457200 EMUs or one-half of an inch. *end example*]
	
	The possible values for this attribute are defined by the ST_WrapDistance simple type (§5.5.3.6).

distL (Distance From Text on Left Edge)	Specifies the minimum distance which shall be maintained between the left edge of this drawing object and any subsequent text within the document when this graphical object is displayed within the document's contents.
	
	The distance shall be measured in EMUs (English Metric Units).
	
	If this object is an inline object (i.e. has a parent element of inline), then this value shall not have any effect when displaying the object in line with text, but may be maintained and used if the object is subsequently changed to floating. If the wrapping element [*Example*: wrapThrough or wrapSquare *end example*] present as a child element also has a distance from text, then this value shall be ignored.
	
	[*Example*: Consider a floating DrawingML object which shall have one-quarter of an inch of padding between its left edge and the nearest text. This setting would be specified as follows:
	
	`<wp:anchor distL="228600" … >`
	`  …`
	`</wp:anchor>`
	
	The distL attribute specifies that the padding distance shall be 228600 EMUs or one-quarter of an inch. *end example*]
	
	The possible values for this attribute are defined by the ST_WrapDistance simple type (§5.5.3.6).

distR (Distance	Specifies the minimum distance which shall be maintained between the right edge of this

Attributes	Description

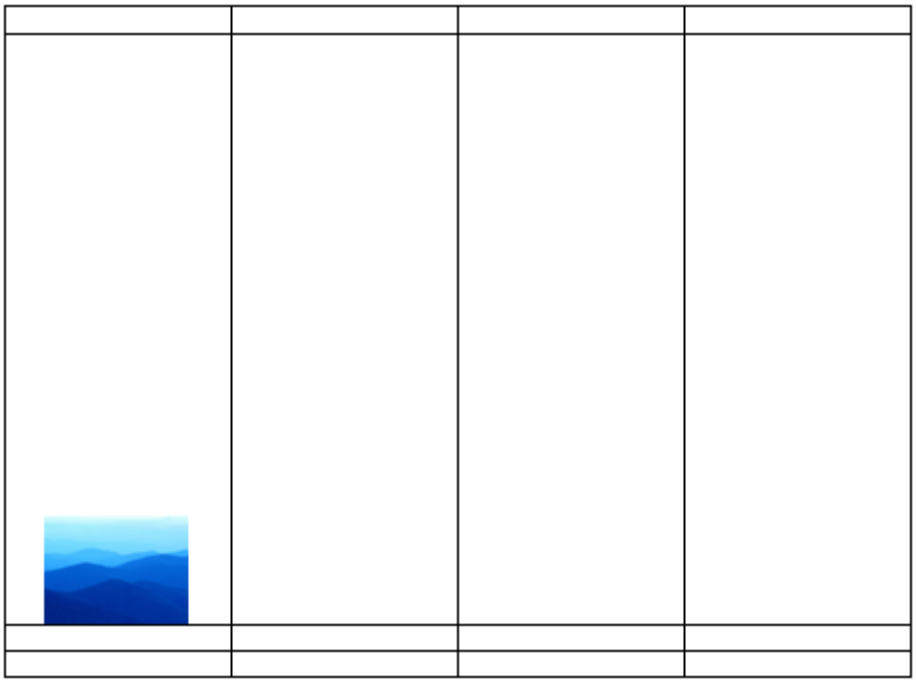
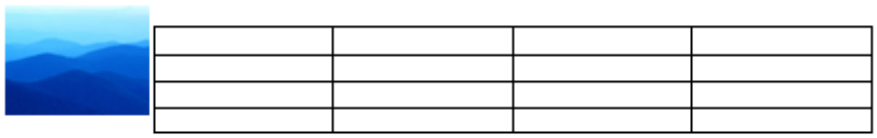
From Text on Right Edge)	drawing object and any subsequent text within the document when this graphical object is displayed within the document's contents.
	
	The distance shall be measured in EMUs (English Metric Units).
	
	If this object is an inline object (i.e. has a parent element of inline), then this value shall not have any effect when displaying the object in line with text, but may be maintained and used if the object is subsequently changed to floating. If the wrapping element [*Example*: wrapThrough or wrapSquare *end example*] present as a child element also has a distance from text, then this value shall be ignored.
	
	[*Example*: Consider a floating DrawingML object which shall have one-quarter of an inch of padding between its right edge and the nearest text. This setting would be specified as follows:
	
	`<wp:anchor distR="228600" … >`
	`  …`
	`</wp:anchor>`
	
	The distR attribute specifies that the padding distance shall be 228600 EMUs or one-quarter of an inch. *end example*]
	
	The possible values for this attribute are defined by the ST_WrapDistance simple type (§5.5.3.6).

distT (Distance From Text on Top Edge)	Specifies the minimum distance which shall be maintained between the top edge of this drawing object and any subsequent text within the document when this graphical object is displayed within the document's contents.
	
	The distance shall be measured in EMUs (English Metric Units).
	
	If this object is an inline object (i.e. has a parent element of inline), then this value shall not have any effect when displaying the object in line with text, but may be maintained and used if the object is subsequently changed to floating. If the wrapping element [*Example*: wrapThrough or wrapSquare *end example*] present as a child element also has a distance from text, then this value shall be ignored.
	
	[*Example*: Consider a floating DrawingML object which shall have one-half of an inch of padding between its top edge and the nearest text. This setting would be specified as follows:
	
	`<wp:anchor distT="457200" … >`
	`  …`
	`</wp:anchor>`
	
	The distT attribute specifies that the padding distance shall be 457200 EMUs or one-half of an inch. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_WrapDistance simple type (§5.5.3.6).

hidden (Hidden)	Specifies whether this floating DrawingML object shall be displayed. When a DrawingML object is displayed within a WordprocessingML document, that object may be hidden (i.e. present, but not visible). This attribute shall determine whether the object shall be rendered or made hidden. [*Note*: An application may have settings which allow this object to be viewed. *end note*]
	
	If this attribute is omitted, then the parent DrawingML object shall be displayed (i.e. not hidden).
	
	[*Example*: Consider a floating DrawingML object which shall be hidden within the document's content. This setting would be specified as follows:
	
	`<wp:anchor hidden="true" … >`
	`…`
	`</wp:anchor>`
	
	The hidden attribute has a value of `true`, which specifies that the DrawingML object is hidden and not displayed when the document is displayed. *end example*]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

layoutInCell (Layout In Table Cell)	Specifies how this DrawingML object shall behave when its anchor is located in a table cell; and its specified position would cause it to intersect with a table cell displayed in the document. That behavior shall be as follows:
	• When this attribute has a value of `true`, then the object shall be positioned within the existing table cell, causing the cell to be resized as needed. This means that all positioning shall be relative to the cell and not the line on which the table appears.
	• When this attribute has a value of false, then the object shall be positioned as specified, but the table shall be resized and/or relocated within the document as needed to accommodate the object. This means that all positioning shall be relative to the line on which the table appears and not the cell in which the anchor is present.
	
	If this attribute is omitted, then its default value shall be considered to be `false`.
	
	[*Example*: Consider a DrawingML picture which shall be displayed in the center of the document. If the object is contained within a table and is defined as follows:
	
	`<wp:anchor layoutInCell="true" … >`
	`…`
	`</wp:anchor>`

Attributes	Description
	The layoutInCell attribute has a value of `true`, which specifies that the object can be placed within the cell if needed, for example:  If the layoutInCell attribute was now set to `false`, the object shall be laid out outside of the cell, causing the table to be repositioned:  *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
locked (Lock Anchor)	Specifies that the anchor location for this object shall not be modified at runtime when an application edits the contents of this document. [*Guidance*: An application might have automatic behaviors which reposition the anchor for a DrawingML object based on user interaction - for example, moving it from one page to another as needed. This element shall tell applications not to perform any such behaviors. *end guidance*] If this attribute is omitted, then the anchor shall not be locked for the parent DrawingML object (i.e. a default value of `false`). [*Example*: Consider a floating DrawingML object which shall have its anchor locked at the current location. This setting would be specified as follows:

Attributes	Description
	<wp:anchor locked="true" … > … </wp:anchor> The locked attribute has a value of true, which specifies that the DrawingML object's current anchor location shall not be changed by applications editing this content. *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
relativeHeight (Relative Z-Ordering Position)	Specifies the relative Z-ordering of all DrawingML objects in this document. Each floating DrawingML object shall have a Z-ordering value, which determines which object is displayed when any two objects intersect. Higher values shall indicate higher Z-order; lower values shall indicate lower Z-order. This attribute shall only indicate the Z-order with respect to other objects in the document which have an identical behindDoc attribute value. All objects with a behindDoc value of false shall be displayed above elements with a value of true. [*Example*: Consider two floating DrawingML objects as follows: <wp:anchor relativeHeight="5" … > … </wp:anchor> … <wp:anchor relativeHeight="8" … > … </wp:anchor> The relativeHeight attribute of the second object is 8, which specifies that the second DrawingML object shall be at a higher Z-order than the first and shall be displayed whenever the two overlap. *end example*] The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
simplePos (Page Positioning)	Specifies that this object shall be positioned using the positioning information in the simplePos child element (§5.5.2.13). This positioning, when specified, will position the object on the page by placing its top left point at the x-y coordinates specified by that element. If this element is omitted, then this object shall not use the simple positioning information in the simplePos element, even when present. [*Example*: Consider a floating DrawingML object which shall be positioned at the top left corner of the page using simple positioning. This setting would be specified as follows:

Attributes	Description
	<wp:anchor simplePos="true" … > <wp:simplePos x="0" y="0" /> … </wp:anchor> The simplePos attribute has a value of true, which specifies that the DrawingML object's current position shall be dictated by the simplePos element, and hence placed at 0,0. *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Anchor">
   <sequence>
      <element name="simplePos" type="a:CT_Point2D"/>
      <element name="positionH" type="CT_PosH"/>
      <element name="positionV" type="CT_PosV"/>
      <element name="extent" type="a:CT_PositiveSize2D"/>
      <element name="effectExtent" type="CT_EffectExtent" minOccurs="0"/>
      <group ref="EG_WrapType"/>
      <element name="docPr" type="a:CT_NonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
      <element name="cNvGraphicFramePr" type="a:CT_NonVisualGraphicFrameProperties" minOccurs="0"
        maxOccurs="1"/>
      <element ref="a:graphic" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="distT" type="ST_WrapDistance" use="optional"/>
   <attribute name="distB" type="ST_WrapDistance" use="optional"/>
   <attribute name="distL" type="ST_WrapDistance" use="optional"/>
   <attribute name="distR" type="ST_WrapDistance" use="optional"/>
   <attribute name="simplePos" type="xsd:boolean"/>
   <attribute name="relativeHeight" type="xsd:unsignedInt" use="required"/>
   <attribute name="behindDoc" type="xsd:boolean" use="required"/>
   <attribute name="locked" type="xsd:boolean" use="required"/>
   <attribute name="layoutInCell" type="xsd:boolean" use="required"/>
   <attribute name="hidden" type="xsd:boolean" use="optional"/>
   <attribute name="allowOverlap" type="xsd:boolean" use="required"/>
</complexType>
```

5.5.2.4 cNvGraphicFramePr (Common DrawingML Non-Visual Properties)

This element specifies common non-visual DrawingML object properties for the parent DrawingML object. These properties are specified as child elements of this element.

[*Example*: Consider a DrawingML object in a WordprocessingML document defined as follows:

```
<wp:inline>
  …
  <wp:cNvGraphicFramePr>
    <a:graphicFrameLocks … />
  </wp:cNvGraphicFramePr>
</wp:inline>
```

The cNvGraphicFramePr element contains a set of common non-visual properties as defined by DrawingML. *end example*]

Parent Elements
anchor (§5.5.2.3); inline (§5.5.2.8)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
graphicFrameLocks (Graphic Frame Locks)	§5.1.2.1.19

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualGraphicFrameProperties">
   <sequence>
      <element name="graphicFrameLocks" type="CT_GraphicalObjectFrameLocking" minOccurs="0"
        maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.5.2.5 docPr (Drawing Object Non-Visual Properties)

This element specifies non-visual object properties for the parent DrawingML object. These properties are specified as child elements of this element.

[*Example*: Consider a DrawingML object in a WordprocessingML document defined as follows:

```
<wp:inline>
  …
  <wp:docPr id="1" name="Example Object">
    <a:hlinkClick … />
    <a:hlinkHover … />
  </wp:docPr>
</wp:inline>
```

The docPr element contains a set of common non-visual properties for this object. *end example*]

Parent Elements
anchor (§5.5.2.3); inline (§5.5.2.8)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
hlinkClick (Click Hyperlink)	§5.1.5.3.5
hlinkHover (Hyperlink for Hover)	§5.1.2.1.23

Attributes	Description
descr (Alternative Text for Object) Namespace: .../drawingml/2006/main	Specifies alternative text for the current DrawingML object, for use by assistive technologies or applications which will not display the current object. If this element is omitted, then no alternative text is present for the parent object. [*Example*: Consider a DrawingML object defined as follows: `<… descr="A picture of a bowl of fruit">` The descr attribute contains alternative text which may be used in place of the actual DrawingML object. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
hidden (Hidden) Namespace: .../drawingml/2006/main	Specifies whether this DrawingML object shall be displayed. When a DrawingML object is displayed within a document, that object may be hidden (i.e., present, but not visible). This attribute shall determine whether the object shall be rendered or made hidden. [*Note*: An application may have settings which allow this object to be viewed. *end note*] If this attribute is omitted, then the parent DrawingML object shall be displayed (i.e., not hidden). [*Example*: Consider an inline DrawingML object which shall be hidden within the document's content. This setting would be specified as follows: `<… hidden="true" />` The hidden attribute has a value of `true`, which specifies that the DrawingML object is hidden and not displayed when the document is displayed. *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
id (Unique Identifier) Namespace: .../drawingml/2006/main	Specifies a unique identifier for the current DrawingML object within the current document. This ID may be used to assist in uniquely identifying this object so that it can be referred to by other parts of the document. If multiple objects within the same document share the same id attribute value, then the document shall be considered non-conformant.

Attributes	Description
	[*Example*: Consider a DrawingML object defined as follows: `<… id="10" … >` The id attribute has a value of 10, which is the unique identifier for this DrawingML object. *end example*] The possible values for this attribute are defined by the ST_DrawingElementId simple type (§5.1.12.19).
name (Name) Namespace: …/drawingml/2006/main	Specifies the name of the object. [*Note*: Typically, this will be used to store the original file name of a picture object. *end note*] [*Example*: Consider a DrawingML object defined as follows: `< … name="foo.jpg" >` The name attribute has a value of foo.jpg, which is the name of this DrawingML object. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

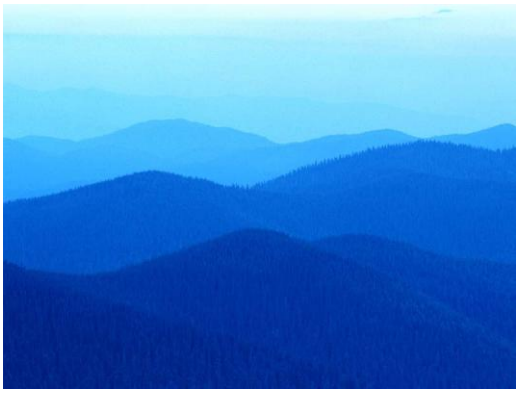
```
<complexType name="CT_NonVisualDrawingProps">
   <sequence>
      <element name="hlinkClick" type="CT_Hyperlink" minOccurs="0" maxOccurs="1"/>
      <element name="hlinkHover" type="CT_Hyperlink" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="id" type="ST_DrawingElementId" use="required"/>
   <attribute name="name" type="xsd:string" use="required"/>
   <attribute name="descr" type="xsd:string" use="optional" default=""/>
   <attribute name="hidden" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.5.2.6 effectExtent (Object Extents Including Effects)

This element specifies the additional extent which shall be added to each edge of the image (top, bottom, left, right) in order to compensate for any drawing effects applied to the DrawingML object.

The extent element (§5.5.2.7) specifies the size of the actual DrawingML object; however, an object may have effects applied which change its overall size [*Example*: A reflection and/or shadow effect. *end example*]. The additional size for each edge of the shape shall be stored on this element, and used to calculate the appropriate wrapping for wrap types without a wrapping polygon and the appropriate line height for inline objects.

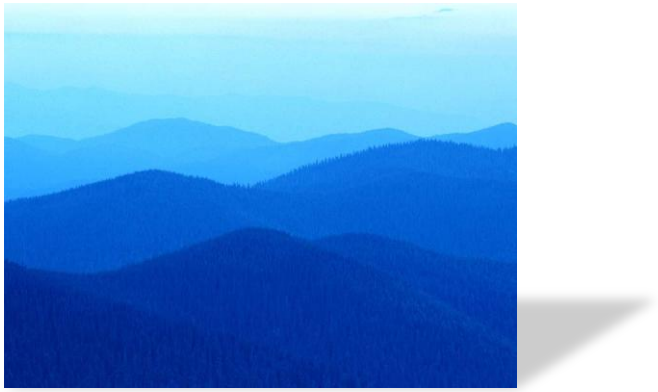
[*Example*: Consider the following DrawingML image:



This object has no effects, and hence would have the following effect extents:

```
<wp:effectExtents b="0" t="0" l="0" r="0" />
```

However, if a shadow effect was applied which added effects to the right of the image:



Then the additional extent the right side would be specified in the r attribute on this element:

```
<wp:effectExtents b="0" t="0" l="0" r="695325" />
```

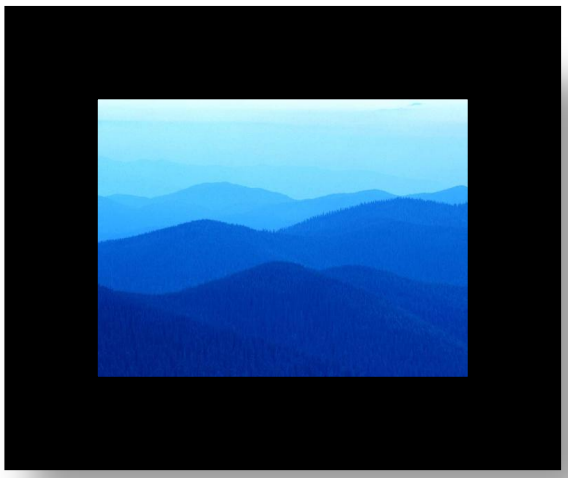
The r attribute has a value of 695325, specifying that that 695325 EMUs shall be added to the right side of the image. *end example*]

Parent Elements
anchor (§5.5.2.3); inline (§5.5.2.8); wrapSquare (§5.5.2.17); wrapTopAndBottom (§5.5.2.20)

Attributes	Description
b (Additional Extent on Bottom Edge)	Specifies the additional length, in EMUs, which shall be added to the bottom edge of the DrawingML object to determine its actual bottom edge including effects. [Example: Consider the following DrawingML image:

Attributes	Description
	 This image has an effect on all four sides, resulting in the following markup: `<wp:effectExtent l="504825" t="447675" r="771525" b="809625" />` The b attribute value of 809625 specifies that 809625 additional EMUs shall be added to the bottom of the image to compensate for the effects on the image. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).
l (Additional Extent on Left Edge)	Specifies the additional length, in EMUs, which shall be added to the bottom edge of the DrawingML object to determine its actual bottom edge including effects. [Example: Consider the following DrawingML image:  This image has an effect on all four sides, resulting in the following markup:

Attributes	Description
	<wp:effectExtent l="504825" t="447675" r="771525" b="809625" /> The l attribute value of 504825 specifies that 504825 additional EMUs shall be added to the bottom of the image to compensate for the effects on the image. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).
r (Additional Extent on Right Edge)	Specifies the additional length, in EMUs, which shall be added to the bottom edge of the DrawingML object to determine its actual bottom edge including effects. [Example: Consider the following DrawingML image:  This image has an effect on all four sides, resulting in the following markup: <wp:effectExtent l="504825" t="447675" r="771525" b="809625" /> The r attribute value of 771525 specifies that 771525 additional EMUs shall be added to the bottom of the image to compensate for the effects on the image. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).
t (Additional Extent on Top Edge)	Specifies the additional length, in EMUs, which shall be added to the bottom edge of the DrawingML object to determine its actual bottom edge including effects. [Example: Consider the following DrawingML image:

Attributes	Description
	 This image has an effect on all four sides, resulting in the following markup: `<wp:effectExtent l="504825" t="447675" r="771525" b="809625" />` The t attribute value of 447675 specifies that 447675 additional EMUs shall be added to the bottom of the image to compensate for the effects on the image. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EffectExtent">
    <attribute name="l" type="a:ST_Coordinate" use="required"/>
    <attribute name="t" type="a:ST_Coordinate" use="required"/>
    <attribute name="r" type="a:ST_Coordinate" use="required"/>
    <attribute name="b" type="a:ST_Coordinate" use="required"/>
</complexType>
```

5.5.2.7 extent (Drawing Object Size)

This element specifies the extents of the parent DrawingML object within the document (i.e. its final height and width).

[*Example*: Consider a DrawingML picture which is present in a WordprocessingML document and has an equal height and width. This object would be specified as follows:

```
<wp:anchor relativeHeight="10" allowOverlap="true">
  …
  <wp:extent cx="1828800" cy="1828800"/>
  …
</wp:anchor>
```

The extent element specifies via its attributes that this object has a height and width of 1828800 EMUs (English Metric Units). *end example*]

Parent Elements
anchor (§5.5.2.3); inline (§5.5.2.8)

Attributes	Description
cx (Extent Length) Namespace: .../drawingml/2006/main	Specifies the length of the extents rectangle in EMUs. This rectangle shall dictate the size of the object as displayed (the result of any scaling to the original object). [*Example*: Consider a DrawingML object specified as follows: <… cx="1828800" cy="200000"/> The cx attributes specifies that this object has a height of 1828800 EMUs (English Metric Units). *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).
cy (Extent Width) Namespace: .../drawingml/2006/main	Specifies the width of the extents rectangle in EMUs. This rectangle shall dictate the size of the object as displayed (the result of any scaling to the original object). [*Example*: Consider a DrawingML object specified as follows: < … cx="1828800" cy="200000"/> The cy attribute specifies that this object has a width of 200000 EMUs (English Metric Units). *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PositiveSize2D">
   <attribute name="cx" type="ST_PositiveCoordinate" use="required"/>
   <attribute name="cy" type="ST_PositiveCoordinate" use="required"/>
</complexType>
```

5.5.2.8 inline (Inline DrawingML Object)

This element specifies that the DrawingML object located at this position in the document is an inline object. Within a WordprocessingML document, drawing objects can exist in two states:

- *Inline* - The drawing object is in line with the text, and affects the line height and layout of its line (like a character glyph of similar size).

- *Floating* - The drawing object is anchored within the text, but may be absolutely positioned in the document relative to the page.

When this element encapsulates the DrawingML object's information, then all child elements shall dictate the positioning of this object in line with text.

[*Example*: Consider a WordprocessingML document where an inline DrawingML object shall be the first piece of run content within a paragraph. That paragraph's content would be specified as follows:

```
<w:p>
  <w:r>
    <w:drawing>
      <wp:inline>
        …
      </wp:inline>
    </w:drawing>
  </w:r>
</w:p>
```

The inline element, when present as the child element of the drawing element, specifies that this DrawingML object shall be positioned in line with the text of this paragraph, modifying line heights, etc. as necessary. *end example*]

Parent Elements
drawing (§2.3.3.9)

Child Elements	Subclause
cNvGraphicFramePr (Common DrawingML Non-Visual Properties)	§5.5.2.4
docPr (Drawing Object Non-Visual Properties)	§5.5.2.5
effectExtent (Object Extents Including Effects)	§5.5.2.6
extent (Drawing Object Size)	§5.5.2.7
graphic (Graphic Object)	§5.1.2.1.16

Attributes	Description
distB (Distance From Text on Bottom Edge)	Specifies the minimum distance which shall be maintained between the bottom edge of this drawing object and any subsequent text within the document when this graphical object is displayed within the document's contents. The distance shall be measured in EMUs (English Metric Units). If this object is an inline object (i.e. has a parent element of inline), then this value shall

Attributes	Description

	not have any effect when displaying the object in line with text, but may be maintained and used if the object is subsequently changed to floating. If the wrapping element [*Example*: wrapThrough or wrapSquare *end example*] present as a child element also has a distance from text, then this value shall be ignored.
	
	[*Example*: Consider a floating DrawingML object which shall have one-half of an inch of padding between its bottom edge and the nearest text. This setting would be specified as follows:
	
	`<wp:anchor distB="457200" … >`
	`  …`
	`</wp:anchor>`
	
	The distB attribute specifies that the padding distance shall be 457200 EMUs or one-half of an inch. *end example*]
	
	The possible values for this attribute are defined by the ST_WrapDistance simple type (§5.5.3.6).

distL (Distance From Text on Left Edge)	Specifies the minimum distance which shall be maintained between the left edge of this drawing object and any subsequent text within the document when this graphical object is displayed within the document's contents.
	
	The distance shall be measured in EMUs (English Metric Units).
	
	If this object is an inline object (i.e. has a parent element of inline), then this value shall not have any effect when displaying the object in line with text, but may be maintained and used if the object is subsequently changed to floating. If the wrapping element [*Example*: wrapThrough or wrapSquare *end example*] present as a child element also has a distance from text, then this value shall be ignored.
	
	[*Example*: Consider a floating DrawingML object which shall have one-quarter of an inch of padding between its left edge and the nearest text. This setting would be specified as follows:
	
	`<wp:anchor distL="228600" … >`
	`  …`
	`</wp:anchor>`
	
	The distL attribute specifies that the padding distance shall be 228600 EMUs or one-quarter of an inch. *end example*]
	
	The possible values for this attribute are defined by the ST_WrapDistance simple type (§5.5.3.6).

distR (Distance From Text on Right Edge)	Specifies the minimum distance which shall be maintained between the right edge of this drawing object and any subsequent text within the document when this graphical object is displayed within the document's contents.

Attributes	Description

	The distance shall be measured in EMUs (English Metric Units).
	
	If this object is an inline object (i.e. has a parent element of inline), then this value shall not have any effect when displaying the object in line with text, but may be maintained and used if the object is subsequently changed to floating. If the wrapping element [*Example*: wrapThrough or wrapSquare *end example*] present as a child element also has a distance from text, then this value shall be ignored.
	
	[*Example*: Consider a floating DrawingML object which shall have one-quarter of an inch of padding between its right edge and the nearest text. This setting would be specified as follows:
	
	`<wp:anchor distR="228600" … >`
	`  …`
	`</wp:anchor>`
	
	The distR attribute specifies that the padding distance shall be 228600 EMUs or one-quarter of an inch. *end example*]
	
	The possible values for this attribute are defined by the ST_WrapDistance simple type (§5.5.3.6).

distT (Distance From Text on Top Edge)	Specifies the minimum distance which shall be maintained between the top edge of this drawing object and any subsequent text within the document when this graphical object is displayed within the document's contents.
	
	The distance shall be measured in EMUs (English Metric Units).
	
	If this object is an inline object (i.e. has a parent element of inline), then this value shall not have any effect when displaying the object in line with text, but may be maintained and used if the object is subsequently changed to floating. If the wrapping element [*Example*: wrapThrough or wrapSquare *end example*] present as a child element also has a distance from text, then this value shall be ignored.
	
	[*Example*: Consider a floating DrawingML object which shall have one-half of an inch of padding between its top edge and the nearest text. This setting would be specified as follows:
	
	`<wp:anchor distT="457200" … >`
	`  …`
	`</wp:anchor>`
	
	The distT attribute specifies that the padding distance shall be 457200 EMUs or one-half of an inch. *end example*]
	
	The possible values for this attribute are defined by the ST_WrapDistance simple type

Attributes	Description
	(§5.5.3.6).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Inline">
   <sequence>
      <element name="extent" type="a:CT_PositiveSize2D"/>
      <element name="effectExtent" type="CT_EffectExtent" minOccurs="0"/>
      <element name="docPr" type="a:CT_NonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
      <element name="cNvGraphicFramePr" type="a:CT_NonVisualGraphicFrameProperties" minOccurs="0"
        maxOccurs="1"/>
      <element ref="a:graphic" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="distT" type="ST_WrapDistance" use="optional"/>
   <attribute name="distB" type="ST_WrapDistance" use="optional"/>
   <attribute name="distL" type="ST_WrapDistance" use="optional"/>
   <attribute name="distR" type="ST_WrapDistance" use="optional"/>
</complexType>
```

5.5.2.9 lineTo (Wrapping Polygon Line End Position)

This element specifies a single point on the wrapping polygon for a DrawingML object. This point shall be the termination of the edge of the wrapping polygon started by the previous start or lineTo element in document order, and shall be the origin of the next edge on the same polygon.

The attributes on this element shall dictate the position of the point relative to the upper-left corner of the actual object.

[*Example*: Consider the following basic wrapping polygon for a DrawingML object:

```
<wp:wrapPolygon>
  <wp:start x="0" y="0" />
  <wp:lineTo x="0" y="100" />
  <wp:lineTo x="100" y="100" />
  <wp:lineTo x="100" y="0" />
  <wp:lineTo x="0" y="0" />
</wp:wrapPolygon>
```

The lineTo element defines each point of the wrapping polygon (in this case, the four points of the wrapping square). *end example*]

Parent Elements
wrapPolygon (§5.5.2.16)

Attributes	Description
x (X-Axis	Specifies a coordinate on the x-axis. The origin point for this coordinate shall be specified

Attributes	Description
Coordinate) Namespace: .../drawingml/2006/main	by the parent XML element. [*Example*: Consider the following point on a basic wrapping polygon for a DrawingML object: `<wp:… x="0" y="100" />` The x attribute defines an x-coordinate of `0`. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).
y (Y-Axis Coordinate) Namespace: .../drawingml/2006/main	Specifies a coordinate on the x-axis. The origin point for this coordinate shall be specified by the parent XML element. [*Example*: Consider the following point on a basic wrapping polygon for a DrawingML object: `<wp:… x="0" y="100" />` The y attribute defines a y-coordinate of `100`. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Point2D">
   <attribute name="x" type="ST_Coordinate" use="required"/>
   <attribute name="y" type="ST_Coordinate" use="required"/>
</complexType>
```

5.5.2.10 positionH (Horizontal Positioning)

This element specifies the horizontal positioning of a floating DrawingML object within a WordprocessingML document. This positioning is specified in two parts:

- Positioning Base - The relativeFrom attribute on this element specifies the part of the document from which the positioning shall be calculated.
- Positioning - The child element of this element (align or posOffset) specifies how the object is positioned relative to that base.

[*Example*: Consider a DrawingML picture which shall be displayed in the center of the printed page on which it appears, modifying the flow of text as necessary. This object would be specified as follows:

```
<wp:anchor … >
  <wp:positionH relativeFrom="margin">
    <wp:align>center</wp:align>
  </wp:positionH>
  <wp:positionV relativeFrom="margin">
    <wp:align>center</wp:align>
  </wp:positionV>
</wp:anchor>
```

The positionH element specifies that the object is horizontally positioned relative to the margin via the relativeFrom attribute; and that the alignment relative to the margin is centered via the align element. *end example*]

Parent Elements
anchor (§5.5.2.3)

Child Elements	Subclause
align (Relative Horizontal Alignment)	§5.5.2.1
posOffset (Absolute Position Offset)	§5.5.2.12

Attributes	Description
relativeFrom (Horizontal Position Relative Base)	Specifies the base to which the relative horizontal positioning of this object shall be calculated. [*Example*: Consider a DrawingML picture which shall be displayed at the bottom center of the page. This object would be specified as follows: `<wp:anchor … > <wp:positionH relativeFrom="page"> <wp:align>center</wp:align> </wp:positionH> … </wp:anchor>` The relativeFrom attribute specifies that the object is horizontally positioned relative to the page. *end example*] The possible values for this attribute are defined by the ST_RelFromH simple type (§5.5.3.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PosH">
   <sequence>
      <choice minOccurs="1" maxOccurs="1">
         <element name="align" type="ST_AlignH" minOccurs="1" maxOccurs="1"/>
         <element name="posOffset" type="ST_PositionOffset" minOccurs="1" maxOccurs="1"/>
      </choice>
   </sequence>
   <attribute name="relativeFrom" type="ST_RelFromH" use="required"/>
</complexType>
```

5.5.2.11 positionV (Vertical Positioning)

This element specifies the vertical positioning of a floating DrawingML object within a WordprocessingML document. This positioning is specified in two parts:

- Positioning Base - The relativeFrom attribute on this element specifies the part of the document from which the positioning shall be calculated.
- Positioning - The child element of this element (align or posOffset) specifies how the object is positioned relative to that base.

[*Example*: Consider a DrawingML picture which shall be displayed in the center of the printed page on which it appears, modifying the flow of text as necessary. This object would be specified as follows:

```
<wp:anchor … >
  <wp:positionH relativeFrom="margin">
    <wp:align>center</wp:align>
  </wp:positionH>
  <wp:positionV relativeFrom="margin">
    <wp:align>center</wp:align>
  </wp:positionV>
</wp:anchor>
```

The positionV element specifies that the object is vertically positioned relative to the margin via the relativeFrom attribute; and that the alignment relative to the margin is centered via the align element. *end example*]

Parent Elements
anchor (§5.5.2.3)

Child Elements	Subclause
align (Relative Vertical Alignment)	§5.5.2.2
posOffset (Absolute Position Offset)	§5.5.2.12

Attributes	Description
relativeFrom (Vertical Position Relative Base)	Specifies the base to which the relative vertical positioning of this object shall be calculated. [*Example*: Consider a DrawingML picture which shall be displayed at the bottom center of the page margins. This object would be specified as follows: `<wp:anchor … > … <wp:positionV relativeFrom="margin"> <wp:align>bottom</wp:align> </wp:positionV> </wp:anchor>` The relativeFrom attribute specifies that the object is horizontally positioned relative to the margin. *end example*] The possible values for this attribute are defined by the ST_RelFromV simple type (§5.5.3.5).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PosV">
    <sequence>
        <choice minOccurs="1" maxOccurs="1">
            <element name="align" type="ST_AlignV" minOccurs="1" maxOccurs="1"/>
            <element name="posOffset" type="ST_PositionOffset" minOccurs="1" maxOccurs="1"/>
        </choice>
    </sequence>
    <attribute name="relativeFrom" type="ST_RelFromV" use="required"/>
</complexType>
```

5.5.2.12 posOffset (Absolute Position Offset)

This element specifies an absolute measurement for the positioning of a floating DrawingML object within a WordprocessingML document. This measurement shall be calculated relative to the top left edge of the positioning base specified by the parent element's relativeFrom attribute.

[*Example*: Consider a DrawingML picture which shall be displayed one inch from the top of the page, and one-half of an inch from the left edge of the page. This object would be specified as follows:

```
<wp:anchor … >
  <wp:positionH relativeFrom="page">
    <wp:posOffset>914400</wp:posOffset>
  </wp:positionH>
  <wp:positionV relativeFrom="page">
    <wp:posOffset>457200</wp:posOffset>
  </wp:positionV>
</wp:anchor>
```

The posOffset element specifies the absolute positioning of the object relative to the top-left edge of the page in EMUs. *end example*]

The possible values for this element are defined by the ST_PositionOffset simple type (§5.5.3.3).

Parent Elements
positionH (§5.5.2.10); positionV (§5.5.2.11)

5.5.2.13 simplePos (Simple Positioning Coordinates)

This element specifies the coordinates at which a DrawingML object shall be positioned relative to the top-left edge of its page, when the simplePos attribute is specified on the anchor element (§5.5.2.3).

[*Example*: Consider a floating DrawingML object which shall be positioned at the top left corner of the page using simple positioning. This setting would be specified as follows:

```
<wp:anchor simplePos="true" … >
  <wp:simplePos x="0" y="0" />
  …
</wp:anchor>
```

The simplePos attribute has a value of `true`, which specifies that the DrawingML object's current position shall be dictated by the simplePos element, and hence placed at `0`,`0`. *end example*]

Parent Elements
anchor (§5.5.2.3)

Attributes	Description
x (X-Axis Coordinate) Namespace: …/drawingml/2006/main	Specifies a coordinate on the x-axis. The origin point for this coordinate shall be specified by the parent XML element. [*Example*: Consider the following point on a basic wrapping polygon for a DrawingML object: `<wp:… x="0" y="100" />` The x attribute defines an x-coordinate of `0`. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).
y (Y-Axis Coordinate) Namespace: …/drawingml/200	Specifies a coordinate on the x-axis. The origin point for this coordinate shall be specified by the parent XML element. [*Example*: Consider the following point on a basic wrapping polygon for a DrawingML object:

Attributes	Description
6/main	<wp:… x="0" y="100" /> The y attribute defines a y-coordinate of 100. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Point2D">
   <attribute name="x" type="ST_Coordinate" use="required"/>
   <attribute name="y" type="ST_Coordinate" use="required"/>
</complexType>
```

5.5.2.14 start (Wrapping Polygon Start)

This element specifies the starting point on the wrapping polygon for a DrawingML object. This point shall be the start and termination of the wrapping polygon for the parent object.

The attributes on this element shall dictate the position of the point relative to the upper-left corner of the actual object.

[*Example*: Consider the following basic wrapping polygon for a DrawingML object:

```
<wp:wrapPolygon>
  <wp:start x="0" y="0" />
  <wp:lineTo x="0" y="100" />
  <wp:lineTo x="100" y="100" />
  <wp:lineTo x="100" y="0" />
  <wp:lineTo x="0" y="0" />
</wp:wrapPolygon>
```

The start element defines the start and end of the wrapping polygon (in this case, the four points of the wrapping square). *end example*]

Parent Elements
wrapPolygon (§5.5.2.16)

Attributes	Description
x (X-Axis Coordinate) Namespace: …/drawingml/200	Specifies a coordinate on the x-axis. The origin point for this coordinate shall be specified by the parent XML element. [*Example*: Consider the following point on a basic wrapping polygon for a DrawingML object:

Attributes	Description
6/main	<wp:… x="0" y="100" /> The x attribute defines an x-coordinate of 0. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).
y (Y-Axis Coordinate) Namespace: …/drawingml/2006/main	Specifies a coordinate on the x-axis. The origin point for this coordinate shall be specified by the parent XML element. [*Example*: Consider the following point on a basic wrapping polygon for a DrawingML object: <wp:… x="0" y="100" /> The y attribute defines a y-coordinate of 100. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Point2D">
   <attribute name="x" type="ST_Coordinate" use="required"/>
   <attribute name="y" type="ST_Coordinate" use="required"/>
</complexType>
```

5.5.2.15 wrapNone (No Text Wrapping)

This element specifies that the parent DrawingML object shall not cause any text wrapping within the contents of the host WordprocessingML document based on its display location. In effect, this setting shall place the object in one of two locations:

- If the behindDoc attribute on the parent element is true, then the object shall be positioned behind the text as it is normally displayed.
- If the behindDoc attribute on the parent element is false, then the object shall be positioned in front of the text as it is normally displayed.

[*Example*: Consider a DrawingML picture which shall be displayed in front of any text on the page. This object would be specified as follows:

```
<wp:anchor relativeHeight="10" behindDoc="false">
  …
  <wp:wrapNone/>
</wp:anchor>
```

The wrapNone element specifies that the DrawingML object shall not cause any text wrapping, and since the behindDoc attribute is `false`, the object shall be displayed in front of the text of the document. *end example*]

Parent Elements
anchor (§5.5.2.3)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_WrapNone"/>
```

5.5.2.16 wrapPolygon (Wrapping Polygon)

This element specifies the wrapping polygon which shall be used to determine the extents to which text may wrap around the specified object in the document. This polygon shall be defined by the following:

- The start element defines the coordinates of the origin of the wrap polygon
- Two or more lineTo elements define the point of the wrap polygon

If the set of child elements does not result in a closed polygon (the last lineTo element does not return to the position specified by the start element), then a single additional line shall be inferred as needed to close the wrapping polygon.

[*Example*: Consider the following basic wrapping polygon for a DrawingML object:

```
<wp:wrapPolygon>
  <wp:start x="0" y="0" />
  <wp:lineTo x="0" y="100" />
  <wp:lineTo x="100" y="100" />
  <wp:lineTo x="100" y="0" />
  <wp:lineTo x="0" y="0" />
</wp:wrapPolygon>
```

The wrapPolygon element defines the object's text wrapping polygon (in this case, the four points of a square). *end example*]

Parent Elements
wrapThrough (§5.5.2.18); wrapTight (§5.5.2.19)

Child Elements	Subclause
lineTo (Wrapping Polygon Line End Position)	§5.5.2.9
start (Wrapping Polygon Start)	§5.5.2.14

Attributes	Description

Attributes	Description

edited (Wrapping Points Modified)	Specifies that the wrap points for the wrapping polygon have been edited, and the resulting extents shall be recalculated to compensate when the document is next opened.
	
	[*Example*: Consider the following basic wrapping polygon for a DrawingML object:
	
	`<wp:wrapPolygon edited="true">`
	`<wp:start x="0" y="0" />`
	`<wp:lineTo x="0" y="100" />`
	`<wp:lineTo x="50" y="50" />`
	`<wp:lineTo x="0" y="0" />`
	`</wp:wrapPolygon>`
	
	The edited attribute specifies that these wrap points have been changed since the document was last rendered. *end example*]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_WrapPath">
   <sequence>
      <element name="start" type="a:CT_Point2D" minOccurs="1" maxOccurs="1"/>
      <element name="lineTo" type="a:CT_Point2D" minOccurs="2" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="edited" type="xsd:boolean" use="optional"/>
</complexType>
```

5.5.2.17 wrapSquare (Square Wrapping)

This element specifies that text shall wrap around a virtual rectangle bounding this object. The bounds of the wrapping rectangle shall be dictated by the extents including the addition of the effectExtent element as a child of this element (if present) or the effectExtent present on the parent element.

[*Example*: Consider a DrawingML object using square wrapping and defined as follows:

```
<wp:anchor … >
  …
  <wp:wrapSquare wrapText="bothSides" />
</wp:anchor>
```

The wrapSquare element specifies that text shall wrap around both sides of a rectangle around this object which includes its effect extents. *end example*]

Parent Elements
anchor (§5.5.2.3)

Child Elements	Subclause
effectExtent (Object Extents Including Effects)	§5.5.2.6

Attributes	Description

distB (Distance From Text on Bottom Edge)	Specifies the minimum distance which shall be maintained between the bottom edge of this drawing object and any subsequent text within the document when this graphical object is displayed within the document's contents.
	
	The distance shall be measured in EMUs (English Metric Units).
	
	[*Example*: Consider a floating DrawingML object which shall have one-half of an inch of padding between its bottom edge and the nearest text. This setting would be specified as follows:
	
	`<wp:anchor … >`
	`  …`
	`  <wp:wrapSquare distB="457200" … />`
	`</wp:anchor>`
	
	The distB attribute specifies that the padding distance shall be 457200 EMUs or one-half of an inch. *end example*]
	
	The possible values for this attribute are defined by the ST_WrapDistance simple type (§5.5.3.6).

distL (Distance From Text on Left Edge)	Specifies the minimum distance which shall be maintained between the left edge of this drawing object and any subsequent text within the document when this graphical object is displayed within the document's contents.
	
	The distance shall be measured in EMUs (English Metric Units).
	
	[*Example*: Consider a floating DrawingML object which shall have one-half of an inch of padding between its left edge and the nearest text. This setting would be specified as follows:
	
	`<wp:anchor … >`
	`  …`
	`  <wp:wrapSquare distL="457200" … />`
	`</wp:anchor>`
	
	The distL attribute specifies that the padding distance shall be 457200 EMUs or one-half of an inch. *end example*]
	
	The possible values for this attribute are defined by the ST_WrapDistance simple type (§5.5.3.6).

distR (Distance	Specifies the minimum distance which shall be maintained between the right edge of this

Attributes	Description

From Text on Right Edge)	drawing object and any subsequent text within the document when this graphical object is displayed within the document's contents.
	
	The distance shall be measured in EMUs (English Metric Units).
	
	[*Example*: Consider a floating DrawingML object which shall have one-half of an inch of padding between its right edge and the nearest text. This setting would be specified as follows:
	
	`<wp:anchor … >`
	`  …`
	`  <wp:wrapSquare distR="457200" … />`
	`</wp:anchor>`
	
	The distR attribute specifies that the padding distance shall be 457200 EMUs or one-half of an inch. *end example*]
	
	The possible values for this attribute are defined by the ST_WrapDistance simple type (§5.5.3.6).

distT (Distance From Text (Top))	Specifies the minimum distance which shall be maintained between the top edge of this drawing object and any subsequent text within the document when this graphical object is displayed within the document's contents.
	
	The distance shall be measured in EMUs (English Metric Units).
	
	[*Example*: Consider a floating DrawingML object which shall have one-half of an inch of padding between its top edge and the nearest text. This setting would be specified as follows:
	
	`<wp:anchor … >`
	`  …`
	`  <wp:wrapSquare distT="457200" … />`
	`</wp:anchor>`
	
	The distT attribute specifies that the padding distance shall be 457200 EMUs or one-half of an inch. *end example*]
	
	The possible values for this attribute are defined by the ST_WrapDistance simple type (§5.5.3.6).

wrapText (Text Wrapping Location)	Specifies how text shall wrap around the object's left and right sides.
	
	[*Example*: Consider a floating DrawingML object which shall allow text to wrap around its left side only. This setting would be specified as follows:
	
	`<wp:anchor … >`
	`  …`

Attributes	Description
	<wp:wrapSquare wrapText="left" … /> </wp:anchor> The wrapText attribute value of `left` specifies that text shall only wrap around the let side of the object. *end example*] The possible values for this attribute are defined by the ST_WrapText simple type (§5.5.3.7).

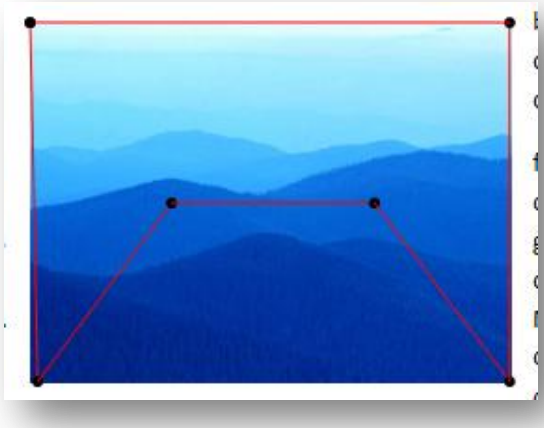
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_WrapSquare">
   <sequence>
      <element name="effectExtent" type="CT_EffectExtent" minOccurs="0"/>
   </sequence>
   <attribute name="wrapText" type="ST_WrapText" use="required"/>
   <attribute name="distT" type="ST_WrapDistance" use="optional"/>
   <attribute name="distB" type="ST_WrapDistance" use="optional"/>
   <attribute name="distL" type="ST_WrapDistance" use="optional"/>
   <attribute name="distR" type="ST_WrapDistance" use="optional"/>
</complexType>
```

5.5.2.18 wrapThrough (Through Wrapping)

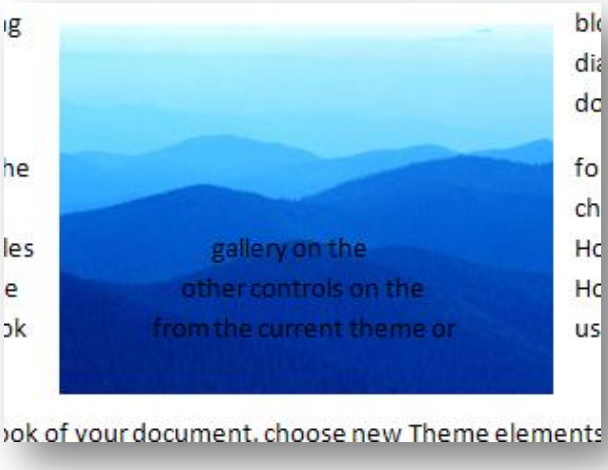
This element specifies that text shall wrap around the wrapping polygon bounding this object as defined by the child wrapPolygon element. When this element specifies a wrapping polygon, it shall allow text to wrap within the object's maximum left and right extents.

[*Example*: Consider an object with the following wrap points:



If this object uses tight wrapping, then text cannot be placed within the maximum left and right extents of the wrap polygon at any location:

However, with through wrapping:



end example]

[*Example*: Consider a DrawingML object using through wrapping and defined as follows:

```
<wp:anchor … >
  …
  <wp:wrapThrough wrapText="bothSides">
    …
  </wp:wrapThrough>
</wp:anchor>
```

The wrapThrough element specifies that text shall wrap through this object as defined by its wrap polygon. *end example*]

Parent Elements
anchor (§5.5.2.3)

Child Elements	Subclause
wrapPolygon (Wrapping Polygon)	§5.5.2.16

Attributes	Description

distL (Distance From Text on Left Edge)	Specifies the minimum distance which shall be maintained between the left edge of this drawing object and any subsequent text within the document when this graphical object is displayed within the document's contents.
	
	The distance shall be measured in EMUs (English Metric Units).
	
	[*Example*: Consider a floating DrawingML object which shall have one-half of an inch of padding between its left edge and the nearest text. This setting would be specified as follows:
	
	`<wp:anchor … >`
	`  …`
	`  <wp:wrapThrough distL="457200" … />`
	`</wp:anchor>`
	
	The distL attribute specifies that the padding distance shall be 457200 EMUs or one-half of an inch. *end example*]
	
	The possible values for this attribute are defined by the ST_WrapDistance simple type (§5.5.3.6).

distR (Distance From Text on Right Edge)	Specifies the minimum distance which shall be maintained between the right edge of this drawing object and any subsequent text within the document when this graphical object is displayed within the document's contents.
	
	The distance shall be measured in EMUs (English Metric Units).
	
	[*Example*: Consider a floating DrawingML object which shall have one-half of an inch of padding between its right edge and the nearest text. This setting would be specified as follows:
	
	`<wp:anchor … >`
	`  …`
	`  <wp:wrapThrough distR="457200" … />`
	`</wp:anchor>`
	
	The distR attribute specifies that the padding distance shall be 457200 EMUs or one-half of an inch. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_WrapDistance simple type (§5.5.3.6).

wrapText (Text Wrapping Location)	Specifies how text shall wrap around the object's left and right sides.
	
	[*Example*: Consider a floating DrawingML object which shall allow text to wrap around its left side only. This setting would be specified as follows:
	
	`<wp:anchor … >`
	`  …`
	`  <wp:wrapThrough wrapText="left" … />`
	`</wp:anchor>`
	
	The wrapText attribute value of `left` specifies that text shall only wrap around the let side of the object. *end example*]
	
	The possible values for this attribute are defined by the ST_WrapText simple type (§5.5.3.7).

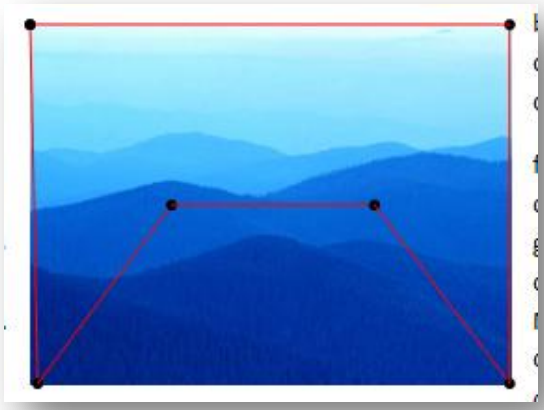
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_WrapThrough">
   <sequence>
      <element name="wrapPolygon" type="CT_WrapPath" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="wrapText" type="ST_WrapText" use="required"/>
   <attribute name="distL" type="ST_WrapDistance" use="optional"/>
   <attribute name="distR" type="ST_WrapDistance" use="optional"/>
</complexType>
```

5.5.2.19 wrapTight (Tight Wrapping)

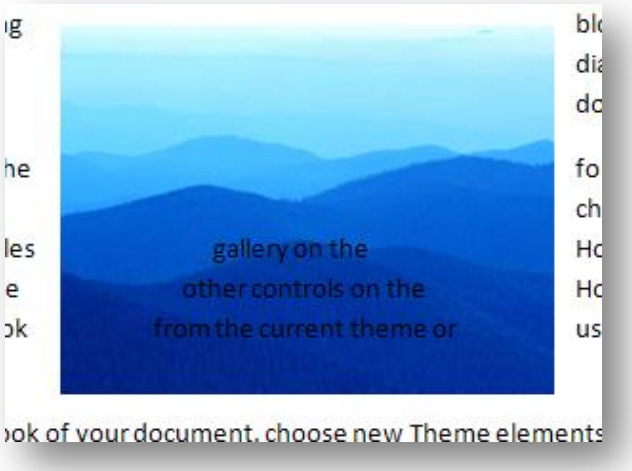
This element specifies that text shall wrap around the wrapping polygon bounding this object as defined by the child wrapPolygon element. When this element specifies a wrapping polygon, it shall not allow text to wrap within the object's maximum left and right extents.

[*Example*: Consider an object with the following wrap points:



If this object uses tight wrapping, then text cannot be placed within the maximum left and right extents of the wrap polygon at any location:

However, with through wrapping:



end example]

[*Example*: Consider a DrawingML object using tight wrapping and defined as follows:

```
<wp:anchor … >
  …
  <wp:wrapTight wrapText="bothSides">
    …
  </wp:wrapTight>
</wp:anchor>
```

The wrapTight element specifies that text shall wrap through this object as defined by its wrap polygon. *end example*]

Parent Elements
anchor (§5.5.2.3)

Child Elements	Subclause
wrapPolygon (Wrapping Polygon)	§5.5.2.16

Attributes	Description
distL (Distance From Test on Left Edge)	Specifies the minimum distance which shall be maintained between the left edge of this drawing object and any subsequent text within the document when this graphical object is displayed within the document's contents. The distance shall be measured in EMUs (English Metric Units). [*Example*: Consider a floating DrawingML object which shall have one-half of an inch of

Attributes	Description

	padding between its left edge and the nearest text. This setting would be specified as follows:
	
	`<wp:anchor … >`
	`  …`
	`  <wp:wrapTight distL="457200" … />`
	`</wp:anchor>`
	
	The distL attribute specifies that the padding distance shall be 457200 EMUs or one-half of an inch. *end example*]
	
	The possible values for this attribute are defined by the ST_WrapDistance simple type (§5.5.3.6).

distR (Distance From Text on Right Edge)	Specifies the minimum distance which shall be maintained between the right edge of this drawing object and any subsequent text within the document when this graphical object is displayed within the document's contents.
	
	The distance shall be measured in EMUs (English Metric Units).
	
	[*Example*: Consider a floating DrawingML object which shall have one-half of an inch of padding between its right edge and the nearest text. This setting would be specified as follows:
	
	`<wp:anchor … >`
	`  …`
	`  <wp:wrapTight distR="457200" … />`
	`</wp:anchor>`
	
	The distR attribute specifies that the padding distance shall be 457200 EMUs or one-half of an inch. *end example*]
	
	The possible values for this attribute are defined by the ST_WrapDistance simple type (§5.5.3.6).

wrapText (Text Wrapping Location)	Specifies how text shall wrap around the object's left and right sides.
	
	[*Example*: Consider a floating DrawingML object which shall allow text to wrap around its left side only. This setting would be specified as follows:
	
	`<wp:anchor … >`
	`  …`
	`  <wp:wrapTight wrapText="left" … />`
	`</wp:anchor>`
	
	The wrapText attribute value of `left` specifies that text shall only wrap around the let side of the object. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_WrapText simple type (§5.5.3.7).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_WrapTight">
   <sequence>
      <element name="wrapPolygon" type="CT_WrapPath" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="wrapText" type="ST_WrapText" use="required"/>
   <attribute name="distL" type="ST_WrapDistance" use="optional"/>
   <attribute name="distR" type="ST_WrapDistance" use="optional"/>
</complexType>
```

5.5.2.20 wrapTopAndBottom (Top and Bottom Wrapping)

This element specifies that text shall wrap around the top and bottom of this object, but not its left or right edges.

[*Example*: Consider a DrawingML object using top and bottom wrapping and defined as follows:

```
<wp:anchor … >
  …
  <wp:wrapTopAndBottom />
</wp:anchor>
```

The wrapTopAndBottom element specifies that text shall wrap around neither side of this object. *end example*]

Parent Elements
anchor (§5.5.2.3)

Child Elements	Subclause
effectExtent (Object Extents Including Effects)	§5.5.2.6

Attributes	Description
distB (Distance From Text on Bottom Edge)	Specifies the minimum distance which shall be maintained between the bottom edge of this drawing object and any subsequent text within the document when this graphical object is displayed within the document's contents. The distance shall be measured in EMUs (English Metric Units). [*Example*: Consider a floating DrawingML object which shall have one-half of an inch of padding between its bottom edge and the nearest text. This setting would be specified as follows:

Attributes	Description
	<wp:anchor … > … <wp:wrapTopAndBottom distB="457200" … /> </wp:anchor> The distB attribute specifies that the padding distance shall be 457200 EMUs or one-half of an inch. *end example*] The possible values for this attribute are defined by the ST_WrapDistance simple type (§5.5.3.6).
distT (Distance From Text on Top Edge)	Specifies the minimum distance which shall be maintained between the top edge of this drawing object and any subsequent text within the document when this graphical object is displayed within the document's contents. The distance shall be measured in EMUs (English Metric Units). [*Example*: Consider a floating DrawingML object which shall have one-half of an inch of padding between its top edge and the nearest text. This setting would be specified as follows: <wp:anchor … > … <wp:wrapTopAndBottom distT="457200" … /> </wp:anchor> The distT attribute specifies that the padding distance shall be 457200 EMUs or one-half of an inch. *end example*] The possible values for this attribute are defined by the ST_WrapDistance simple type (§5.5.3.6).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_WrapTopBottom">
   <sequence>
      <element name="effectExtent" type="CT_EffectExtent" minOccurs="0"/>
   </sequence>
   <attribute name="distT" type="ST_WrapDistance" use="optional"/>
   <attribute name="distB" type="ST_WrapDistance" use="optional"/>
</complexType>
```

5.5.3 Simple Types

This is the complete list of simple types in the http://schemas.openxmlformats.org/drawingml/2006/wordprocessingDrawing namespace.

5.5.3.1 ST_AlignH (Relative Horizontal Alignment Positions)

This simple type contains the possible settings specifying how a DrawingML object may be horizontally aligned relative to the horizontal alignment base defined by the parent element.

[*Example*: Consider a picture in a WordprocessingML document which has been aligned relative to the edge of the page - the left of the page horizontally, and the top of the page vertically. This alignment would be specified as follows:

```
<wp:anchor … >
  <wp:positionH relativeFrom="page">
    <wp:align>left</wp:align>
  </wp:positionH>
  …
</wp:anchor>
```

The align element with a value of `left` specifies that for the horizontal positioning defined by the parent element (in this case, positioning relative to the page), the picture shall be aligned to the left edge of the page. *end example*]

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
center (Center Alignment)	Specifies that the object shall be centered with respect to the horizontal alignment base. [*Example*: Centered on the page. *end example*]
inside (Inside)	Specifies that the object shall be inside of the horizontal alignment base. [*Example*: Inside the outside margin. *end example*]
left (Left Alignment)	Specifies that the object shall be left aligned to the horizontal alignment base. [*Example*: Left aligned relative to the margins. *end example*]
outside (Outside)	Specifies that the object shall be outside of the horizontal alignment base. [*Example*: Outside the left margin. *end example*]
right (Right Alignment)	Specifies that the object shall be right aligned to the horizontal alignment base.

Enumeration Value	Description
	[*Example*: Right aligned relative to the margins. *end example*]

Referenced By
align (§5.5.2.1)

5.5.3.2 ST_AlignV (Vertical Alignment Definition)

This simple type contains the possible settings specifying how a DrawingML object may be vertically aligned relative to the vertical alignment base defined by the parent element.

[*Example*: Consider a picture in a WordprocessingML document which has been aligned relative to the edge of the page - the left of the page horizontally, and the top of the page vertically. This alignment would be specified as follows:

```
<wp:anchor … >
  <wp:positionV relativeFrom="page">
    <wp:align>top</wp:align>
  </wp:positionH>
  …
</wp:anchor>
```

The align element with a value of top specifies that for the vertical positioning defined by the parent element (in this case, positioning relative to the page), the picture shall be aligned to the top edge of the page. *end example*]

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
bottom (Bottom)	Specifies that the object shall be at the bottom of the vertical alignment base. [*Example*: Bottom of the page. *end example*]
center (Center Alignment)	Specifies that the object shall be centered with respect to the vertical alignment base. [*Example*: Centered on the page. *end example*]
inside (Inside)	Specifies that the object shall be inside of the horizontal alignment base. [*Example*: Inside the top margin. *end example*]

Enumeration Value	Description
outside (Outside)	Specifies that the object shall be outside of the vertical alignment base. [*Example*: Outside the top margin. *end example*]
top (Top)	Specifies that the object shall be at the top of the vertical alignment base. [*Example*: Top of the page. *end example*]

Referenced By
align (§5.5.2.2)

5.5.3.3 ST_PositionOffset (Absolute Position Offset Value)

This simple type represents a one dimensional distance which shall be used to offset an objet from its base positioning location stored in EMUs.

[*Example*: Consider a DrawingML picture which shall be displayed one inch from the top of the page, and one-half of an inch from the left edge of the page. This object would be specified as follows:

```
<wp:anchor … >
  <wp:positionH relativeFrom="page">
    <wp:posOffset>914400</wp:posOffset>
  </wp:positionH>
  <wp:positionV relativeFrom="page">
    <wp:posOffset>457200</wp:posOffset>
  </wp:positionV>
</wp:anchor>
```

The posOffset element specifies the absolute positioning of the object relative to the top-left edge of the page in EMUs. *end example*]

This simple type's contents are a restriction of the XML Schema int datatype.

Referenced By
posOffset (§5.5.2.12)

5.5.3.4 ST_RelFromH (Horizontal Relative Positioning)

This simple type specifies the possible values for the base from which the relative horizontal positioning of an object shall be calculated.

[*Example*: Consider a DrawingML picture which shall be displayed at the bottom center of the page. This object would be specified as follows:

```
<wp:anchor … >
  <wp:positionH relativeFrom="page">
    <wp:align>center</wp:align>
  </wp:positionH>
  …
</wp:anchor>
```

The relativeFrom attribute specifies that the object is horizontally positioned relative to the page. *end example*]

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
character (Character)	Specifies that the horizontal positioning shall be relative to the position of the anchor within its run content.
column (Column)	Specifies that the horizontal positioning shall be relative to the extents of the column which contains its anchor.
insideMargin (Inside Margin)	Specifies that the horizontal positioning shall be relative to the inside margin of the current page (the left margin on odd pages, right on even pages).
leftMargin (Left Margin)	Specifies that the horizontal positioning shall be relative to the left margin of the page.
margin (Page Margin)	Specifies that the horizontal positioning shall be relative to the page margins.
outsideMargin (Outside Margin)	Specifies that the horizontal positioning shall be relative to the outside margin of the current page (the right margin on odd pages, left on even pages).
page (Page Edge)	Specifies that the horizontal positioning shall be relative to the edge of the page.
rightMargin (Right Margin)	Specifies that the horizontal positioning shall be relative to the right margin of the page.

Referenced By
positionH@relativeFrom (§5.5.2.10)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_RelFromH">
   <restriction base="xsd:token">
      <enumeration value="margin"/>
      <enumeration value="page"/>
      <enumeration value="column"/>
      <enumeration value="character"/>
      <enumeration value="leftMargin"/>
      <enumeration value="rightMargin"/>
      <enumeration value="insideMargin"/>
      <enumeration value="outsideMargin"/>
   </restriction>
</simpleType>
```

5.5.3.5 ST_RelFromV (Vertical Relative Positioning)

This simple type specifies the possible values for the base from which the relative vertical positioning of an object shall be calculated.

[*Example*: Consider a DrawingML picture which shall be displayed at the bottom center of the page. This object would be specified as follows:

```
<wp:anchor … >
  <wp:positionV relativeFrom="page">
    <wp:align>bottom</wp:align>
  </wp:positionV>
  …
</wp:anchor>
```

The relativeFrom attribute specifies that the object is horizontally positioned relative to the page. *end example*]

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
bottomMargin (Bottom Margin)	Specifies that the vertical positioning shall be relative to the bottom margin of the current page.
insideMargin (Inside Margin)	Specifies that the vertical positioning shall be relative to the inside margin of the current page.
line (Line)	Specifies that the vertical positioning shall be relative to the line containing the anchor character.
margin (Page Margin)	Specifies that the vertical positioning shall be relative to the page margins.
outsideMargin (Outside Margin)	Specifies that the vertical positioning shall be relative to the outside margin of the current page.

Enumeration Value	Description
page (Page Edge)	Specifies that the vertical positioning shall be relative to the edge of the page.
paragraph (Paragraph)	Specifies that the vertical positioning shall be relative to the paragraph which contains the drawing anchor.
topMargin (Top Margin)	Specifies that the vertical positioning shall be relative to the top margin of the current page.

Referenced By
positionV@relativeFrom (§5.5.2.11)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_RelFromV">
   <restriction base="xsd:token">
      <enumeration value="margin"/>
      <enumeration value="page"/>
      <enumeration value="paragraph"/>
      <enumeration value="line"/>
      <enumeration value="topMargin"/>
      <enumeration value="bottomMargin"/>
      <enumeration value="insideMargin"/>
      <enumeration value="outsideMargin"/>
   </restriction>
</simpleType>
```

5.5.3.6 ST_WrapDistance (Distance from Text)

This simple type represents a one dimensional distance which shall be used to offset an object from text, stored in EMUs.

[*Example*: Consider a floating DrawingML object which shall have one-half of an inch of padding between its left edge and the nearest text. This setting would be specified as follows:

```
<wp:anchor … >
  …
  <wp:wrapThrough distL="457200" … />
</wp:anchor>
```

The distL attribute specifies that the padding distance shall be `457200` EMUs or one-half of an inch. *end example*]

This simple type's contents are a restriction of the XML Schema unsignedInt datatype.

Referenced By
anchor@distB (§5.5.2.3); anchor@distL (§5.5.2.3); anchor@distR (§5.5.2.3); anchor@distT (§5.5.2.3);

Referenced By
inline@distB (§5.5.2.8); inline@distL (§5.5.2.8); inline@distR (§5.5.2.8); inline@distT (§5.5.2.8); wrapSquare@distB (§5.5.2.17); wrapSquare@distL (§5.5.2.17); wrapSquare@distR (§5.5.2.17); wrapSquare@distT (§5.5.2.17); wrapThrough@distL (§5.5.2.18); wrapThrough@distR (§5.5.2.18); wrapTight@distL (§5.5.2.19); wrapTight@distR (§5.5.2.19); wrapTopAndBottom@distB (§5.5.2.20); wrapTopAndBottom@distT (§5.5.2.20)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_WrapDistance">
   <restriction base="xsd:unsignedInt"/>
</simpleType>
```

5.5.3.7 ST_WrapText (Text Wrapping Location)

This simple type specifies the possible settings for how text may wrap around the object's left and right sides.

[*Example*: Consider a floating DrawingML object which shall allow text to wrap around its left side only. This setting would be specified as follows:

```
<wp:anchor … >
  …
  <wp:wrapTight wrapText="left" … />
</wp:anchor>
```

The wrapText attribute value of `left` specifies that text shall only wrap around the let side of the object. *end example*]

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
bothSides (Both Sides)	Specifies that text shall wrap around both sides of the object.
largest (Largest Side Only)	Specifies that text shall only wrap around the largest side of the object. If the object is positioned in the exact center of the page, the text shall wrap around the side on which text is first encountered: • If the first line of text intersecting the object is using left-to-right reading order, the text shall wrap to the object's left. • If the first line of text intersecting the object is using right-to-left reading order, the text shall wrap to the object's right.

Enumeration Value	Description
left (Left Side Only)	Specifies that text shall only wrap around the left side of the object.
right (Right Side Only)	Specifies that text shall only wrap around the right side of the object.

Referenced By
wrapSquare@wrapText (§5.5.2.17); wrapThrough@wrapText (§5.5.2.18); wrapTight@wrapText (§5.5.2.19)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_WrapText">
   <restriction base="xsd:token">
      <enumeration value="bothSides"/>
      <enumeration value="left"/>
      <enumeration value="right"/>
      <enumeration value="largest"/>
   </restriction>
</simpleType>
```

5.6 DrawingML - SpreadsheetML Drawing

Within a SpreadsheetML document, it is possible to include graphical DrawingML objects:

- Pictures (§5.2)
- Locked Canvases (§5.4)
- Diagrams (§5.9)
- Charts (§5.7)

When these objects are present in a spreadsheet document, it is necessary to include information which specifies how the objects shall be positioned relative to the parent worksheet. [*Example*: Whether the object is anchored to a specific row, whether it resizes with cells, and so on. *end example*]

The SpreadsheetML Drawing namespace acts in this capacity, specifying all information necessary to anchor and display DrawingML objects within a spreadsheet document.

[*Example*: Consider a DrawingML picture which shall be anchored to a specific cell for its top left and bottom right corners, resizing as those cells are relocated. This object would be specified as follows:

```
<xdr:twoCellAnchor>
  <xdr:from>
    ...
  </xdr:from>
  <xdr:to>
    ...
  </xdr:to>
```

```
    <xdr:graphicFrame>
      ...
      <a:graphic>
        <a:graphicData
           uri="http://schemas.openxmlformats.org/drawingml/2006/diagram">
          <dgm:relIds xmlns:dgm="..." xmlns:r="..." r:dm="rId1" r:lo="rId2"
             r:qs="rId3" r:cs="rId4" />
        </a:graphicData>
      </a:graphic>
    </xdr:graphicFrame>
  </xdr:twoCellAnchor>
```

The twoCellAnchor element (§5.6.2.32) specifies that this object anchored to the cells specified by the to (§5.6.2.31) and from (§5.6.2.14) elements. *end example*]

5.6.1 Table of Contents

This subclause is informative.

End of informative text.

5.6.2 Elements

The following elements define the contents of the Spreadsheet Drawing namespace:

5.6.2.1 absoluteAnchor (Absolute Anchor Shape Size)

This element is used as an anchor placeholder for a shape or group of shapes. It will anchor the object in the same position relative to sheet position and its extents are in EMU unit.

Parent Elements
wsDr (§5.6.2.34)

Child Elements	Subclause
clientData (Client Data)	§5.6.2.3
cxnSp (Connection Shape)	§5.6.2.12
ext (Shape Extent)	§5.6.2.13
graphicFrame (Graphic Frame)	§5.6.2.15
grpSp (Group Shape)	§5.6.2.16
pic (Picture)	§5.6.2.24
pos (Position)	§5.6.2.25
sp (Shape)	§5.6.2.28

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AbsoluteAnchor">
   <sequence>
      <element name="pos" type="a:CT_Point2D"/>
      <element name="ext" type="a:CT_PositiveSize2D"/>
      <group ref="EG_ObjectChoices"/>
      <element name="clientData" type="CT_AnchorClientData" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.6.2.2 blipFill (Picture Fill)

This element specifies the type of picture fill that the picture object will have. Because a picture has a picture fill already by default, it is possible to have two fills specified for a picture object. An example of this is shown below.

[Example: Consider the picture below that has a blip fill applied to it. The image used to fill this picture object has transparent pixels instead of white pixels.

```
<xdr:pic>
  ..
  <xdr:blipFill>
    <a:blip r:embed="rId2"/>
    <a:stretch>
      <a:fillRect/>
    </a:stretch>
  </xdr:blipFill>
  ..
</xdr:pic>
```

The above picture object is shown as an example of this fill type. End example]

[Example: Consider now the same picture object but with an additional gradient fill applied within the shape properties portion of the picture.

```
<xdr:pic>
  ..
  <xdr:blipFill>
    <a:blip r:embed="rId2"/>
    <a:stretch>
      <a:fillRect/>
    </a:stretch>
  </xdr:blipFill>
  <xdr:spPr>
    <a:gradFill>
      <a:gsLst>
        <a:gs pos="0">
          <a:schemeClr val="tx2">
            <a:shade val="50000"/>
          </a:schemeClr>
        </a:gs>
        <a:gs pos="39999">
          <a:schemeClr val="tx2">
            <a:tint val="20000"/>
          </a:schemeClr>
        </a:gs>
        <a:gs pos="70000">
          <a:srgbClr val="C4D6EB"/>
        </a:gs>
        <a:gs pos="100000">
          <a:schemeClr val="bg1"/>
        </a:gs>
      </a:gsLst>
    </a:gradFill>
  </xdr:spPr>
  ..
</xdr:pic>
```

The above picture object is shown as an example of this double fill type. End example]

Parent Elements
pic (§5.6.2.24)

Child Elements	Subclause
blip (Blip)	§5.1.10.13
srcRect (Source Rectangle)	§5.1.10.55
stretch (Stretch)	§5.1.10.56
tile (Tile)	§5.1.10.58

Attributes	Description
dpi (DPI Setting) Namespace: .../drawingml/2006/main	Specifies the DPI (dots per inch) used to calculate the size of the blip. If not present or zero, the DPI in the blip is used. [*Note*: This attribute is primarily used to keep track of the picture quality within a document. There are different levels of quality needed for print than on-screen viewing and thus a need to track this information. *end note*] The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
rotWithShape (Rotate With Shape) Namespace: .../drawingml/2006/main	Specifies that the fill should rotate with the shape. That is, when the shape that has been filled with a picture and the containing shape (say a rectangle) is transformed with a rotation then the fill will be transformed with the same rotation. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BlipFillProperties">
   <sequence>
      <element name="blip" type="CT_Blip" minOccurs="0" maxOccurs="1"/>
      <element name="srcRect" type="CT_RelativeRect" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_FillModeProperties" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="dpi" type="xsd:unsignedInt" use="optional"/>
   <attribute name="rotWithShape" type="xsd:boolean" use="optional"/>
</complexType>
```

5.6.2.3 clientData (Client Data)

This element is used to set certain properties related to a drawing element on the client spreadsheet application.

Parent Elements
absoluteAnchor (§5.6.2.1); oneCellAnchor (§5.6.2.23); twoCellAnchor (§5.6.2.32)

Attributes	Description
fLocksWithSheet (Locks With Sheet Flag)	This attribute indicates whether to disable selection on drawing elements when the sheet is protected. The possible values for this attribute are defined by the XML Schema boolean datatype.
fPrintsWithSheet (Prints With Sheet Flag)	This attribute indicates whether to print drawing elements when printing the sheet. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AnchorClientData">
   <attribute name="fLocksWithSheet" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="fPrintsWithSheet" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.6.2.4 cNvCxnSpPr (Non-Visual Connector Shape Drawing Properties)

This element specifies the non-visual properties for a connector shape. These are the set of properties on a shape which do not affect its display within a spreadsheet.

Parent Elements
nvCxnSpPr (§5.6.2.18)

Child Elements	Subclause
cxnSpLocks (Connection Shape Locks)	§5.1.2.1.11

Child Elements	Subclause
endCxn (Connection End)	§5.1.2.1.13
extLst (Extension List)	§5.1.2.1.15
stCxn (Connection Start)	§5.1.2.1.36

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualConnectorProperties">
   <sequence>
      <element name="cxnSpLocks" type="CT_ConnectorLocking" minOccurs="0" maxOccurs="1"/>
      <element name="stCxn" type="CT_Connection" minOccurs="0" maxOccurs="1"/>
      <element name="endCxn" type="CT_Connection" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.6.2.5 cNvGraphicFramePr (Non-Visual Graphic Frame Drawing Properties)

This element specifies the non-visual properties for a single graphical object frame within a spreadsheet. These are the set of properties of a frame which do not affect its display within a spreadsheet.

Parent Elements
nvGraphicFramePr (§5.6.2.19)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
graphicFrameLocks (Graphic Frame Locks)	§5.1.2.1.19

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualGraphicFrameProperties">
   <sequence>
      <element name="graphicFrameLocks" type="CT_GraphicalObjectFrameLocking" minOccurs="0"
        maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.6.2.6 cNvGrpSpPr (Non-Visual Group Shape Drawing Properties)

This element specifies the non-visual properties of a hierarchical grouping of shapes, graphical object frames, and child groups. These are the set of properties of a group which do not affect its display within a spreadsheet.

Parent Elements
nvGrpSpPr (§5.6.2.20)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
grpSpLocks (Group Shape Locks)	§5.1.2.1.21

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualGroupDrawingShapeProps">
   <sequence>
      <element name="grpSpLocks" type="CT_GroupLocking" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.6.2.7 cNvPicPr (Non-Visual Picture Drawing Properties)

This element describes the non-visual properties of a picture within a spreadsheet. These are the set of properties of a picture which do not affect its display within a spreadsheet.

[Example: Consider the following SpreadsheetDrawingML.

```
<xdr:pic>
  ..
  <xdr:nvPicPr>
    <xdr:cNvPr id="4" name="Lilly_by_Lisher.jpg"/>
    <xdr:cNvPicPr>
      <a:picLocks noChangeAspect="1"/>
    </xdr:cNvPicPr>
    <xdr:nvPr/>
  </xdr:nvPicPr>
  ..
</xdr:pic>
```

The above example defines some non-visual picture drawing properties for the inserted picture. End example]

Parent Elements
nvPicPr (§5.6.2.21)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
picLocks (Picture Locks)	§5.1.2.1.31

Attributes	Description
preferRelativeResize (Relative Resize	Specifies if the user interface should show the resizing of the picture based on the picture's current size or its original size. If this attribute is set to true, then scaling will be

Attributes	Description
Preferred) Namespace: .../drawingml/2006/main	relative to the original picture size as opposed to the current picture size. [*Example*: Consider the case where a picture has been resized within a document and is now 50% of the originally inserted picture size. Now if the user chooses to make a later adjustment to the size of this picture within the generating application, then the value of this attribute should be checked. If this attribute is set to true then a value of 50% will be shown. Similarly, if this attribute is set to false, then a value of 100% should be shown because the picture has not yet been resized from its current (smaller) size. *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualPictureProperties">
   <sequence>
      <element name="picLocks" type="CT_PictureLocking" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="preferRelativeResize" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.6.2.8 cNvPr (Non-Visual Drawing Properties)

This element specifies the set of non-visual properties for the parent element. These properties specify all the data about the parent which does not affect its display within the spreadsheet.

[Example: Consider the following SpreadSheetDrawingML.

```
<xdr:pic>
  ..
  <xdr:nvPicPr>
    <xdr:cNvPr id="4" name="Lilly_by_Lisher.jpg"/>
  </xdr:nvPicPr>
  ..
</xdr:pic>
```

The above example defines some non-visual drawing properties for the inserted picture. End example]

Parent Elements
nvCxnSpPr (§5.6.2.18); nvGraphicFramePr (§5.6.2.19); nvGrpSpPr (§5.6.2.20); nvPicPr (§5.6.2.21); nvSpPr (§5.6.2.22)

Child Elements	Subclause

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
hlinkClick (Click Hyperlink)	§5.1.5.3.5
hlinkHover (Hyperlink for Hover)	§5.1.2.1.23

Attributes	Description
descr (Alternative Text for Object) Namespace: .../drawingml/2006/main	Specifies alternative text for the current DrawingML object, for use by assistive technologies or applications which will not display the current object. If this element is omitted, then no alternative text is present for the parent object. [*Example*: Consider a DrawingML object defined as follows: `<… descr="A picture of a bowl of fruit">` The descr attribute contains alternative text which may be used in place of the actual DrawingML object. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
hidden (Hidden) Namespace: .../drawingml/2006/main	Specifies whether this DrawingML object shall be displayed. When a DrawingML object is displayed within a document, that object may be hidden (i.e., present, but not visible). This attribute shall determine whether the object shall be rendered or made hidden. [*Note*: An application may have settings which allow this object to be viewed. *end note*] If this attribute is omitted, then the parent DrawingML object shall be displayed (i.e., not hidden). [*Example*: Consider an inline DrawingML object which shall be hidden within the document's content. This setting would be specified as follows: `<… hidden="true" />` The hidden attribute has a value of `true`, which specifies that the DrawingML object is hidden and not displayed when the document is displayed. *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
id (Unique Identifier) Namespace: .../drawingml/2006/main	Specifies a unique identifier for the current DrawingML object within the current document. This ID may be used to assist in uniquely identifying this object so that it can be referred to by other parts of the document. If multiple objects within the same document share the same id attribute value, then the document shall be considered non-conformant. [*Example*: Consider a DrawingML object defined as follows:

Attributes	Description
	<… id="10" … > The id attribute has a value of 10, which is the unique identifier for this DrawingML object. *end example*] The possible values for this attribute are defined by the ST_DrawingElementId simple type (§5.1.12.19).
name (Name) Namespace: …/drawingml/2006/main	Specifies the name of the object. [*Note*: Typically, this will be used to store the original file name of a picture object. *end note*] [*Example*: Consider a DrawingML object defined as follows: < … name="foo.jpg" > The name attribute has a value of foo.jpg, which is the name of this DrawingML object. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualDrawingProps">
   <sequence>
      <element name="hlinkClick" type="CT_Hyperlink" minOccurs="0" maxOccurs="1"/>
      <element name="hlinkHover" type="CT_Hyperlink" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="id" type="ST_DrawingElementId" use="required"/>
   <attribute name="name" type="xsd:string" use="required"/>
   <attribute name="descr" type="xsd:string" use="optional" default=""/>
   <attribute name="hidden" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.6.2.9 cNvSpPr (Connection Non-Visual Shape Properties)

This element specifies the set of non-visual properties for a connection shape. These properties specify all data about the connection shape which do not affect its display within a spreadsheet.

[Example: Consider the shape that has a shape lock applied to it.

```
<xdr:sp>
  <xdr:nvSpPr>
    <xdr:cNvPr id="2" name="Rectangle 1"/>
    <xdr:cNvSpPr>
      <a:spLocks noGrp="1"/>
    </xdr:cNvSpPr>
  </xdr:nvSpPr>
```

```
    ..
</xdr:sp>
```

This shape lock is stored within the non-visual drawing properties for this shape. End example]

Parent Elements
nvSpPr (§5.6.2.22)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
spLocks (Shape Locks)	§5.1.2.1.34

Attributes	Description
txBox (Text Box) Namespace: .../drawingml/2006/main	Specifies that the corresponding shape is a text box and thus should be treated as such by the generating application. If this attribute is omitted then it is assumed that the corresponding shape is not specifically a text box. [*Note*: Because a shape is not specified to be a text box does not mean that it cannot have text attached to it. A text box is merely a specialized shape with specific properties. *end note*] The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualDrawingShapeProps">
   <sequence>
      <element name="spLocks" type="CT_ShapeLocking" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="txBox" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.6.2.10 col (Column))

This element specifies the column that will be used within the from and to elements to specify anchoring information for a shape within a spreadsheet

The possible values for this element are defined by the ST_ColID simple type (§5.6.3.1).

Parent Elements
from (§5.6.2.14); to (§5.6.2.31)

5.6.2.11 colOff (Column Offset)

This element is used to specify the column offset within a cell. The units for which this attribute is specified in reside within the simple type definition referenced below.

The possible values for this element are defined by the ST_Coordinate simple type (§5.1.12.16).

Parent Elements
from (§5.6.2.14); to (§5.6.2.31)

5.6.2.12 cxnSp (Connection Shape)

This element specifies the properties for a connection shape drawing element. A connection shape is a line, etc. that connects two other shapes in this drawing.

Parent Elements
absoluteAnchor (§5.6.2.1); grpSp (§5.6.2.16); oneCellAnchor (§5.6.2.23); twoCellAnchor (§5.6.2.32)

Child Elements	Subclause
nvCxnSpPr (Non-Visual Properties for a Connection Shape)	§5.6.2.18
spPr (Shape Properties)	§5.6.2.29
style (Shape Style)	§5.6.2.30

Attributes	Description
fPublished (Publish to Server Flag)	This attribute indicates whether the shape shall be published with the worksheet when sent to the server. The possible values for this attribute are defined by the XML Schema boolean datatype.
macro (Reference to Custom Function)	This element specifies the custom function associated with the object. [*Example*: A macro script, add-in function, and so on. *end example*] The format of this string shall be application-defined, and should be ignored if not understood. [*Example*: `< ... macro="DoWork()" />` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Connector">
   <sequence>
      <element name="nvCxnSpPr" type="CT_ConnectorNonVisual" minOccurs="1" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="1" maxOccurs="1"/>
      <element name="style" type="a:CT_ShapeStyle" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="macro" type="xsd:string" use="optional"/>
   <attribute name="fPublished" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.6.2.13 ext (Shape Extent)

This element describes the length and width properties for how far a drawing element should extend for.

Parent Elements
absoluteAnchor (§5.6.2.1); oneCellAnchor (§5.6.2.23)

Attributes	Description
cx (Extent Length) Namespace: .../drawingml/2006/main	Specifies the length of the extents rectangle in EMUs. This rectangle shall dictate the size of the object as displayed (the result of any scaling to the original object). [*Example*: Consider a DrawingML object specified as follows: `<… cx="1828800" cy="200000"/>` The cx attributes specifies that this object has a height of `1828800` EMUs (English Metric Units). *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).
cy (Extent Width) Namespace: .../drawingml/2006/main	Specifies the width of the extents rectangle in EMUs. This rectangle shall dictate the size of the object as displayed (the result of any scaling to the original object). [*Example*: Consider a DrawingML object specified as follows: `< … cx="1828800" cy="200000"/>` The cy attribute specifies that this object has a width of `200000` EMUs (English Metric Units). *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PositiveSize2D">
   <attribute name="cx" type="ST_PositiveCoordinate" use="required"/>
   <attribute name="cy" type="ST_PositiveCoordinate" use="required"/>
</complexType>
```

5.6.2.14 from (Starting Anchor Point)

This element specifies the first anchor point for the drawing element. This will be used to anchor the top and left sides of the shape within the spreadsheet. That is when the cell that is specified in the from element is adjusted, the shape will also be adjusted.

[Example: Consider the following SpreadsheetDrawingML

```
<xdr:twoCellAnchor>
  <xdr:from>
    <xdr:col>3</xdr:col>
    <xdr:colOff>447675</xdr:colOff>
    <xdr:row>8</xdr:row>
    <xdr:rowOff>28575</xdr:rowOff>
  </xdr:from>
  <xdr:to>
    <xdr:col>5</xdr:col>
    <xdr:colOff>466725</xdr:colOff>
    <xdr:row>14</xdr:row>
    <xdr:rowOff>9525</xdr:rowOff>
  </xdr:to>
  <xdr:sp macro="" textlink="">
  ...
  </xdr:sp>
  <xdr:clientData/>
</xdr:twoCellAnchor>
```

The above example shows the first anchor point being specified via the from element. End example]

Parent Elements
oneCellAnchor (§5.6.2.23); twoCellAnchor (§5.6.2.32)

Child Elements	Subclause
col (Column))	§5.6.2.10
colOff (Column Offset)	§5.6.2.11
row (Row)	§5.6.2.26
rowOff (Row Offset)	§5.6.2.27

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Marker">
   <sequence>
      <element name="col" type="ST_ColID"/>
      <element name="colOff" type="a:ST_Coordinate"/>
      <element name="row" type="ST_RowID"/>
      <element name="rowOff" type="a:ST_Coordinate"/>
   </sequence>
</complexType>
```

5.6.2.15 graphicFrame (Graphic Frame)

This element describes a single graphical object frame for a spreadsheet which contains a graphical object.

Parent Elements
absoluteAnchor (§5.6.2.1); grpSp (§5.6.2.16); oneCellAnchor (§5.6.2.23); twoCellAnchor (§5.6.2.32)

Child Elements	Subclause
graphic (Graphic Object)	§5.1.2.1.16
nvGraphicFramePr (Non-Visual Properties for a Graphic Frame)	§5.6.2.19
xfrm (2D Transform for Graphic Frames)	§5.6.2.35

Attributes	Description
fPublished (Publish to Server Flag)	This attribute indicates whether the shape shall be published with the worksheet when sent to the server. The possible values for this attribute are defined by the XML Schema boolean datatype.
macro (Reference To Custom Function)	This element specifies the custom function associated with the object. [*Example*: A macro script, add-in function, and so on. *end example*] The format of this string shall be application-defined, and should be ignored if not understood. [*Example*: `< ... macro="DoWork()" />` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GraphicalObjectFrame">
   <sequence>
      <element name="nvGraphicFramePr" type="CT_GraphicalObjectFrameNonVisual" minOccurs="1"
        maxOccurs="1"/>
      <element name="xfrm" type="a:CT_Transform2D" minOccurs="1" maxOccurs="1"/>
      <element ref="a:graphic" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="macro" type="xsd:string" use="optional"/>
   <attribute name="fPublished" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.6.2.16 grpSp (Group Shape)

This element specifies a group shape that represents many shapes grouped together. This shape is to be treated just as if it were a regular shape but instead of being described by a single geometry it is made up of all the shape geometries encompassed within it. Within a group shape each of the shapes that make up the group are specified just as they normally would. The idea behind grouping elements however is that a single transform can apply to many shapes at the same time.

[Example: Consider the following group shape.

```
<xdr:grpSp>
  <xdr:nvGrpSpPr>
    <xdr:cNvPr id="10" name="Group 9"/>
    <xdr:cNvGrpSpPr/>
    <xdr:nvPr/>
  </xdr:nvGrpSpPr>
  <xdr:grpSpPr>
    <a:xfrm>
      <a:off x="838200" y="990600"/>
      <a:ext cx="2426208" cy="978408"/>
      <a:chOff x="838200" y="990600"/>
      <a:chExt cx="2426208" cy="978408"/>
   </a:xfrm>
  </xdr:grpSpPr>
  <xdr:sp>
  ..
  </xdr:sp>
  <xdr:sp>
  ..
  </xdr:sp>
  <xdr:sp>
  ..
  </xdr:sp>
</xdr:grpSp>
```

In the above example we see three shapes specified within a single group. These three shapes have their position and sizes specified just as they normally would within the shape tree. The generating application should apply the transformation after the bounding box for the group shape has been calculated. End example]

Parent Elements
absoluteAnchor (§5.6.2.1); grpSp (§5.6.2.16); oneCellAnchor (§5.6.2.23); twoCellAnchor (§5.6.2.32)

Child Elements	Subclause
cxnSp (Connection Shape)	§5.6.2.12
graphicFrame (Graphic Frame)	§5.6.2.15
grpSp (Group Shape)	§5.6.2.16
grpSpPr (Group Shape Properties)	§5.6.2.17
nvGrpSpPr (Non-Visual Properties for a Group Shape)	§5.6.2.20
pic (Picture)	§5.6.2.24
sp (Shape)	§5.6.2.28

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GroupShape">
   <sequence>
      <element name="nvGrpSpPr" type="CT_GroupShapeNonVisual" minOccurs="1" maxOccurs="1"/>
      <element name="grpSpPr" type="a:CT_GroupShapeProperties" minOccurs="1" maxOccurs="1"/>
      <choice minOccurs="0" maxOccurs="unbounded">
         <element name="sp" type="CT_Shape"/>
         <element name="grpSp" type="CT_GroupShape"/>
         <element name="graphicFrame" type="CT_GraphicalObjectFrame"/>
         <element name="cxnSp" type="CT_Connector"/>
         <element name="pic" type="CT_Picture"/>
      </choice>
   </sequence>
</complexType>
```

5.6.2.17 grpSpPr (Group Shape Properties)

This element specifies the properties that are to be common across all of the shapes within the corresponding group. If there are any conflicting properties within the group shape properties and the individual shape properties then the individual shape properties should take precedence.

Parent Elements
grpSp (§5.6.2.16)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14

Child Elements	Subclause
effectDag (Effect Container)	§5.1.10.25
effectLst (Effect Container)	§5.1.10.26
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
scene3d (3D Scene Properties)	§5.1.4.1.26
solidFill (Solid Fill)	§5.1.10.54
xfrm (2D Transform for Grouped Objects)	§5.1.9.5

Attributes	Description
bwMode (Black and White Mode) Namespace: .../drawingml/2006/main	Specifies that the group shape should be rendered using only black and white coloring. That is the coloring information for the group shape should be converted to either black or white when rendering the corresponding shapes. No gray is to be used in rendering this image, only stark black and stark white. [*Note*: This does not mean that the group shapes themselves are stored with only black and white color information. This attribute instead sets the rendering mode that the shapes will use when rendering. *end note*] The possible values for this attribute are defined by the ST_BlackWhiteMode simple type (§5.1.12.10).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GroupShapeProperties">
   <sequence>
      <element name="xfrm" type="CT_GroupTransform2D" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_FillProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_EffectProperties" minOccurs="0" maxOccurs="1"/>
      <element name="scene3d" type="CT_Scene3D" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="bwMode" type="ST_BlackWhiteMode" use="optional"/>
</complexType>
```

5.6.2.18 nvCxnSpPr (Non-Visual Properties for a Connection Shape)

This element specifies all non-visual properties for a connection shape. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a

connection shape. This allows for additional information that does not affect the appearance of the connection shape to be stored.

Parent Elements
cxnSp (§5.6.2.12)

Child Elements	Subclause
cNvCxnSpPr (Non-Visual Connector Shape Drawing Properties)	§5.6.2.4
cNvPr (Non-Visual Drawing Properties)	§5.6.2.8

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ConnectorNonVisual">
   <sequence>
      <element name="cNvPr" type="a:CT_NonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
      <element name="cNvCxnSpPr" type="a:CT_NonVisualConnectorProperties" minOccurs="1"
        maxOccurs="1"/>
   </sequence>
</complexType>
```

5.6.2.19 nvGraphicFramePr (Non-Visual Properties for a Graphic Frame)

This element specifies all non-visual properties for a graphic frame. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a graphic frame. This allows for additional information that does not affect the appearance of the graphic frame to be stored.

Parent Elements
graphicFrame (§5.6.2.15)

Child Elements	Subclause
cNvGraphicFramePr (Non-Visual Graphic Frame Drawing Properties)	§5.6.2.5
cNvPr (Non-Visual Drawing Properties)	§5.6.2.8

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GraphicalObjectFrameNonVisual">
   <sequence>
      <element name="cNvPr" type="a:CT_NonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
      <element name="cNvGraphicFramePr" type="a:CT_NonVisualGraphicFrameProperties" minOccurs="1"
        maxOccurs="1"/>
   </sequence>
</complexType>
```

5.6.2.20 nvGrpSpPr (Non-Visual Properties for a Group Shape)

This element specifies all non-visual properties for a group shape. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a group shape. This allows for additional information that does not affect the appearance of the group shape to be stored.

Parent Elements
grpSp (§5.6.2.16)

Child Elements	Subclause
cNvGrpSpPr (Non-Visual Group Shape Drawing Properties)	§5.6.2.6
cNvPr (Non-Visual Drawing Properties)	§5.6.2.8

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GroupShapeNonVisual">
   <sequence>
      <element name="cNvPr" type="a:CT_NonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
      <element name="cNvGrpSpPr" type="a:CT_NonVisualGroupDrawingShapeProps" minOccurs="1"
        maxOccurs="1"/>
   </sequence>
</complexType>
```

5.6.2.21 nvPicPr (Non-Visual Properties for a Picture)

This element specifies all non-visual properties for a picture. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a picture. This allows for additional information that does not affect the appearance of the picture to be stored.

[Example: Consider the following SpreadsheetDrawingML.

```
<xdr:pic>
  ..
  <xdr:nvPicPr>
    ..
  </xdr:nvPicPr>
  ..
</xdr:pic>
```

The above example shows the defining of non-visual picture properties. End example]

Parent Elements
pic (§5.6.2.24)

Child Elements	Subclause
cNvPicPr (Non-Visual Picture Drawing Properties)	§5.6.2.7
cNvPr (Non-Visual Drawing Properties)	§5.6.2.8

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PictureNonVisual">
   <sequence>
      <element name="cNvPr" type="a:CT_NonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
      <element name="cNvPicPr" type="a:CT_NonVisualPictureProperties" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.6.2.22 nvSpPr (Non-Visual Properties for a Shape)

This element specifies all non-visual properties for a shape. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a shape. This allows for additional information that does not affect the appearance of the shape to be stored.

Parent Elements
sp (§5.6.2.28)

Child Elements	Subclause
cNvPr (Non-Visual Drawing Properties)	§5.6.2.8
cNvSpPr (Connection Non-Visual Shape Properties)	§5.6.2.9

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShapeNonVisual">
   <sequence>
      <element name="cNvPr" type="a:CT_NonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
      <element name="cNvSpPr" type="a:CT_NonVisualDrawingShapeProps" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.6.2.23 oneCellAnchor (One Cell Anchor Shape Size)

This element specifies a one cell anchor placeholder for a group, a shape, or a drawing element. It moves with the cell and its extents is in EMU units.

Parent Elements
wsDr (§5.6.2.34)

Child Elements	Subclause
clientData (Client Data)	§5.6.2.3

Child Elements	Subclause
cxnSp (Connection Shape)	§5.6.2.12
ext (Shape Extent)	§5.6.2.13
from (Starting Anchor Point)	§5.6.2.14
graphicFrame (Graphic Frame)	§5.6.2.15
grpSp (Group Shape)	§5.6.2.16
pic (Picture)	§5.6.2.24
sp (Shape)	§5.6.2.28

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OneCellAnchor">
   <sequence>
      <element name="from" type="CT_Marker"/>
      <element name="ext" type="a:CT_PositiveSize2D"/>
      <group ref="EG_ObjectChoices"/>
      <element name="clientData" type="CT_AnchorClientData" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.6.2.24 pic (Picture)

This element specifies the existence of a picture object within the spreadsheet.

[Example: Consider the following SpreadsheetDrawingML that specifies the existence of a picture within a document. This picture can have non-visual properties, a picture fill as well as shape properties attached to it.

```
<xdr:pic>
  <xdr:nvPicPr>
    <xdr:cNvPr id="4" name="lake.JPG" descr="Picture of a Lake" />
    <xdr:cNvPicPr>
      <a:picLocks noChangeAspect="1"/>
    </xdr:cNvPicPr>
    <xdr:nvPr/>
  </xdr:nvPicPr>
  <xdr:blipFill>
  ...
  </xdr:blipFill>
  <xdr:spPr>
  ...
  </xdr:spPr>
</xdr:pic>
```

End example]

Parent Elements

Parent Elements
absoluteAnchor (§5.6.2.1); grpSp (§5.6.2.16); oneCellAnchor (§5.6.2.23); twoCellAnchor (§5.6.2.32)

Child Elements	Subclause
blipFill (Picture Fill)	§5.6.2.2
nvPicPr (Non-Visual Properties for a Picture)	§5.6.2.21
spPr (Shape Properties)	§5.6.2.29
style (Shape Style)	§5.6.2.30

Attributes	Description
fPublished (Publish to Server Flag)	This attribute indicates whether the shape shall be published with the worksheet when sent to the server. The possible values for this attribute are defined by the XML Schema boolean datatype.
macro (Reference To Custom Function)	This element specifies the custom function associated with the object. [*Example*: A macro script, add-in function, and so on. *end example*] The format of this string shall be application-defined, and should be ignored if not understood. [*Example*: `< ... macro="DoWork()" />` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Picture">
   <sequence>
      <element name="nvPicPr" type="CT_PictureNonVisual" minOccurs="1" maxOccurs="1"/>
      <element name="blipFill" type="a:CT_BlipFillProperties" minOccurs="1" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="1" maxOccurs="1"/>
      <element name="style" type="a:CT_ShapeStyle" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="macro" type="xsd:string" use="optional" default=""/>
   <attribute name="fPublished" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.6.2.25 pos (Position)

This element describes the position of a drawing element within a spreadsheet.

Parent Elements
absoluteAnchor (§5.6.2.1)

Attributes	Description
x (X-Axis Coordinate) Namespace: .../drawingml/2006/main	Specifies a coordinate on the x-axis. The origin point for this coordinate shall be specified by the parent XML element. [*Example*: Consider the following point on a basic wrapping polygon for a DrawingML object: `<wp:… x="0" y="100" />` The x attribute defines an x-coordinate of `0`. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).
y (Y-Axis Coordinate) Namespace: .../drawingml/2006/main	Specifies a coordinate on the x-axis. The origin point for this coordinate shall be specified by the parent XML element. [*Example*: Consider the following point on a basic wrapping polygon for a DrawingML object: `<wp:… x="0" y="100" />` The y attribute defines a y-coordinate of `100`. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Point2D">
   <attribute name="x" type="ST_Coordinate" use="required"/>
   <attribute name="y" type="ST_Coordinate" use="required"/>
</complexType>
```

5.6.2.26 row (Row)

This element specifies the row that will be used within the from and to elements to specify anchoring information for a shape within a spreadsheet.

The possible values for this element are defined by the ST_RowID simple type (§5.6.3.3).

Parent Elements
from (§5.6.2.14); to (§5.6.2.31)

5.6.2.27 rowOff (Row Offset)

This element is used to specify the row offset within a cell. The units for which this attribute is specified in reside within the simple type definition referenced below.

The possible values for this element are defined by the ST_Coordinate simple type (§5.1.12.16).

Parent Elements
from (§5.6.2.14); to (§5.6.2.31)

5.6.2.28 sp (Shape)

This element specifies the existence of a single shape. A shape can either be a preset or a custom geometry, defined using the SpreadsheetDrawingML framework. In addition to a geometry each shape can have both visual and non-visual properties attached. Text and corresponding styling information can also be attached to a shape. This shape is specified along with all other shapes within either the shape tree or group shape elements.

Parent Elements
absoluteAnchor (§5.6.2.1); grpSp (§5.6.2.16); oneCellAnchor (§5.6.2.23); twoCellAnchor (§5.6.2.32)

Child Elements	Subclause
nvSpPr (Non-Visual Properties for a Shape)	§5.6.2.22
spPr (Shape Properties)	§5.6.2.29
style (Shape Style)	§5.6.2.30
txBody (Shape Text Body)	§5.6.2.33

Attributes	Description
fLocksText (Lock Text Flag)	This attribute indicates whether to allow text editing within this drawing object when the parent worksheet is protected. The possible values for this attribute are defined by the XML Schema boolean datatype.
fPublished (Publish to Server Flag)	This attribute indicates whether the shape shall be published with the worksheet when sent to the server. The possible values for this attribute are defined by the XML Schema boolean datatype.
macro (Reference to Custom Function)	This element specifies the custom function associated with the object. [*Example*: A macro script, add-in function, and so on. *end example*] The format of this string shall be application-defined, and should be ignored if not understood. [*Example*:

Attributes	Description
	< ... macro="DoWork()" /> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
textlink (Text Link)	This attribute specifies a formula linking to spreadsheet cell data. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Shape">
   <sequence>
      <element name="nvSpPr" type="CT_ShapeNonVisual" minOccurs="1" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="1" maxOccurs="1"/>
      <element name="style" type="a:CT_ShapeStyle" minOccurs="0" maxOccurs="1"/>
      <element name="txBody" type="a:CT_TextBody" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="macro" type="xsd:string" use="optional"/>
   <attribute name="textlink" type="xsd:string" use="optional"/>
   <attribute name="fLocksText" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="fPublished" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.6.2.29 spPr (Shape Properties)

This element specifies the visual shape properties that can be applied to a special shape such as a connector shape or picture. These are the same properties that are allowed to describe the visual properties of a shape but are used here to describe additional object-specific properties within a document. This allows for these shapes to have both the properties of a shape as well as specific properties that are unique to only them.

Parent Elements
cxnSp (§5.6.2.12); pic (§5.6.2.24); sp (§5.6.2.28)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
custGeom (Custom Geometry)	§5.1.11.8
effectDag (Effect Container)	§5.1.10.25
effectLst (Effect Container)	§5.1.10.26
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35

Child Elements	Subclause
ln (Outline)	§5.1.2.1.24
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
prstGeom (Preset geometry)	§5.1.11.18
scene3d (3D Scene Properties)	§5.1.4.1.26
solidFill (Solid Fill)	§5.1.10.54
sp3d (Apply 3D shape properties)	§5.1.7.12
xfrm (2D Transform for Individual Objects)	§5.1.9.6

Attributes	Description
bwMode (Black and White Mode) Namespace: .../drawingml/2006/main	Specifies that the picture should be rendered using only black and white coloring. That is the coloring information for the picture should be converted to either black or white when rendering the picture. No gray is to be used in rendering this image, only stark black and stark white. [*Note*: This does not mean that the picture itself that is stored within the file is necessarily a black and white picture. This attribute instead sets the rendering mode that the picture will have applied to when rendering. *end note*] The possible values for this attribute are defined by the ST_BlackWhiteMode simple type (§5.1.12.10).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShapeProperties">
   <sequence>
      <element name="xfrm" type="CT_Transform2D" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_Geometry" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_FillProperties" minOccurs="0" maxOccurs="1"/>
      <element name="ln" type="CT_LineProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_EffectProperties" minOccurs="0" maxOccurs="1"/>
      <element name="scene3d" type="CT_Scene3D" minOccurs="0" maxOccurs="1"/>
      <element name="sp3d" type="CT_Shape3D" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="bwMode" type="ST_BlackWhiteMode" use="optional"/>
</complexType>
```

5.6.2.30 style (Shape Style)

The element specifies the style that will be applied to a shape and the corresponding references for each of the style components such as lines and fills.

Parent Elements
cxnSp (§5.6.2.12); pic (§5.6.2.24); sp (§5.6.2.28)

Child Elements	Subclause
effectRef (Effect Reference)	§5.1.4.2.8
fillRef (Fill Reference)	§5.1.4.2.10
fontRef (Font Reference)	§5.1.4.1.17
lnRef (Line Reference)	§5.1.4.2.19

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShapeStyle">
   <sequence>
      <element name="lnRef" type="CT_StyleMatrixReference" minOccurs="1" maxOccurs="1"/>
      <element name="fillRef" type="CT_StyleMatrixReference" minOccurs="1" maxOccurs="1"/>
      <element name="effectRef" type="CT_StyleMatrixReference" minOccurs="1" maxOccurs="1"/>
      <element name="fontRef" type="CT_FontReference" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.6.2.31 to (Ending Anchor Point)

This element specifies the second anchor point for the drawing element. This will be used to anchor the bottom and right sides of the shape within the spreadsheet. That is when the cell that is specified in the to element is adjusted, the shape will also be adjusted.

[Example: Consider the following SpreadsheetDrawingML

```
<xdr:twoCellAnchor>
  <xdr:from>
    <xdr:col>3</xdr:col>
    <xdr:colOff>447675</xdr:colOff>
    <xdr:row>8</xdr:row>
    <xdr:rowOff>28575</xdr:rowOff>
  </xdr:from>
  <xdr:to>
    <xdr:col>5</xdr:col>
    <xdr:colOff>466725</xdr:colOff>
    <xdr:row>14</xdr:row>
    <xdr:rowOff>9525</xdr:rowOff>
  </xdr:to>
  <xdr:sp macro="" textlink="">
  ...
  </xdr:sp>
  <xdr:clientData/>
```

```
</xdr:twoCellAnchor>
```

The above example shows the second anchor point being specified via the to element. End example]

Parent Elements
twoCellAnchor (§5.6.2.32)

Child Elements	Subclause
col (Column))	§5.6.2.10
colOff (Column Offset)	§5.6.2.11
row (Row)	§5.6.2.26
rowOff (Row Offset)	§5.6.2.27

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Marker">
   <sequence>
      <element name="col" type="ST_ColID"/>
      <element name="colOff" type="a:ST_Coordinate"/>
      <element name="row" type="ST_RowID"/>
      <element name="rowOff" type="a:ST_Coordinate"/>
   </sequence>
</complexType>
```

5.6.2.32 twoCellAnchor (Two Cell Anchor Shape Size)

This element specifies a two cell anchor placeholder for a group, a shape, or a drawing element. It moves with cells and its extents are in EMU units.

Parent Elements
wsDr (§5.6.2.34)

Child Elements	Subclause
clientData (Client Data)	§5.6.2.3
cxnSp (Connection Shape)	§5.6.2.12
from (Starting Anchor Point)	§5.6.2.14
graphicFrame (Graphic Frame)	§5.6.2.15
grpSp (Group Shape)	§5.6.2.16
pic (Picture)	§5.6.2.24
sp (Shape)	§5.6.2.28
to (Ending Anchor Point)	§5.6.2.31

Attributes	Description

editAs (Positioning and Resizing Behaviors)	Specifies how the DrawingML contents shall be moved and/or resized when the rows and columns between its start and ending anchor (the from and to child elements) are resized, or have additional rows/columns inserted within them, or additional row/columns are added before them. The behaviors are discussed in the simple type referenced below.
	
	If this attribute is omitted, then its default value shall be assumed to be `twoCell`.
	
	[*Example*: Consider a drawing defined as follows:
	
	`<ws:twoCellAnchor editAs="absolute">`
	`...`
	`</ws:twoCellAnchor>`
	
	The editAs attribute has a value of `absolute`, which specifies that the sizing of this object shall not change, instead the anchor locations should be moved as needed to maintain the same size and position. *end example*]
	
	The possible values for this attribute are defined by the ST_EditAs simple type (§5.6.3.2).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TwoCellAnchor">
   <sequence>
      <element name="from" type="CT_Marker"/>
      <element name="to" type="CT_Marker"/>
      <group ref="EG_ObjectChoices"/>
      <element name="clientData" type="CT_AnchorClientData" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="editAs" type="ST_EditAs" use="optional" default="twoCell"/>
</complexType>
```

5.6.2.33 txBody (Shape Text Body)

This element specifies the existence of text to be contained within the corresponding shape. All visible text and visible text related properties are contained within this element. There can be multiple paragraphs and within paragraphs multiple runs of text.

Parent Elements
sp (§5.6.2.28)

Child Elements	Subclause
bodyPr (Body Properties)	§5.1.5.1.1
lstStyle (Text List Styles)	§5.1.5.4.12
p (Text Paragraphs)	§5.1.5.2.6

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextBody">
   <sequence>
      <element name="bodyPr" type="CT_TextBodyProperties" minOccurs="1" maxOccurs="1"/>
      <element name="lstStyle" type="CT_TextListStyle" minOccurs="0" maxOccurs="1"/>
      <element name="p" type="CT_TextParagraph" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.6.2.34 wsDr (Worksheet Drawing)

This element specifies all drawing objects within the worksheet. It acts much like the spTree element within the DrawingML framework. Allowing for the specification of all shapes for a given part of a document, in this case a single Worksheet.

Parent Elements
Root element of SpreadsheetML Drawing part

Child Elements	Subclause
absoluteAnchor (Absolute Anchor Shape Size)	§5.6.2.1
oneCellAnchor (One Cell Anchor Shape Size)	§5.6.2.23
twoCellAnchor (Two Cell Anchor Shape Size)	§5.6.2.32

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Drawing">
   <sequence>
      <group ref="EG_Anchor" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.6.2.35 xfrm (2D Transform for Graphic Frames)

This element specifies a two dimensional transform for a Graphic Frame.

Parent Elements
graphicFrame (§5.6.2.15)

Child Elements	Subclause
ext (Extents)	§5.1.9.3
off (Offset)	§5.1.9.4

Attributes	Description

Attributes	Description
flipH (Horizontal Flip) Namespace: .../drawingml/2006/main	Specifies a horizontal flip. When true, this attribute defines that the shape will be flipped horizontally about the center of its bounding box. [*Example*: The following illustrates the effect of a horizontal flip.  *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
flipV (Vertical Flip) Namespace: .../drawingml/2006/main	Specifies a vertical flip. When true, this attribute defines that the group will be flipped vertically about the center of its bounding box. [*Example*: The following illustrates the effect of a vertical flip.  *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
rot (Rotation) Namespace: .../drawingml/2006/main	Specifies the rotation of the Graphic Frame. The units for which this attribute is specified in reside within the simple type definition referenced below. The possible values for this attribute are defined by the ST_Angle simple type (§5.1.12.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Transform2D">
   <sequence>
      <element name="off" type="CT_Point2D" minOccurs="0" maxOccurs="1"/>
      <element name="ext" type="CT_PositiveSize2D" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="rot" type="ST_Angle" use="optional" default="0"/>
   <attribute name="flipH" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="flipV" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.6.3 Simple Types

This is the complete list of simple types in the http://schemas.openxmlformats.org/drawingml/2006/spreadsheetDrawing namespace.

5.6.3.1 ST_ColID (Column ID)

This type specifies a column identification. The numerical value used for the column id should be non-negative and never exceed the number of total columns within the spreadsheet document.

This simple type's contents are a restriction of the XML Schema int datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 0.

Referenced By
col (§5.6.2.10)

5.6.3.2 ST_EditAs (Resizing Behaviors)

This simple type specifies all possible settings for how DrawingML contents shall be resized when the rows and columns between its start and ending anchor (the from and to child elements) are resized, or have additional rows/columns inserted within them.

[*Example*: Consider a drawing defined as follows:

```
<ws:twoCellAnchor editAs="absolute">
  ...
</ws:twoCellAnchor>
```

The editAs attribute has a value of absolute, which specifies that the sizing of this object shall not change, instead the anchor locations should be moved as needed to maintain the same size. *end example*]

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
absolute (Do Not Move or Resize With Underlying Rows/Columns)	Specifies that the current start and end positions shall be maintained with respect to the distances from the absolute start point of the worksheet. If additional rows/columns are added before the drawing, the drawing shall move its anchors as needed to maintain this same absolute position.
oneCell (Move With Cells but Do Not Resize)	Specifies that the current drawing shall move with its row and column (i.e. the object is anchored to the actual from row and column), but that the size shall remain absolute. If additional rows/columns are added between the from and to locations of the drawing, the drawing

Enumeration Value	Description
	shall move its to anchors as needed to maintain this same absolute size.
twoCell (Move and Resize With Anchor Cells)	Specifies that the current drawing shall move and resize to maintain its row and column anchors (i.e. the object is anchored to the actual from and to row and column).

Referenced By
twoCellAnchor@editAs (§5.6.2.32)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_EditAs">
   <restriction base="xsd:token">
      <enumeration value="twoCell"/>
      <enumeration value="oneCell"/>
      <enumeration value="absolute"/>
   </restriction>
</simpleType>
```

5.6.3.3 ST_RowID (Row ID)

This type specifies a row identification. The numerical value used for the row id should be non-negative and never exceed the number of total rows within the spreadsheet document.

This simple type's contents are a restriction of the XML Schema int datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 0.

Referenced By
row (§5.6.2.26)

5.7 DrawingML - Charts

The chart namespace in DrawingML is for representing visualizations of numeric data with column charts, pie charts, scatter charts, or other types of charts.

5.7.1 Table of Contents

This subclause is informative.

End of informative text.

5.7.2 Elements

In DrawingML, charts define a visualization of numeric data. The definition includes where the data shall come from, a cache of the data, and how the data shall be represented graphically. Other DrawingML elements are reused to define aspects of the formatting of the visualization.

See the informative material in Part 3 for a description and overview of the basic chart types and chart components.

5.7.2.1 applyToEnd (Apply to End)

This element specifies the picture shall be applied to the end of the point or series.

Parent Elements
pictureOptions (§5.7.2.139)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.2 applyToFront (Apply To Front)

This element specifies the picture shall be applied to the front of the point or series.

Parent Elements
pictureOptions (§5.7.2.139)

Attributes	Description

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.3 applyToSides (Apply To Sides)

This element specifies the picture shall be applied to the sides of the point or series.

Parent Elements
pictureOptions (§5.7.2.139)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.4 area3DChart (3D Area Charts)

This element specifies the 3-D area series on this chart.

Parent Elements
plotArea (§5.7.2.146)

Child Elements	Subclause
axId (Axis ID)	§5.7.2.9
dLbls (Data Labels)	§5.7.2.49
dropLines (Drop Lines)	§5.7.2.53
extLst (Chart Extensibility)	§5.7.2.64
gapDepth (Gap Depth)	§5.7.2.74
grouping (Grouping)	§5.7.2.76
ser (Area Chart Series)	§5.7.2.174
varyColors (Vary Colors by Point)	§5.7.2.228

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Area3DChart">
   <sequence>
      <group ref="EG_AreaChartShared" minOccurs="1" maxOccurs="1"/>
      <element name="gapDepth" type="CT_GapAmount" minOccurs="0" maxOccurs="1"/>
      <element name="axId" type="CT_UnsignedInt" minOccurs="2" maxOccurs="3"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.5 areaChart (Area Charts)

This element specifies the 2-D area series on this chart.

Parent Elements
plotArea (§5.7.2.146)

Child Elements	Subclause
axId (Axis ID)	§5.7.2.9
dLbls (Data Labels)	§5.7.2.49
dropLines (Drop Lines)	§5.7.2.53
extLst (Chart Extensibility)	§5.7.2.64
grouping (Grouping)	§5.7.2.76
ser (Area Chart Series)	§5.7.2.174
varyColors (Vary Colors by Point)	§5.7.2.228

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AreaChart">
   <sequence>
      <group ref="EG_AreaChartShared" minOccurs="1" maxOccurs="1"/>
      <element name="axId" type="CT_UnsignedInt" minOccurs="2" maxOccurs="2"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.6 auto (Automatic Category Axis)

This element specifies that this axis is a date or text axis based on the data that is used for the axis labels, not a specific choice.

Parent Elements
catAx (§5.7.2.25); dateAx (§5.7.2.39)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.7 autoTitleDeleted (Auto Title Is Deleted)

This element specifies the title shall not be shown for this chart.

Parent Elements
chart (§5.7.2.27)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is

Attributes	Description
	omitted. A value of `off`, `0`, or `false` specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.8 autoUpdate (Update Automatically)

This element specifies the external data shall be updated automatically when the document containing the chart is opened.

Parent Elements
externalData (§5.7.2.63)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.9 axId (Axis ID)

When specified as a child element of valAx, dateAx, catAx, or serAx, this element specifies the identifier for the axis. When specified as a child element of a chart, this element specifies the identifier of an axis that defines the coordinate space of the chart.

Parent Elements
area3DChart (§5.7.2.4); areaChart (§5.7.2.5); bar3DChart (§5.7.2.15); barChart (§5.7.2.16); bubbleChart (§5.7.2.20); catAx (§5.7.2.25); dateAx (§5.7.2.39); line3DChart (§5.7.2.97); lineChart (§5.7.2.98); radarChart

Parent Elements
(§5.7.2.154); scatterChart (§5.7.2.162); serAx (§5.7.2.176); stockChart (§5.7.2.199); surface3DChart (§5.7.2.204); surfaceChart (§5.7.2.205); valAx (§5.7.2.227)

Attributes	Description
val (Integer Value)	Specifies that the contents of this attribute will contain an integer number. The contents of this number are interpreted based on the context of the parent XML element. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_UnsignedInt">
   <attribute name="val" type="xsd:unsignedInt" use="required"/>
</complexType>
```

5.7.2.10 axPos (Axis Position)

This element specifies the position of the axis on the chart.

Parent Elements
catAx (§5.7.2.25); dateAx (§5.7.2.39); serAx (§5.7.2.176); valAx (§5.7.2.227)

Attributes	Description
val (Axis Position Value)	Specifies the position of the axis on the chart. The possible values for this attribute are defined by the ST_AxPos simple type (§5.7.3.2).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AxPos">
   <attribute name="val" type="ST_AxPos" use="required"/>
</complexType>
```

5.7.2.11 backWall (Back Wall)

This element specifies the back wall of the chart.

Parent Elements
chart (§5.7.2.27)

Child Elements	Subclause

Child Elements	Subclause
extLst (Chart Extensibility)	§5.7.2.64
pictureOptions (Picture Options)	§5.7.2.139
spPr (Shape Properties)	§5.7.2.198
thickness (Thickness)	§5.7.2.207

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Surface">
   <sequence>
      <element name="thickness" type="CT_UnsignedInt" minOccurs="0" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
      <element name="pictureOptions" type="CT_PictureOptions" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.12 backward (Backward)

This element specifies the number of categories (or units on a scatter chart) that the trend line extends before the data for the series that is being trended. On non-scatter charts, the value shall be 0 or 0.5.

Parent Elements
trendline (§5.7.2.212)

Attributes	Description
val (Floating Point Value)	Specifies that the contents of this attribute will contain a floating point number. The contents of this number are interpreted based on the context of the parent XML element. The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Double">
   <attribute name="val" type="xsd:double" use="required"/>
</complexType>
```

5.7.2.13 bandFmt (Band Format)

This element specifies the formatting band of a surface chart.

Parent Elements
bandFmts (§5.7.2.14)

Child Elements	Subclause
idx (Index)	§5.7.2.84
spPr (Shape Properties)	§5.7.2.198

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BandFmt">
   <sequence>
      <element name="idx" type="CT_UnsignedInt" minOccurs="1" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.14 bandFmts (Band Formats)

This element contains a collection of formatting bands for a surface chart indexed from low to high.

Parent Elements
surface3DChart (§5.7.2.204); surfaceChart (§5.7.2.205)

Child Elements	Subclause
bandFmt (Band Format)	§5.7.2.13

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BandFmts">
   <sequence>
      <element name="bandFmt" type="CT_BandFmt" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.7.2.15 bar3DChart (3D Bar Charts)

This element contains the 3-D bar or column series on this chart.

Parent Elements
plotArea (§5.7.2.146)

Child Elements	Subclause
axId (Axis ID)	§5.7.2.9
barDir (Bar Direction)	§5.7.2.17
dLbls (Data Labels)	§5.7.2.49
extLst (Chart Extensibility)	§5.7.2.64
gapDepth (Gap Depth)	§5.7.2.74

Child Elements	Subclause
gapWidth (Gap Width)	§5.7.2.75
grouping (Bar Grouping)	§5.7.2.77
ser (Bar Chart Series)	§5.7.2.175
shape (Shape)	§5.7.2.178
varyColors (Vary Colors by Point)	§5.7.2.228

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Bar3DChart">
   <sequence>
      <group ref="EG_BarChartShared" minOccurs="1" maxOccurs="1"/>
      <element name="gapWidth" type="CT_GapAmount" minOccurs="0" maxOccurs="1"/>
      <element name="gapDepth" type="CT_GapAmount" minOccurs="0" maxOccurs="1"/>
      <element name="shape" type="CT_Shape" minOccurs="0" maxOccurs="1"/>
      <element name="axId" type="CT_UnsignedInt" minOccurs="2" maxOccurs="3"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.16 barChart (Bar Charts)

This element contains the 2-D bar or column series on this chart.

Parent Elements
plotArea (§5.7.2.146)

Child Elements	Subclause
axId (Axis ID)	§5.7.2.9
barDir (Bar Direction)	§5.7.2.17
dLbls (Data Labels)	§5.7.2.49
extLst (Chart Extensibility)	§5.7.2.64
gapWidth (Gap Width)	§5.7.2.75
grouping (Bar Grouping)	§5.7.2.77
overlap (Overlap)	§5.7.2.132
ser (Bar Chart Series)	§5.7.2.175
serLines (Series Lines)	§5.7.2.177
varyColors (Vary Colors by Point)	§5.7.2.228

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BarChart">
   <sequence>
      <group ref="EG_BarChartShared" minOccurs="1" maxOccurs="1"/>
      <element name="gapWidth" type="CT_GapAmount" minOccurs="0" maxOccurs="1"/>
      <element name="overlap" type="CT_Overlap" minOccurs="0" maxOccurs="1"/>
      <element name="serLines" type="CT_ChartLines" minOccurs="0" maxOccurs="unbounded"/>
      <element name="axId" type="CT_UnsignedInt" minOccurs="2" maxOccurs="2"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.17 barDir (Bar Direction)

This element specifies whether the series form a bar (horizontal) chart or a column (vertical) chart

Parent Elements
bar3DChart (§5.7.2.15); barChart (§5.7.2.16)

Attributes	Description
val (Bar Direction Value)	Specifies the direction of the series. The possible values for this attribute are defined by the ST_BarDir simple type (§5.7.3.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BarDir">
   <attribute name="val" type="ST_BarDir" default="col"/>
</complexType>
```

5.7.2.18 baseTimeUnit (Base Time Unit)

This element specifies the smallest time unit that is represented on the date axis.

Parent Elements
dateAx (§5.7.2.39)

Attributes	Description
val (Time Unit Value)	Specifies the time unit for the tick marks. The possible values for this attribute are defined by the ST_TimeUnit simple type (§5.7.3.50).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TimeUnit">
   <attribute name="val" type="ST_TimeUnit" default="days"/>
</complexType>
```

5.7.2.19 bubble3D (3D Bubble)

This element specifies that the bubbles have a 3-D effect applied to them.

Parent Elements
bubbleChart (§5.7.2.20); dPt (§5.7.2.52); ser (§5.7.2.168)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.20 bubbleChart (Bubble Charts)

This element contains the bubble series on this chart.

Parent Elements
plotArea (§5.7.2.146)

Child Elements	Subclause
axId (Axis ID)	§5.7.2.9
bubble3D (3D Bubble)	§5.7.2.19
bubbleScale (Bubble Scale)	§5.7.2.21
dLbls (Data Labels)	§5.7.2.49
extLst (Chart Extensibility)	§5.7.2.64
ser (Bubble Chart Series)	§5.7.2.168

Child Elements	Subclause
showNegBubbles (Show Negative Bubbles)	§5.7.2.186
sizeRepresents (Size Represents)	§5.7.2.194
varyColors (Vary Colors by Point)	§5.7.2.228

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BubbleChart">
   <sequence>
      <element name="varyColors" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="ser" type="CT_BubbleSer" minOccurs="0" maxOccurs="unbounded"/>
      <element name="dLbls" type="CT_DLbls" minOccurs="0" maxOccurs="1"/>
      <element name="bubble3D" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="bubbleScale" type="CT_BubbleScale" minOccurs="0" maxOccurs="1"/>
      <element name="showNegBubbles" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="sizeRepresents" type="CT_SizeRepresents" minOccurs="0" maxOccurs="1"/>
      <element name="axId" type="CT_UnsignedInt" minOccurs="2" maxOccurs="2"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.21 bubbleScale (Bubble Scale)

This element specifies the scale factor for the bubble chart. This element can be an integer value from 0 to 300, corresponding to a percentage of the default size.

Parent Elements
bubbleChart (§5.7.2.20)

Attributes	Description
val (Bubble Scale Value)	Specifies how to scale bubbles on a bubble chart. The possible values for this attribute are defined by the ST_BubbleScale simple type (§5.7.3.5).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BubbleScale">
   <attribute name="val" type="ST_BubbleScale" default="100"/>
</complexType>
```

5.7.2.22 bubbleSize (Bubble Size)

This element specifies the data for the sizes of the bubbles on the bubble chart.

Parent Elements
ser (§5.7.2.168)

Child Elements	Subclause
numLit (Number Literal)	§5.7.2.123
numRef (Number Reference)	§5.7.2.124

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NumDataSource">
   <sequence>
      <choice minOccurs="1" maxOccurs="1">
         <element name="numRef" type="CT_NumRef" minOccurs="1" maxOccurs="1"/>
         <element name="numLit" type="CT_NumData" minOccurs="1" maxOccurs="1"/>
      </choice>
   </sequence>
</complexType>
```

5.7.2.23 builtInUnit (Built in Display Unit Value)

This element specifies the display unit is one of the built in values.

Parent Elements
dispUnits (§5.7.2.45)

Attributes	Description
val (Built In Unit Value)	Specifies the type of display unit scaling applied to the axis. The possible values for this attribute are defined by the ST_BuiltInUnit simple type (§5.7.3.6).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BuiltInUnit">
   <attribute name="val" type="ST_BuiltInUnit" default="thousands"/>
</complexType>
```

5.7.2.24 cat (Category Axis Data)

This element specifies the data used for the category axis.

Parent Elements
ser (§5.7.2.174); ser (§5.7.2.171); ser (§5.7.2.175); ser (§5.7.2.170); ser (§5.7.2.169); ser (§5.7.2.173)

Child Elements	Subclause
multiLvlStrRef (Multi Level String Reference)	§5.7.2.116
numLit (Number Literal)	§5.7.2.123
numRef (Number Reference)	§5.7.2.124

Child Elements	Subclause
strLit (String Literal)	§5.7.2.201
strRef (String Reference)	§5.7.2.202

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AxDataSource">
   <sequence>
      <choice minOccurs="1" maxOccurs="1">
         <element name="multiLvlStrRef" type="CT_MultiLvlStrRef" minOccurs="1" maxOccurs="1"/>
         <element name="numRef" type="CT_NumRef" minOccurs="1" maxOccurs="1"/>
         <element name="numLit" type="CT_NumData" minOccurs="1" maxOccurs="1"/>
         <element name="strRef" type="CT_StrRef" minOccurs="1" maxOccurs="1"/>
         <element name="strLit" type="CT_StrData" minOccurs="1" maxOccurs="1"/>
      </choice>
   </sequence>
</complexType>
```

5.7.2.25 catAx (Category Axis Data)

This element specifies the category axis of the chart.

Parent Elements
plotArea (§5.7.2.146)

Child Elements	Subclause
auto (Automatic Category Axis)	§5.7.2.6
axId (Axis ID)	§5.7.2.9
axPos (Axis Position)	§5.7.2.10
crossAx (Crossing Axis ID)	§5.7.2.31
crosses (Crosses)	§5.7.2.33
crossesAt (Crossing Value)	§5.7.2.34
delete (Delete)	§5.7.2.40
extLst (Chart Extensibility)	§5.7.2.64
lblAlgn (Label Alignment)	§5.7.2.90
lblOffset (Label Offset)	§5.7.2.91
majorGridlines (Major Gridlines)	§5.7.2.101
majorTickMark (Major Tick Mark)	§5.7.2.102
minorGridlines (Minor Gridlines)	§5.7.2.110
minorTickMark (Minor Tick Mark)	§5.7.2.111
noMultiLvlLbl (No Multi-level Labels)	§5.7.2.120
numFmt (Number Format)	§5.7.2.122

Child Elements	Subclause
scaling (Scaling)	§5.7.2.161
spPr (Shape Properties)	§5.7.2.198
tickLblPos (Tick Label Position)	§5.7.2.208
tickLblSkip (Tick Label Skip)	§5.7.2.209
tickMarkSkip (Tick Mark Skip)	§5.7.2.210
title (Title)	§5.7.2.211
txPr (Text Properties)	§5.7.2.217

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CatAx">
   <sequence>
      <group ref="EG_AxShared" minOccurs="1" maxOccurs="1"/>
      <element name="auto" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="lblAlgn" type="CT_LblAlgn" minOccurs="0" maxOccurs="1"/>
      <element name="lblOffset" type="CT_LblOffset" minOccurs="0" maxOccurs="1"/>
      <element name="tickLblSkip" type="CT_Skip" minOccurs="0" maxOccurs="1"/>
      <element name="tickMarkSkip" type="CT_Skip" minOccurs="0" maxOccurs="1"/>
      <element name="noMultiLvlLbl" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.26 chart (Reference to Chart Part)

This element specifies the chart.

Attributes	Description
id (Relationship Reference) Namespace: .../officeDocument/2006/relationships	Specifies the relationship ID for the relationship for this Chart, Chart Drawing, or VML Drawing part. The type of relationship needed is specified by the parent element. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RelId">
   <attribute ref="r:id" use="required"/>
</complexType>
```

5.7.2.27 chart (Chart)

This element specifies the chart.

Parent Elements

Parent Elements
chartSpace (§5.7.2.29)

Child Elements	Subclause
autoTitleDeleted (Auto Title Is Deleted)	§5.7.2.7
backWall (Back Wall)	§5.7.2.11
dispBlanksAs (Display Blanks As)	§5.7.2.42
extLst (Chart Extensibility)	§5.7.2.64
floor (Floor)	§5.7.2.69
legend (Legend)	§5.7.2.94
pivotFmts (Pivot Formats)	§5.7.2.144
plotArea (Plot Area)	§5.7.2.146
plotVisOnly (Plot Visible Only)	§5.7.2.147
showDLblsOverMax (Show Data Labels over Maximum)	§5.7.2.181
sideWall (Side Wall)	§5.7.2.192
title (Title)	§5.7.2.211
view3D (View In 3D)	§5.7.2.229

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Chart">
   <sequence>
      <element name="title" type="CT_Title" minOccurs="0" maxOccurs="1"/>
      <element name="autoTitleDeleted" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="pivotFmts" type="CT_PivotFmts" minOccurs="0" maxOccurs="1"/>
      <element name="view3D" type="CT_View3D" minOccurs="0" maxOccurs="1"/>
      <element name="floor" type="CT_Surface" minOccurs="0" maxOccurs="1"/>
      <element name="sideWall" type="CT_Surface" minOccurs="0" maxOccurs="1"/>
      <element name="backWall" type="CT_Surface" minOccurs="0" maxOccurs="1"/>
      <element name="plotArea" type="CT_PlotArea" minOccurs="1" maxOccurs="1"/>
      <element name="legend" type="CT_Legend" minOccurs="0" maxOccurs="1"/>
      <element name="plotVisOnly" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="dispBlanksAs" type="CT_DispBlanksAs" minOccurs="0" maxOccurs="1"/>
      <element name="showDLblsOverMax" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.28 chartObject (Chart Object)

This element specifies that the chart cannot be edited by the user

Parent Elements
protection (§5.7.2.150)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.29 chartSpace (Chart Space)

This element specifies overall settings for a single chart, and is the root node for the chart part.

Parent Elements
Root element of DrawingML Chart part

Child Elements	Subclause
chart (Chart)	§5.7.2.27
clrMapOvr (Color Map Override)	§5.7.2.30
date1904 (1904 Date System)	§5.7.2.38
externalData (External Data Relationship)	§5.7.2.63
extLst (Chart Extensibility)	§5.7.2.64
lang (Editing Language)	§5.7.2.87
pivotSource (Pivot Source)	§5.7.2.145
printSettings (Print Settings)	§5.7.2.149
protection (Protection)	§5.7.2.150
roundedCorners (Rounded Corners)	§5.7.2.160
spPr (Shape Properties)	§5.7.2.198
style (Style)	§5.7.2.203
txPr (Text Properties)	§5.7.2.217
userShapes (Reference to Chart Drawing Part)	§5.7.2.222

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ChartSpace">
   <sequence>
      <element name="date1904" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="lang" type="CT_TextLanguageID" minOccurs="0" maxOccurs="1"/>
      <element name="roundedCorners" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="style" type="CT_Style" minOccurs="0" maxOccurs="1"/>
      <element name="clrMapOvr" type="a:CT_ColorMapping" minOccurs="0" maxOccurs="1"/>
      <element name="pivotSource" type="CT_PivotSource" minOccurs="0" maxOccurs="1"/>
      <element name="protection" type="CT_Protection" minOccurs="0" maxOccurs="1"/>
      <element name="chart" type="CT_Chart" minOccurs="1" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
      <element name="txPr" type="a:CT_TextBody" minOccurs="0" maxOccurs="1"/>
      <element name="externalData" type="CT_ExternalData" minOccurs="0" maxOccurs="1"/>
      <element name="printSettings" type="CT_PrintSettings" minOccurs="0" maxOccurs="1"/>
      <element name="userShapes" type="CT_RelId" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.30 clrMapOvr (Color Map Override)

This element represents color mapping information. It is used to override the applications color mapping if the user has selected keep source formatting after a copy-paste.

Parent Elements
chartSpace (§5.7.2.29)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15

Attributes	Description
accent1 (Accent 1) Namespace: .../drawingml/2006/main	Specifies a color defined which is associated as the accent 1 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
accent2 (Accent 2) Namespace: .../drawingml/2006/main	Specifies a color defined which is associated as the accent 2 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
accent3 (Accent 3) Namespace:	Specifies a color defined which is associated as the accent 3 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple

Attributes	Description
.../drawingml/2006/main	type (§5.1.12.14).
accent4 (Accent 4) Namespace: .../drawingml/2006/main	Specifies a color defined which is associated as the accent 4 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
accent5 (Accent 5) Namespace: .../drawingml/2006/main	Specifies a color defined which is associated as the accent 5 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
accent6 (Accent 6) Namespace: .../drawingml/2006/main	Specifies a color defined which is associated as the accent 6 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
bg1 (Background 1) Namespace: .../drawingml/2006/main	A color defined which is associated as the first background color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
bg2 (Background 2) Namespace: .../drawingml/2006/main	Specifies a color defined which is associated as the second background color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
folHlink (Followed Hyperlink) Namespace: .../drawingml/2006/main	Specifies a color defined which is associated as the color for a followed hyperlink. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
hlink (Hyperlink) Namespace: .../drawingml/2006/main	Specifies a color defined which is associated as the color for a hyperlink. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
tx1 (Text 1) Namespace: .../drawingml/2006/main	Specifies a color defined which is associated as the first text color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).
tx2 (Text 2)	Specifies a color defined which is associated as the second text color.

Attributes	Description
Namespace: .../drawingml/2006/main	The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§5.1.12.14).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ColorMapping">
   <sequence>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="bg1" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="tx1" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="bg2" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="tx2" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent1" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent2" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent3" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent4" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent5" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="accent6" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="hlink" type="ST_ColorSchemeIndex" use="required"/>
   <attribute name="folHlink" type="ST_ColorSchemeIndex" use="required"/>
</complexType>
```

5.7.2.31 crossAx (Crossing Axis ID)

This element specifies the ID of axis that this axis crosses. For instance, a category axis may cross a value axis, and the category axis's crossAx would contain the ID of the value axis.

Parent Elements
catAx (§5.7.2.25); dateAx (§5.7.2.39); serAx (§5.7.2.176); valAx (§5.7.2.227)

Attributes	Description
val (Integer Value)	Specifies that the contents of this attribute will contain an integer number. The contents of this number are interpreted based on the context of the parent XML element. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_UnsignedInt">
   <attribute name="val" type="xsd:unsignedInt" use="required"/>
</complexType>
```

5.7.2.32 crossBetween (Cross Between)

This element specifies whether the value axis crosses the category axis between categories.

If not specified, then the application should choose an appropriate behavior.

Parent Elements
valAx (§5.7.2.227)

Attributes	Description
val (Cross Between Value)	Specifies whether the value axis crosses the category axis between categories or on categories. The possible values for this attribute are defined by the ST_CrossBetween simple type (§5.7.3.7).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CrossBetween">
   <attribute name="val" type="ST_CrossBetween" use="required"/>
</complexType>
```

5.7.2.33 crosses (Crosses)

This element specifies how this axis crosses the perpendicular axis.

Parent Elements
catAx (§5.7.2.25); dateAx (§5.7.2.39); serAx (§5.7.2.176); valAx (§5.7.2.227)

Attributes	Description
val (Crosses Value)	Specifies where the axis crosses its perpendicular axis. The possible values for this attribute are defined by the ST_Crosses simple type (§5.7.3.8).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Crosses">
   <attribute name="val" type="ST_Crosses" use="required"/>
</complexType>
```

5.7.2.34 crossesAt (Crossing Value)

This element specifies where on the axis the perpendicular axis crosses. The units are dependent on the type of axis.

When specified as a child element of valAx, the value is a decimal number on the value axis. When specified as a child element of dateAx, the date is defined as a integer number of days since Dec. 31, 1899, unless date1904 is set, in which case they are defined as an integer number of days since Dec. 31, 1903. When specified as a child element of catAx, the value is an integer category number, starting with 1 as the first category.

Parent Elements
catAx (§5.7.2.25); dateAx (§5.7.2.39); serAx (§5.7.2.176); valAx (§5.7.2.227)

Attributes	Description
val (Floating Point Value)	Specifies that the contents of this attribute will contain a floating point number. The contents of this number are interpreted based on the context of the parent XML element. The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Double">
   <attribute name="val" type="xsd:double" use="required"/>
</complexType>
```

5.7.2.35 custSplit (Custom Split)

This element contains the custom split information for a pie-of-pie or bar-of-pie chart with a custom split type.

Parent Elements
ofPieChart (§5.7.2.127)

Child Elements	Subclause
secondPiePt (Second Pie Point)	§5.7.2.164

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CustSplit">
   <sequence>
      <element name="secondPiePt" type="CT_UnsignedInt" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.7.2.36 custUnit (Custom Display Unit)

This element specifies a custom value for the display unit.

Parent Elements
dispUnits (§5.7.2.45)

Attributes	Description
val (Floating Point Value)	Specifies that the contents of this attribute will contain a floating point number. The contents of this number are interpreted based on the context of the parent XML element. The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Double">
   <attribute name="val" type="xsd:double" use="required"/>
</complexType>
```

5.7.2.37 data (Data Cannot Be Changed)

This element specifies that the user cannot change the choice of data used for the chart

Parent Elements
protection (§5.7.2.150)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.38 date1904 (1904 Date System)

This element specifies that the chart uses the 1904 date system. If the 1904 date system is used, then all dates and times shall be specified as a decimal number of days since Dec. 31, 1903. If the 1904 date system is not used, then all dates and times shall be specified as a decimal number of days since Dec. 31, 1899.

Parent Elements

Parent Elements
chartSpace (§5.7.2.29)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.39 dateAx (Date Axis)

This element specifies a date axis for the chart.

Parent Elements
plotArea (§5.7.2.146)

Child Elements	Subclause
auto (Automatic Category Axis)	§5.7.2.6
axId (Axis ID)	§5.7.2.9
axPos (Axis Position)	§5.7.2.10
baseTimeUnit (Base Time Unit)	§5.7.2.18
crossAx (Crossing Axis ID)	§5.7.2.31
crosses (Crosses)	§5.7.2.33
crossesAt (Crossing Value)	§5.7.2.34
delete (Delete)	§5.7.2.40
extLst (Chart Extensibility)	§5.7.2.64
lblOffset (Label Offset)	§5.7.2.91
majorGridlines (Major Gridlines)	§5.7.2.101
majorTickMark (Major Tick Mark)	§5.7.2.102
majorTimeUnit (Major Time Unit)	§5.7.2.103

Child Elements	Subclause
majorUnit (Major Unit)	§5.7.2.104
minorGridlines (Minor Gridlines)	§5.7.2.110
minorTickMark (Minor Tick Mark)	§5.7.2.111
minorTimeUnit (Minor Time Unit)	§5.7.2.112
minorUnit (Minor Unit)	§5.7.2.113
numFmt (Number Format)	§5.7.2.122
scaling (Scaling)	§5.7.2.161
spPr (Shape Properties)	§5.7.2.198
tickLblPos (Tick Label Position)	§5.7.2.208
title (Title)	§5.7.2.211
txPr (Text Properties)	§5.7.2.217

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DateAx">
   <sequence>
      <group ref="EG_AxShared" minOccurs="1" maxOccurs="1"/>
      <element name="auto" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="lblOffset" type="CT_LblOffset" minOccurs="0" maxOccurs="1"/>
      <element name="baseTimeUnit" type="CT_TimeUnit" minOccurs="0" maxOccurs="1"/>
      <element name="majorUnit" type="CT_AxisUnit" minOccurs="0" maxOccurs="1"/>
      <element name="majorTimeUnit" type="CT_TimeUnit" minOccurs="0" maxOccurs="1"/>
      <element name="minorUnit" type="CT_AxisUnit" minOccurs="0" maxOccurs="1"/>
      <element name="minorTimeUnit" type="CT_TimeUnit" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.40 delete (Delete)

This element specifies that the chart element specified by its containing element shall be deleted from the chart.

This should be set to true if the application adds these elements by default even after the user has specified that they should be removed from the chart.

Parent Elements
catAx (§5.7.2.25); dateAx (§5.7.2.39); dLbl (§5.7.2.47); dLbls (§5.7.2.49); legendEntry (§5.7.2.95); serAx (§5.7.2.176); valAx (§5.7.2.227)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property is applied. This is the default value for

Attributes	Description
	this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.41 depthPercent (Depth Percent)

This element specifies the depth of a 3-D chart as a percentage of the chart width (between 20 and 2000 percent).

Parent Elements
view3D (§5.7.2.229)

Attributes	Description
val (Depth Percent Value)	Specifies an integer value for the property defined by the parent XML element. The possible values for this attribute are defined by the ST_DepthPercent simple type (§5.7.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DepthPercent">
   <attribute name="val" type="ST_DepthPercent" default="100"/>
</complexType>
```

5.7.2.42 dispBlanksAs (Display Blanks As)

This element specifies how blank cells shall be plotted on a chart .

Parent Elements
chart (§5.7.2.27)

Attributes	Description
val (Display Blanks As Value)	Specifies how blank cells are plotted on the chart. The possible values for this attribute are defined by the ST_DispBlanksAs simple type (§5.7.3.10).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DispBlanksAs">
   <attribute name="val" type="ST_DispBlanksAs" default="zero"/>
</complexType>
```

5.7.2.43 dispEq (Display Equation)

This element specifies that the equation for the trendline is displayed on the chart (in the same label as the R-squared value).

Parent Elements
trendline (§5.7.2.212)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.44 dispRSqr (Display R Squared Value)

This element specifies that the R-squared value of the trendline is displayed on the chart (in the same label as the equation).

Parent Elements
trendline (§5.7.2.212)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted.

Attributes	Description
	A value of `off`, `0`, or `false` specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.45 dispUnits (Display Units)

This element specifies the scaling value of the display units for the value axis.

Parent Elements
valAx (§5.7.2.227)

Child Elements	Subclause
builtInUnit (Built in Display Unit Value)	§5.7.2.23
custUnit (Custom Display Unit)	§5.7.2.36
dispUnitsLbl (Display Units Label)	§5.7.2.46
extLst (Chart Extensibility)	§5.7.2.64

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DispUnits">
   <sequence>
      <choice>
         <element name="custUnit" type="CT_Double" minOccurs="1" maxOccurs="1"/>
         <element name="builtInUnit" type="CT_BuiltInUnit" minOccurs="1" maxOccurs="1"/>
      </choice>
      <element name="dispUnitsLbl" type="CT_DispUnitsLbl" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.46 dispUnitsLbl (Display Units Label)

This element specifies the display unit label for the value axis in the specified chart.

Parent Elements
dispUnits (§5.7.2.45)

Child Elements	Subclause
layout (Layout)	§5.7.2.88

Child Elements	Subclause
spPr (Shape Properties)	§5.7.2.198
tx (Chart Text)	§5.7.2.215
txPr (Text Properties)	§5.7.2.217

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DispUnitsLbl">
   <sequence>
      <element name="layout" type="CT_Layout" minOccurs="0" maxOccurs="1"/>
      <element name="tx" type="CT_Tx" minOccurs="0" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
      <element name="txPr" type="a:CT_TextBody" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.47 dLbl (Data Label)

This element specifies a data label.

Parent Elements
dLbls (§5.7.2.49); pivotFmt (§5.7.2.143)

Child Elements	Subclause
delete (Delete)	§5.7.2.40
dLblPos (Data Label Position)	§5.7.2.48
extLst (Chart Extensibility)	§5.7.2.64
idx (Index)	§5.7.2.84
layout (Layout)	§5.7.2.88
numFmt (Number Format)	§5.7.2.122
separator (Separator)	§5.7.2.167
showBubbleSize (Show Bubble Size)	§5.7.2.179
showCatName (Show Category Name)	§5.7.2.180
showLegendKey (Show Legend Key)	§5.7.2.185
showPercent (Show Percent)	§5.7.2.188
showSerName (Show Series Name)	§5.7.2.189
showVal (Show Value)	§5.7.2.190
spPr (Shape Properties)	§5.7.2.198
tx (Chart Text)	§5.7.2.215
txPr (Text Properties)	§5.7.2.217

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DLbl">
   <sequence>
      <element name="idx" type="CT_UnsignedInt" minOccurs="1" maxOccurs="1"/>
      <choice>
         <element name="delete" type="CT_Boolean" minOccurs="1" maxOccurs="1"/>
         <group ref="Group_DLbl" minOccurs="1" maxOccurs="1"/>
      </choice>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.48 dLblPos (Data Label Position)

This element specifies the position of the data label.

Parent Elements
dLbl (§5.7.2.47); dLbls (§5.7.2.49)

Attributes	Description
val (Data Label Position Value)	Specifies how the data label is positioned on the chart. The possible values for this attribute are defined by the ST_DLblPos simple type (§5.7.3.11).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DLblPos">
   <attribute name="val" type="ST_DLblPos" use="required"/>
</complexType>
```

5.7.2.49 dLbls (Data Labels)

This element serves as a root element that specifies the settings for the data labels for an entire series or the entire chart. It contains child elements that specify the specific formatting and positioning settings.

Parent Elements
area3DChart (§5.7.2.4); areaChart (§5.7.2.5); bar3DChart (§5.7.2.15); barChart (§5.7.2.16); bubbleChart (§5.7.2.20); doughnutChart (§5.7.2.50); line3DChart (§5.7.2.97); lineChart (§5.7.2.98); ofPieChart (§5.7.2.127); pie3DChart (§5.7.2.141); pieChart (§5.7.2.142); radarChart (§5.7.2.154); scatterChart (§5.7.2.162); ser (§5.7.2.168); ser (§5.7.2.173); ser (§5.7.2.174); ser (§5.7.2.175); ser (§5.7.2.172); ser (§5.7.2.169); ser (§5.7.2.170); stockChart (§5.7.2.199)

Child Elements	Subclause
delete (Delete)	§5.7.2.40
dLbl (Data Label)	§5.7.2.47

Child Elements	Subclause
dLblPos (Data Label Position)	§5.7.2.48
extLst (Chart Extensibility)	§5.7.2.64
leaderLines (Leader Lines)	§5.7.2.92
numFmt (Number Format)	§5.7.2.122
separator (Separator)	§5.7.2.167
showBubbleSize (Show Bubble Size)	§5.7.2.179
showCatName (Show Category Name)	§5.7.2.180
showLeaderLines (Show Leader Lines)	§5.7.2.184
showLegendKey (Show Legend Key)	§5.7.2.185
showPercent (Show Percent)	§5.7.2.188
showSerName (Show Series Name)	§5.7.2.189
showVal (Show Value)	§5.7.2.190
spPr (Shape Properties)	§5.7.2.198
txPr (Text Properties)	§5.7.2.217

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DLbls">
   <sequence>
      <element name="dLbl" type="CT_DLbl" minOccurs="0" maxOccurs="unbounded"/>
      <choice>
         <element name="delete" type="CT_Boolean" minOccurs="1" maxOccurs="1"/>
         <group ref="Group_DLbls" minOccurs="1" maxOccurs="1"/>
      </choice>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.50 doughnutChart (Doughnut Charts)

This element contains the doughnut chart series.

Parent Elements
plotArea (§5.7.2.146)

Child Elements	Subclause
dLbls (Data Labels)	§5.7.2.49
extLst (Chart Extensibility)	§5.7.2.64
firstSliceAng (First Slice Angle)	§5.7.2.68
holeSize (Hole Size)	§5.7.2.82

Child Elements	Subclause
ser (Pie Chart Series)	§5.7.2.170
varyColors (Vary Colors by Point)	§5.7.2.228

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DoughnutChart">
   <sequence>
      <group ref="EG_PieChartShared" minOccurs="1" maxOccurs="1"/>
      <element name="firstSliceAng" type="CT_FirstSliceAng" minOccurs="0" maxOccurs="1"/>
      <element name="holeSize" type="CT_HoleSize" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.51 downBars (Down Bars)

This element specifies the down bars.

Parent Elements
upDownBars (§5.7.2.219)

Child Elements	Subclause
spPr (Shape Properties)	§5.7.2.198

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_UpDownBar">
   <sequence>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.52 dPt (Data Point)

This element specifies a single data point.

Parent Elements
ser (§5.7.2.172); ser (§5.7.2.170); ser (§5.7.2.174); ser (§5.7.2.173); ser (§5.7.2.169); ser (§5.7.2.175); ser (§5.7.2.168)

Child Elements	Subclause
bubble3D (3D Bubble)	§5.7.2.19
explosion (Explosion)	§5.7.2.61
extLst (Chart Extensibility)	§5.7.2.64

Child Elements	Subclause
idx (Index)	§5.7.2.84
invertIfNegative (Invert if Negative)	§5.7.2.86
marker (Marker)	§5.7.2.106
pictureOptions (Picture Options)	§5.7.2.139
spPr (Shape Properties)	§5.7.2.198

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DPt">
    <sequence>
        <element name="idx" type="CT_UnsignedInt" minOccurs="1" maxOccurs="1"/>
        <element name="invertIfNegative" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
        <element name="marker" type="CT_Marker" minOccurs="0" maxOccurs="1"/>
        <element name="bubble3D" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
        <element name="explosion" type="CT_UnsignedInt" minOccurs="0" maxOccurs="1"/>
        <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
        <element name="pictureOptions" type="CT_PictureOptions" minOccurs="0" maxOccurs="1"/>
        <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
    </sequence>
</complexType>
```

5.7.2.53 dropLines (Drop Lines)

This element specifies drop lines.

Parent Elements
area3DChart (§5.7.2.4); areaChart (§5.7.2.5); line3DChart (§5.7.2.97); lineChart (§5.7.2.98); stockChart (§5.7.2.199)

Child Elements	Subclause
spPr (Shape Properties)	§5.7.2.198

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ChartLines">
    <sequence>
        <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
    </sequence>
</complexType>
```

5.7.2.54 dTable (Data Table)

This element specifies a data table.

Parent Elements
plotArea (§5.7.2.146)

Child Elements	Subclause
extLst (Chart Extensibility)	§5.7.2.64
showHorzBorder (Show Horizontal Border)	§5.7.2.182
showKeys (Show Legend Keys)	§5.7.2.183
showOutline (Show Outline Border)	§5.7.2.187
showVertBorder (Show Vertical Border)	§5.7.2.191
spPr (Shape Properties)	§5.7.2.198
txPr (Text Properties)	§5.7.2.217

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DTable">
   <sequence>
      <element name="showHorzBorder" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="showVertBorder" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="showOutline" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="showKeys" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
      <element name="txPr" type="a:CT_TextBody" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.55 errBars (Error Bars)

This element specifies error bars. The errValType element controls whether the minus, plus, or val elements are used.

Parent Elements
ser (§5.7.2.172); ser (§5.7.2.168); ser (§5.7.2.174); ser (§5.7.2.169); ser (§5.7.2.175)

Child Elements	Subclause
errBarType (Error Bar Type)	§5.7.2.56
errDir (Error Bar Direction)	§5.7.2.57
errValType (Error Bar Value Type)	§5.7.2.58
extLst (Chart Extensibility)	§5.7.2.64
minus (Minus)	§5.7.2.114
noEndCap (No End Cap)	§5.7.2.119
plus (Plus)	§5.7.2.148
spPr (Shape Properties)	§5.7.2.198

Child Elements	Subclause
val (Error Bar Value)	§5.7.2.226

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ErrBars">
   <sequence>
      <element name="errDir" type="CT_ErrDir" minOccurs="0" maxOccurs="1"/>
      <element name="errBarType" type="CT_ErrBarType" minOccurs="1" maxOccurs="1"/>
      <element name="errValType" type="CT_ErrValType" minOccurs="1" maxOccurs="1"/>
      <element name="noEndCap" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="plus" type="CT_NumDataSource" minOccurs="0" maxOccurs="1"/>
      <element name="minus" type="CT_NumDataSource" minOccurs="0" maxOccurs="1"/>
      <element name="val" type="CT_Double" minOccurs="0" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.56 errBarType (Error Bar Type)

This element specifies the type of the error bars - positive, negative, or both.

Parent Elements
errBars (§5.7.2.55)

Attributes	Description
val (Error Bar Type Value)	Specifies the type of error bars. The possible values for this attribute are defined by the ST_ErrBarType simple type (§5.7.3.12).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ErrBarType">
   <attribute name="val" type="ST_ErrBarType" default="both"/>
</complexType>
```

5.7.2.57 errDir (Error Bar Direction)

This element specifies the direction of the error bars.

Parent Elements
errBars (§5.7.2.55)

Attributes	Description
val (Error Bar	Specifies the direction of the error bars.

Attributes	Description
Direction Value)	The possible values for this attribute are defined by the ST_ErrDir simple type (§5.7.3.13).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ErrDir">
   <attribute name="val" type="ST_ErrDir" use="required"/>
</complexType>
```

5.7.2.58 errValType (Error Bar Value Type)

This element specifies the type of values used to determine the length of the error bars.

Parent Elements
errBars (§5.7.2.55)

Attributes	Description
val (Error Bar Type Value)	Specifies the type of values of the error bars. The possible values for this attribute are defined by the ST_ErrValType simple type (§5.7.3.14).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ErrValType">
   <attribute name="val" type="ST_ErrValType" default="fixedVal"/>
</complexType>
```

5.7.2.59 evenFooter (Even Footer)

This element specifies the footer to use on even numbered pages.

The possible values for this element are defined by the ST_Xstring simple type (§5.7.3.52).

Parent Elements
headerFooter (§5.7.2.79)

5.7.2.60 evenHeader (Even Header)

This element specifies the header to use on even numbered pages.

The possible values for this element are defined by the ST_Xstring simple type (§5.7.3.52).

Parent Elements
headerFooter (§5.7.2.79)

5.7.2.61 explosion (Explosion)

This element specifies the amount the data point shall be moved from the center of the pie.

Parent Elements
dPt (§5.7.2.52); ser (§5.7.2.170)

Attributes	Description
val (Integer Value)	Specifies that the contents of this attribute will contain an integer number. The contents of this number are interpreted based on the context of the parent XML element. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_UnsignedInt">
   <attribute name="val" type="xsd:unsignedInt" use="required"/>
</complexType>
```

5.7.2.62 ext (Extension)

This element specifies an extension that is used for future extensions to the current version of DrawingML. This allows for the specifying of currently unknown elements in the future that will be used for later versions of generating applications.

Parent Elements
extLst (§5.7.2.64)

Child Elements	Subclause
Any element from any namespace	n/a

Attributes	Description
uri (Uniform Resource Identifier)	Specifies the URI, or uniform resource identifier that represents the data stored under this tag. The URI is used to identify the correct 'server' that can process the contents of this tag. The possible values for this attribute are defined by the XML Schema token datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Extension">
   <sequence>
      <any processContents="lax"/>
   </sequence>
   <attribute name="uri" type="xsd:token"/>
</complexType>
```

5.7.2.63 externalData (External Data Relationship)

This element specifies the relationship to the data for this chart.

The data may be linked, pointing to a spreadsheet in another file. Or, the data may be embedded, contained in a separate part within the same xml package containing the chart. In this case, it shall be stored as an embedded Spreadsheet object in Open XML format.

This is not used by a spreadsheet application as the spreadsheet application can maintain its own reference to the data in the spreadsheet via the formula <f> element.

Parent Elements
chartSpace (§5.7.2.29)

Child Elements	Subclause
autoUpdate (Update Automatically)	§5.7.2.8

Attributes	Description
id (Relationship Reference) Namespace: .../officeDocument/2006/relationships	Specifies the relationship ID for the relationship for this chart. The relationship explicitly targeted by this attribute shall be of type http://schemas.openxmlformats.org/officeDocument/2006/relationships/package. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ExternalData">
   <sequence>
      <element name="autoUpdate" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute ref="r:id" use="required"/>
</complexType>
```

5.7.2.64 extLst (Chart Extensibility)

This element contains tags used for future extensibility of the file format.

Parent Elements
area3DChart (§5.7.2.4); areaChart (§5.7.2.5); backWall (§5.7.2.11); bar3DChart (§5.7.2.15); barChart (§5.7.2.16); bubbleChart (§5.7.2.20); catAx (§5.7.2.25); chart (§5.7.2.27); chartSpace (§5.7.2.29); dateAx (§5.7.2.39); dispUnits (§5.7.2.45); dLbl (§5.7.2.47); dLbls (§5.7.2.49); doughnutChart (§5.7.2.50); dPt (§5.7.2.52); dTable (§5.7.2.54); errBars (§5.7.2.55); floor (§5.7.2.69); layout (§5.7.2.88); legend (§5.7.2.94); legendEntry (§5.7.2.95); line3DChart (§5.7.2.97); lineChart (§5.7.2.98); manualLayout (§5.7.2.105); marker (§5.7.2.106); multiLvlStrCache (§5.7.2.115); multiLvlStrRef (§5.7.2.116); numCache (§5.7.2.121); numLit (§5.7.2.123); numRef (§5.7.2.124); ofPieChart (§5.7.2.127); pie3DChart (§5.7.2.141); pieChart (§5.7.2.142); pivotFmt (§5.7.2.143); pivotSource (§5.7.2.145); plotArea (§5.7.2.146); radarChart (§5.7.2.154); scaling (§5.7.2.161); scatterChart (§5.7.2.162); ser (§5.7.2.174); ser (§5.7.2.170); ser (§5.7.2.172); ser (§5.7.2.175); ser (§5.7.2.169); ser (§5.7.2.168); ser (§5.7.2.173); ser (§5.7.2.171); serAx (§5.7.2.176); sideWall (§5.7.2.192); stockChart (§5.7.2.199); strCache (§5.7.2.200); strLit (§5.7.2.201); strRef (§5.7.2.202); surface3DChart (§5.7.2.204); surfaceChart (§5.7.2.205); title (§5.7.2.211); trendline (§5.7.2.212); trendlineLbl (§5.7.2.213); upDownBars (§5.7.2.219); valAx (§5.7.2.227); view3D (§5.7.2.229)

Child Elements	Subclause
ext (Extension)	§5.7.2.62

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ExtensionList">
   <sequence>
      <element name="ext" type="CT_Extension" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.7.2.65 f (Formula)

This element specifies a reference to source of the data contained in this chart. This shall be used by the spreadsheet application only. A presentation, or word processing application should use the `externalData` element.

This reference is in the form of a book, sheet, and cell reference or a book, optional sheet, and defined name reference. This reference does not include the equals sign. [*Example*:

```
<c:cat>
  <c:strRef>
    <c:f>Sheet1!$A$1:$C$1</c:f>
    <c:strCache>
…
    </c:strCache>
  </c:strRef>
</c:cat>
```

The above example shows a formula reference used for the string cahce. In this case the series names, which are referenced by the formula element, are in cells A1, B1, and C1.

end example

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
multiLvlStrRef (§5.7.2.116); numRef (§5.7.2.124); strRef (§5.7.2.202)

5.7.2.66 firstFooter (First Footer)

This element specifies the footer to use on the first page.

The possible values for this element are defined by the ST_Xstring simple type (§5.7.3.52).

Parent Elements
headerFooter (§5.7.2.79)

5.7.2.67 firstHeader (First Header)

This element specifies the header to use on the first page.

The possible values for this element are defined by the ST_Xstring simple type (§5.7.3.52).

Parent Elements
headerFooter (§5.7.2.79)

5.7.2.68 firstSliceAng (First Slice Angle)

This element specifies the angle of the first pie or doughnut chart slice, in degrees (clockwise from up).

Parent Elements
doughnutChart (§5.7.2.50); pieChart (§5.7.2.142)

Attributes	Description
val (First Slice Angle Value)	Specifies the angle of the first slice. The possible values for this attribute are defined by the ST_FirstSliceAng simple type (§5.7.3.15).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FirstSliceAng">
   <attribute name="val" type="ST_FirstSliceAng" default="0"/>
</complexType>
```

5.7.2.69 floor (Floor)

This element specifies the floor of a 3D chart.

Parent Elements
chart (§5.7.2.27)

Child Elements	Subclause
extLst (Chart Extensibility)	§5.7.2.64
pictureOptions (Picture Options)	§5.7.2.139
spPr (Shape Properties)	§5.7.2.198
thickness (Thickness)	§5.7.2.207

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Surface">
   <sequence>
      <element name="thickness" type="CT_UnsignedInt" minOccurs="0" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
      <element name="pictureOptions" type="CT_PictureOptions" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.70 fmtId (Format ID)

This element represents a pivot format ID. It serves as a link back to the correct pivotTable which in turn specifies a link that then defines which set of chart format rules apply to this chart.

This ID shall match the chartFormat element, `chart` attribute, described in §3.10.1.12 of the SpreadsheetML reference material. The chartFormat element also contains a `format` attribute which is used to index into the pivotFmts collection (§5.7.2.144).

Parent Elements
pivotSource (§5.7.2.145)

Attributes	Description
val (Integer Value)	Specifies that the contents of this attribute will contain an integer number. The contents of this number are interpreted based on the context of the parent XML element. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_UnsignedInt">
   <attribute name="val" type="xsd:unsignedInt" use="required"/>
</complexType>
```

5.7.2.71 formatCode (Format Code)

This element specifies a string representing the format code to apply. For more information see the SpreadsheetML numFmt element's (§3.8.30) formatCode attribute.

The possible values for this element are defined by the ST_Xstring simple type (§5.7.3.52).

Parent Elements
numCache (§5.7.2.121); numLit (§5.7.2.123)

5.7.2.72 formatting (Formatting)

This element specifies that a user cannot change formatting on chart elements.

Parent Elements
protection (§5.7.2.150)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.73 forward (Forward)

This element specifies the number of categories (or units on a scatter chart) that the trendline extends after the data for the series that is being trended. On non-scatter charts, the value must be a multiple of 0.5.

Parent Elements
trendline (§5.7.2.212)

Attributes	Description
val (Floating Point Value)	Specifies that the contents of this attribute will contain a floating point number. The contents of this number are interpreted based on the context of the parent XML element. The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Double">
   <attribute name="val" type="xsd:double" use="required"/>
</complexType>
```

5.7.2.74 gapDepth (Gap Depth)

This element specifies the space between bar or column clusters, as a percentage of the bar or column width.

Parent Elements
area3DChart (§5.7.2.4); bar3DChart (§5.7.2.15); line3DChart (§5.7.2.97)

Attributes	Description
val (Gap Size Value)	Specifies that the contents of this attribute will contain a gap amount between 0 and 500. The possible values for this attribute are defined by the ST_GapAmount simple type (§5.7.3.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GapAmount">
   <attribute name="val" type="ST_GapAmount" default="150"/>
</complexType>
```

5.7.2.75 gapWidth (Gap Width)

This element specifies the space between bar or column clusters, as a percentage of the bar or column width.

Parent Elements
bar3DChart (§5.7.2.15); barChart (§5.7.2.16); ofPieChart (§5.7.2.127); upDownBars (§5.7.2.219)

Attributes	Description
val (Gap Size Value)	Specifies that the contents of this attribute will contain a gap amount between 0 and 500. The possible values for this attribute are defined by the ST_GapAmount simple type

Attributes	Description
	(§5.7.3.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GapAmount">
   <attribute name="val" type="ST_GapAmount" default="150"/>
</complexType>
```

5.7.2.76 grouping (Grouping)

This element specifies the type of grouping for a column, line, or area chart.

Parent Elements
area3DChart (§5.7.2.4); areaChart (§5.7.2.5); line3DChart (§5.7.2.97); lineChart (§5.7.2.98)

Attributes	Description
val (Grouping Value)	Specifies the grouping value. The possible values for this attribute are defined by the ST_Grouping simple type (§5.7.3.17).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Grouping">
   <attribute name="val" type="ST_Grouping" default="standard"/>
</complexType>
```

5.7.2.77 grouping (Bar Grouping)

This element specifies the type of grouping for a bar chart.

Parent Elements
bar3DChart (§5.7.2.15); barChart (§5.7.2.16)

Attributes	Description
val (Bar Grouping Value)	Specifies the bar grouping value. The possible values for this attribute are defined by the ST_BarGrouping simple type (§5.7.3.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BarGrouping">
   <attribute name="val" type="ST_BarGrouping" default="clustered"/>
</complexType>
```

5.7.2.78 h (Height)

This element specifies the height (if Height Mode is Factor) or bottom (if Height Mode is edge) of the chart element as a fraction of the height of the chart.

Parent Elements
manualLayout (§5.7.2.105)

Attributes	Description
val (Floating Point Value)	Specifies that the contents of this attribute will contain a floating point number. The contents of this number are interpreted based on the context of the parent XML element. The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Double">
   <attribute name="val" type="xsd:double" use="required"/>
</complexType>
```

5.7.2.79 headerFooter (Header and Footer)

This element specifies the headers and footers that shall be used when the chart is printed.

Parent Elements
printSettings (§5.7.2.149)

Child Elements	Subclause
evenFooter (Even Footer)	§5.7.2.59
evenHeader (Even Header)	§5.7.2.60
firstFooter (First Footer)	§5.7.2.66
firstHeader (First Header)	§5.7.2.67
oddFooter (Odd Footer)	§5.7.2.125
oddHeader (Odd Header)	§5.7.2.126

Attributes	Description
alignWithMargins (Align With Margins)	Specifies the header and footer should align with the left and right margins of the chart. A value of on, 1, or `true` specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is

Attributes	Description
	omitted. A value of `off`, `0`, or `false` specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.
differentFirst (Different First)	Specifies the header and footer are different for the first page. A value of on, `1`, or `true` specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.
differentOddEven (Different Odd Even)	Specifies the header and footer are different on odd-numbered pages and even-numbered pages. A value of on, `1`, or `true` specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HeaderFooter">
   <sequence>
      <element name="oddHeader" type="ST_Xstring" minOccurs="0" maxOccurs="1"/>
      <element name="oddFooter" type="ST_Xstring" minOccurs="0" maxOccurs="1"/>
      <element name="evenHeader" type="ST_Xstring" minOccurs="0" maxOccurs="1"/>
      <element name="evenFooter" type="ST_Xstring" minOccurs="0" maxOccurs="1"/>
      <element name="firstHeader" type="ST_Xstring" minOccurs="0" maxOccurs="1"/>
      <element name="firstFooter" type="ST_Xstring" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="alignWithMargins" type="xsd:boolean" default="true"/>
   <attribute name="differentOddEven" type="xsd:boolean" default="false"/>
   <attribute name="differentFirst" type="xsd:boolean" default="false"/>
</complexType>
```

5.7.2.80 hiLowLines (High Low Lines)

This element specifies the high-low lines for the series.

Parent Elements
lineChart (§5.7.2.98); stockChart (§5.7.2.199)

Child Elements	Subclause
spPr (Shape Properties)	§5.7.2.198

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ChartLines">
   <sequence>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.81 hMode (Height Mode)

This element specifies how to interpret the Height element for this manual layout.

Parent Elements
manualLayout (§5.7.2.105)

Attributes	Description
val (Layout Mode Value)	Specifies the layout mode for the width. The possible values for this attribute are defined by the ST_LayoutMode simple type (§5.7.3.20).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LayoutMode">
   <attribute name="val" type="ST_LayoutMode" default="factor"/>
</complexType>
```

5.7.2.82 holeSize (Hole Size)

This element specifies the size of the hole in a doughnut chart group.

Parent Elements
doughnutChart (§5.7.2.50)

Attributes	Description
val (Hole Size Value)	Specifies that the contents of this attribute will contain a hole size between 10 and 90 that is measured as a percentage of the size of the plot area. The possible values for this attribute are defined by the ST_HoleSize simple type (§5.7.3.18).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HoleSize">
   <attribute name="val" type="ST_HoleSize" default="10"/>
</complexType>
```

5.7.2.83 hPercent (Height Percent)

This element specifies the height of a 3-D chart as a percentage of the chart width.

Parent Elements
view3D (§5.7.2.229)

Attributes	Description
val (Height Percent Value)	Specifies that the contents of this attribute will contain a height percent between 5 and 500. The possible values for this attribute are defined by the ST_HPercent simple type (§5.7.3.19).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HPercent">
   <attribute name="val" type="ST_HPercent" default="100"/>
</complexType>
```

5.7.2.84 idx (Index)

This element specifies the index of the containing element. This index shall determine which of the parent's children collection this element applies to.

Parent Elements
bandFmt (§5.7.2.13); dLbl (§5.7.2.47); dPt (§5.7.2.52); legendEntry (§5.7.2.95); pivotFmt (§5.7.2.143); ser (§5.7.2.168); ser (§5.7.2.169); ser (§5.7.2.170); ser (§5.7.2.171); ser (§5.7.2.172); ser (§5.7.2.173); ser (§5.7.2.174); ser (§5.7.2.175)

Attributes	Description
val (Integer Value)	Specifies that the contents of this attribute will contain an integer number. The contents of this number are interpreted based on the context of the parent XML element. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_UnsignedInt">
   <attribute name="val" type="xsd:unsignedInt" use="required"/>
</complexType>
```

5.7.2.85 intercept (Intercept)

This element specifies the value where the trendline shall cross the y axis. This property shall be supported only when the trendline type is exp, linear, or poly.

Parent Elements
trendline (§5.7.2.212)

Attributes	Description
val (Floating Point Value)	Specifies that the contents of this attribute will contain a floating point number. The contents of this number are interpreted based on the context of the parent XML element. The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Double">
   <attribute name="val" type="xsd:double" use="required"/>
</complexType>
```

5.7.2.86 invertIfNegative (Invert if Negative)

This element specifies the parent element shall invert its colors if the value is negative.

Parent Elements
dPt (§5.7.2.52); ser (§5.7.2.175); ser (§5.7.2.168)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of `on`, `1`, or `true` specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.87 lang (Editing Language)

This element specifies the primary editing language which was use when this chart was last modified.

Parent Elements
chartSpace (§5.7.2.29)

Attributes	Description
val (Language Code)	Specifies a language tag as defined by RFC 3066. See simple type for additional information. The possible values for this attribute are defined by the ST_TextLanguageID simple type (§5.7.3.47).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextLanguageID">
   <attribute name="val" type="ST_TextLanguageID" use="required"/>
</complexType>
```

5.7.2.88 layout (Layout)

This element specifies how the chart element is placed on the chart.

Parent Elements
dispUnitsLbl (§5.7.2.46); dLbl (§5.7.2.47); legend (§5.7.2.94); plotArea (§5.7.2.146); title (§5.7.2.211); trendlineLbl (§5.7.2.213)

Child Elements	Subclause
extLst (Chart Extensibility)	§5.7.2.64
manualLayout (Manual Layout)	§5.7.2.105

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Layout">
   <sequence>
      <element name="manualLayout" type="CT_ManualLayout" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.89 layoutTarget (Layout Target)

This element specifies whether to layout the plot area by its inside (not including axis and axis labels) or outside (including axis and axis labels).

Parent Elements
manualLayout (§5.7.2.105)

Attributes	Description
val (Layout Target Value)	Specifies the layout target value. The possible values for this attribute are defined by the ST_LayoutTarget simple type (§5.7.3.21).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LayoutTarget">
   <attribute name="val" type="ST_LayoutTarget" default="outer"/>
</complexType>
```

5.7.2.90 lblAlgn (Label Alignment)

This element specifies the text alignment for the tick labels on the axis.

Parent Elements
catAx (§5.7.2.25)

Attributes	Description
val (Label Alignment Value)	Specifies the label alignment. The possible values for this attribute are defined by the ST_LblAlgn simple type (§5.7.3.22).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LblAlgn">
   <attribute name="val" type="ST_LblAlgn" use="required"/>
</complexType>
```

5.7.2.91 lblOffset (Label Offset)

This element specifies the distance of labels from the axis.

Parent Elements
catAx (§5.7.2.25); dateAx (§5.7.2.39)

Attributes	Description
val (Label Offset Value)	Specifies the contents of this attribute will contain an integer between 0 and 1000. The possible values for this attribute are defined by the ST_LblOffset simple type (§5.7.3.23).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LblOffset">
   <attribute name="val" type="ST_LblOffset" default="100"/>
</complexType>
```

5.7.2.92 leaderLines (Leader Lines)

This element specifies the leader lines for data labels.

Parent Elements
dLbls (§5.7.2.49)

Child Elements	Subclause
spPr (Shape Properties)	§5.7.2.198

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ChartLines">
   <sequence>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.93 legacyDrawingHF (Legacy Drawing for Headers and Footers)

This element specifies the VML Drawing part that contains any pictures used in the header or footer of the chart.

Parent Elements
printSettings (§5.7.2.149)

Attributes	Description
id (Relationship Reference) Namespace: .../officeDocument/2006/relationships	Specifies the relationship ID for the relationship for this Chart, Chart Drawing, or VML Drawing part. The type of relationship needed is specified by the parent element. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RelId">
   <attribute ref="r:id" use="required"/>
</complexType>
```

5.7.2.94 legend (Legend)

This element specifies the legend.

Parent Elements
chart (§5.7.2.27)

Child Elements	Subclause
extLst (Chart Extensibility)	§5.7.2.64
layout (Layout)	§5.7.2.88
legendEntry (Legend Entry)	§5.7.2.95
legendPos (Legend Position)	§5.7.2.96
overlay (Overlay)	§5.7.2.133
spPr (Shape Properties)	§5.7.2.198
txPr (Text Properties)	§5.7.2.217

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Legend">
   <sequence>
      <element name="legendPos" type="CT_LegendPos" minOccurs="0" maxOccurs="1"/>
      <element name="legendEntry" type="CT_LegendEntry" minOccurs="0" maxOccurs="unbounded"/>
      <element name="layout" type="CT_Layout" minOccurs="0" maxOccurs="1"/>
      <element name="overlay" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
      <element name="txPr" type="a:CT_TextBody" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.95 legendEntry (Legend Entry)

This element specifies a legend entry.

Parent Elements
legend (§5.7.2.94)

Child Elements	Subclause
delete (Delete)	§5.7.2.40

Child Elements	Subclause
extLst (Chart Extensibility)	§5.7.2.64
idx (Index)	§5.7.2.84
txPr (Text Properties)	§5.7.2.217

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LegendEntry">
   <sequence>
      <element name="idx" type="CT_UnsignedInt" minOccurs="1" maxOccurs="1"/>
      <choice>
         <element name="delete" type="CT_Boolean" minOccurs="1" maxOccurs="1"/>
         <group ref="EG_LegendEntryData" minOccurs="1" maxOccurs="1"/>
      </choice>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.96 legendPos (Legend Position)

This element specifies the position of the legend.

Parent Elements
legend (§5.7.2.94)

Attributes	Description
val (Legend Position Value)	Specifies the position of the legend. The possible values for this attribute are defined by the ST_LegendPos simple type (§5.7.3.24).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LegendPos">
   <attribute name="val" type="ST_LegendPos" default="r"/>
</complexType>
```

5.7.2.97 line3DChart (3D Line Charts)

This element contains the 3-D line chart series.

Parent Elements
plotArea (§5.7.2.146)

Child Elements	Subclause
axId (Axis ID)	§5.7.2.9

Child Elements	Subclause
dLbls (Data Labels)	§5.7.2.49
dropLines (Drop Lines)	§5.7.2.53
extLst (Chart Extensibility)	§5.7.2.64
gapDepth (Gap Depth)	§5.7.2.74
grouping (Grouping)	§5.7.2.76
ser (Line Chart Series)	§5.7.2.169
varyColors (Vary Colors by Point)	§5.7.2.228

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Line3DChart">
   <sequence>
      <group ref="EG_LineChartShared" minOccurs="1" maxOccurs="1"/>
      <element name="gapDepth" type="CT_GapAmount" minOccurs="0" maxOccurs="1"/>
      <element name="axId" type="CT_UnsignedInt" minOccurs="3" maxOccurs="3"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.98 lineChart (Line Charts)

This element contains the 2-D line chart series.

Parent Elements
plotArea (§5.7.2.146)

Child Elements	Subclause
axId (Axis ID)	§5.7.2.9
dLbls (Data Labels)	§5.7.2.49
dropLines (Drop Lines)	§5.7.2.53
extLst (Chart Extensibility)	§5.7.2.64
grouping (Grouping)	§5.7.2.76
hiLowLines (High Low Lines)	§5.7.2.80
marker (Show Marker)	§5.7.2.107
ser (Line Chart Series)	§5.7.2.169
smooth (Smoothing)	§5.7.2.195
upDownBars (Up/Down Bars)	§5.7.2.219
varyColors (Vary Colors by Point)	§5.7.2.228

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LineChart">
   <sequence>
      <group ref="EG_LineChartShared" minOccurs="1" maxOccurs="1"/>
      <element name="hiLowLines" type="CT_ChartLines" minOccurs="0" maxOccurs="1"/>
      <element name="upDownBars" type="CT_UpDownBars" minOccurs="0" maxOccurs="1"/>
      <element name="marker" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="smooth" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="axId" type="CT_UnsignedInt" minOccurs="2" maxOccurs="2"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.99 logBase (Logarithmic Base)

This element specifies the logarithmic base for a logarithmic axis.

Parent Elements
scaling (§5.7.2.161)

Attributes	Description
val (Logarithmic Base Value)	Specifies the contents of this attribute will contain a floating point value greater than or equal to 2. The possible values for this attribute are defined by the ST_LogBase simple type (§5.7.3.25).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LogBase">
   <attribute name="val" type="ST_LogBase" use="required"/>
</complexType>
```

5.7.2.100 lvl (Level)

This element specifies data for a single level of labels for a category axis.

Parent Elements
multiLvlStrCache (§5.7.2.115)

Child Elements	Subclause
pt (String Point)	§5.7.2.152

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Lvl">
   <sequence>
      <element name="pt" type="CT_StrVal" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.7.2.101 majorGridlines (Major Gridlines)

This element specifies major gridlines.

Parent Elements
catAx (§5.7.2.25); dateAx (§5.7.2.39); serAx (§5.7.2.176); valAx (§5.7.2.227)

Child Elements	Subclause
spPr (Shape Properties)	§5.7.2.198

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ChartLines">
   <sequence>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.102 majorTickMark (Major Tick Mark)

This element specifies the major tick marks.

Parent Elements
catAx (§5.7.2.25); dateAx (§5.7.2.39); serAx (§5.7.2.176); valAx (§5.7.2.227)

Attributes	Description
val (Tick Mark Value)	Specifies the minor tick mark position. The possible values for this attribute are defined by the ST_TickMark simple type (§5.7.3.49).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TickMark">
   <attribute name="val" type="ST_TickMark" default="cross"/>
</complexType>
```

5.7.2.103 majorTimeUnit (Major Time Unit)

This element specifies the time unit for major tick marks.

Parent Elements
dateAx (§5.7.2.39)

Attributes	Description
val (Time Unit Value)	Specifies the time unit for the tick marks. The possible values for this attribute are defined by the ST_TimeUnit simple type (§5.7.3.50).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TimeUnit">
   <attribute name="val" type="ST_TimeUnit" default="days"/>
</complexType>
```

5.7.2.104 majorUnit (Major Unit)

This element specifies the distance between major ticks.

Parent Elements
dateAx (§5.7.2.39); valAx (§5.7.2.227)

Attributes	Description
val (Major Unit Value)	Specifies the contents of this attribute will contain a positive floating point number. The possible values for this attribute are defined by the ST_AxisUnit simple type (§5.7.3.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AxisUnit">
   <attribute name="val" type="ST_AxisUnit" use="required"/>
</complexType>
```

5.7.2.105 manualLayout (Manual Layout)

This element specifies the exact position of a chart element.

Parent Elements
layout (§5.7.2.88)

Child Elements	Subclause
extLst (Chart Extensibility)	§5.7.2.64
h (Height)	§5.7.2.78

Child Elements	Subclause
hMode (Height Mode)	§5.7.2.81
layoutTarget (Layout Target)	§5.7.2.89
w (Width)	§5.7.2.230
wMode (Width Mode)	§5.7.2.232
x (Left)	§5.7.2.233
xMode (Left Mode)	§5.7.2.234
y (Top)	§5.7.2.236
yMode (Top Mode)	§5.7.2.237

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ManualLayout">
   <sequence>
      <element name="layoutTarget" type="CT_LayoutTarget" minOccurs="0" maxOccurs="1"/>
      <element name="xMode" type="CT_LayoutMode" minOccurs="0" maxOccurs="1"/>
      <element name="yMode" type="CT_LayoutMode" minOccurs="0" maxOccurs="1"/>
      <element name="wMode" type="CT_LayoutMode" minOccurs="0" maxOccurs="1"/>
      <element name="hMode" type="CT_LayoutMode" minOccurs="0" maxOccurs="1"/>
      <element name="x" type="CT_Double" minOccurs="0" maxOccurs="1"/>
      <element name="y" type="CT_Double" minOccurs="0" maxOccurs="1"/>
      <element name="w" type="CT_Double" minOccurs="0" maxOccurs="1"/>
      <element name="h" type="CT_Double" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.106 marker (Marker)

This element specifies a data marker.

Parent Elements
dPt (§5.7.2.52); pivotFmt (§5.7.2.143); ser (§5.7.2.172); ser (§5.7.2.173); ser (§5.7.2.169)

Child Elements	Subclause
extLst (Chart Extensibility)	§5.7.2.64
size (Size)	§5.7.2.193
spPr (Shape Properties)	§5.7.2.198
symbol (Symbol)	§5.7.2.206

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Marker">
   <sequence>
      <element name="symbol" type="CT_MarkerStyle" minOccurs="0" maxOccurs="1"/>
      <element name="size" type="CT_MarkerSize" minOccurs="0" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.107 marker (Show Marker)

This element is a Boolean that, when `true`, specifies that the marker shall be shown.

Parent Elements
lineChart (§5.7.2.98)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.108 max (Maximum)

This element specifies the maximum value of the axis.

Parent Elements
scaling (§5.7.2.161)

Attributes	Description
val (Floating Point Value)	Specifies that the contents of this attribute will contain a floating point number. The contents of this number are interpreted based on the context of the parent XML element.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Double">
   <attribute name="val" type="xsd:double" use="required"/>
</complexType>
```

5.7.2.109 min (Minimum)

This element specifies the minimum value of the axis.

Parent Elements
scaling (§5.7.2.161)

Attributes	Description
val (Floating Point Value)	Specifies that the contents of this attribute will contain a floating point number. The contents of this number are interpreted based on the context of the parent XML element. The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Double">
   <attribute name="val" type="xsd:double" use="required"/>
</complexType>
```

5.7.2.110 minorGridlines (Minor Gridlines)

This element specifies the minor gridlines.

Parent Elements
catAx (§5.7.2.25); dateAx (§5.7.2.39); serAx (§5.7.2.176); valAx (§5.7.2.227)

Child Elements	Subclause
spPr (Shape Properties)	§5.7.2.198

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ChartLines">
   <sequence>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.111 minorTickMark (Minor Tick Mark)

This element specifies the minor tick marks for the axis.

Parent Elements
catAx (§5.7.2.25); dateAx (§5.7.2.39); serAx (§5.7.2.176); valAx (§5.7.2.227)

Attributes	Description
val (Tick Mark Value)	Specifies the minor tick mark position. The possible values for this attribute are defined by the ST_TickMark simple type (§5.7.3.49).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TickMark">
   <attribute name="val" type="ST_TickMark" default="cross"/>
</complexType>
```

5.7.2.112 minorTimeUnit (Minor Time Unit)

This element specifies the time unit for the minor tick marks.

Parent Elements
dateAx (§5.7.2.39)

Attributes	Description
val (Time Unit Value)	Specifies the time unit for the tick marks. The possible values for this attribute are defined by the ST_TimeUnit simple type (§5.7.3.50).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TimeUnit">
   <attribute name="val" type="ST_TimeUnit" default="days"/>
</complexType>
```

5.7.2.113 minorUnit (Minor Unit)

This element specifies the distance between minor tick marks.

Parent Elements
dateAx (§5.7.2.39); valAx (§5.7.2.227)

Attributes	Description
val (Major Unit Value)	Specifies the contents of this attribute will contain a positive floating point number. The possible values for this attribute are defined by the ST_AxisUnit simple type (§5.7.3.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AxisUnit">
   <attribute name="val" type="ST_AxisUnit" use="required"/>
</complexType>
```

5.7.2.114 minus (Minus)

This element specifies the error bar value in the negative direction. It shall be used only when the errValType is cust.

Parent Elements
errBars (§5.7.2.55)

Child Elements	Subclause
numLit (Number Literal)	§5.7.2.123
numRef (Number Reference)	§5.7.2.124

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NumDataSource">
   <sequence>
      <choice minOccurs="1" maxOccurs="1">
         <element name="numRef" type="CT_NumRef" minOccurs="1" maxOccurs="1"/>
         <element name="numLit" type="CT_NumData" minOccurs="1" maxOccurs="1"/>
      </choice>
   </sequence>
</complexType>
```

5.7.2.115 multiLvlStrCache (Multi Level String Cache)

This element specifies the last data shown on the chart for a category axis.

Parent Elements
multiLvlStrRef (§5.7.2.116)

Child Elements	Subclause
extLst (Chart Extensibility)	§5.7.2.64
lvl (Level)	§5.7.2.100
ptCount (Point Count)	§5.7.2.153

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MultiLvlStrData">
    <sequence>
        <element name="ptCount" type="CT_UnsignedInt" minOccurs="0" maxOccurs="1"/>
        <element name="lvl" type="CT_Lvl" minOccurs="0" maxOccurs="unbounded"/>
        <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
    </sequence>
</complexType>
```

5.7.2.116 multiLvlStrRef (Multi Level String Reference)

This element specifies a reference to data for the category axis with a cache of the last values used.

Parent Elements
cat (§5.7.2.24); xVal (§5.7.2.235)

Child Elements	Subclause
extLst (Chart Extensibility)	§5.7.2.64
f (Formula)	§5.7.2.65
multiLvlStrCache (Multi Level String Cache)	§5.7.2.115

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MultiLvlStrRef">
    <sequence>
        <element name="f" type="xsd:string" minOccurs="1" maxOccurs="1"/>
        <element name="multiLvlStrCache" type="CT_MultiLvlStrData" minOccurs="0" maxOccurs="1"/>
        <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
    </sequence>
</complexType>
```

5.7.2.117 name (Trendline Name)

This element specifies the name of the trendline.

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
trendline (§5.7.2.212)

5.7.2.118 name (Pivot Name)

This element specifies the name of the pivot table to get the data for the chart from.

The possible values for this element are defined by the ST_Xstring simple type (§5.7.3.52).

Parent Elements
pivotSource (§5.7.2.145)

5.7.2.119 noEndCap (No End Cap)

This element specifies an end cap is not drawn on the error bars.

Parent Elements
errBars (§5.7.2.55)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.120 noMultiLvlLbl (No Multi-level Labels)

This element specifies the labels shall be shown as flat text. If this element is not included or is set to false, then the labels shall be drawn as a hierarchy.

Parent Elements
catAx (§5.7.2.25)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is

Attributes	Description
	omitted. A value of `off`, `0`, or `false` specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.121 numCache (Number Cache)

This element specifies the last data shown on the chart for a series.

Parent Elements
numRef (§5.7.2.124)

Child Elements	Subclause
extLst (Chart Extensibility)	§5.7.2.64
formatCode (Format Code)	§5.7.2.71
pt (Numeric Point)	§5.7.2.151
ptCount (Point Count)	§5.7.2.153

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NumData">
   <sequence>
      <element name="formatCode" type="ST_Xstring" minOccurs="0" maxOccurs="1"/>
      <element name="ptCount" type="CT_UnsignedInt" minOccurs="0" maxOccurs="1"/>
      <element name="pt" type="CT_NumVal" minOccurs="0" maxOccurs="unbounded"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.122 numFmt (Number Format)

This element specifies number formatting for the parent element.

Parent Elements
catAx (§5.7.2.25); dateAx (§5.7.2.39); dLbl (§5.7.2.47); dLbls (§5.7.2.49); serAx (§5.7.2.176); trendlineLbl (§5.7.2.213); valAx (§5.7.2.227)

Attributes	Description

Attributes	Description
formatCode (Number Format Code)	This element specifies a string representing the format code to apply. For more information see the SpreadsheetML numFmt element's (§3.8.30) formatCode attribute. The possible values for this attribute are defined by the ST_Xstring simple type (§5.7.3.52).
sourceLinked (Linked to Source)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NumFmt">
   <attribute name="formatCode" type="ST_Xstring" use="required"/>
   <attribute name="sourceLinked" type="xsd:boolean"/>
</complexType>
```

5.7.2.123 numLit (Number Literal)

This element specifies a set of numbers used for the parent element.

Parent Elements
bubbleSize (§5.7.2.22); cat (§5.7.2.24); minus (§5.7.2.114); plus (§5.7.2.148); val (§5.7.2.225); xVal (§5.7.2.235); yVal (§5.7.2.238)

Child Elements	Subclause
extLst (Chart Extensibility)	§5.7.2.64
formatCode (Format Code)	§5.7.2.71
pt (Numeric Point)	§5.7.2.151
ptCount (Point Count)	§5.7.2.153

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NumData">
   <sequence>
      <element name="formatCode" type="ST_Xstring" minOccurs="0" maxOccurs="1"/>
      <element name="ptCount" type="CT_UnsignedInt" minOccurs="0" maxOccurs="1"/>
      <element name="pt" type="CT_NumVal" minOccurs="0" maxOccurs="unbounded"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.124 numRef (Number Reference)

This element specifies a reference to numeric data with a cache of the last values used.

Parent Elements
bubbleSize (§5.7.2.22); cat (§5.7.2.24); minus (§5.7.2.114); plus (§5.7.2.148); val (§5.7.2.225); xVal (§5.7.2.235); yVal (§5.7.2.238)

Child Elements	Subclause
extLst (Chart Extensibility)	§5.7.2.64
f (Formula)	§5.7.2.65
numCache (Number Cache)	§5.7.2.121

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NumRef">
   <sequence>
      <element name="f" type="xsd:string" minOccurs="1" maxOccurs="1"/>
      <element name="numCache" type="CT_NumData" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.125 oddFooter (Odd Footer)

This element specifies the footer to use on odd numbered pages.

The possible values for this element are defined by the ST_Xstring simple type (§5.7.3.52).

Parent Elements
headerFooter (§5.7.2.79)

5.7.2.126 oddHeader (Odd Header)

This element specifies the header to use on odd numbered pages.

The possible values for this element are defined by the ST_Xstring simple type (§5.7.3.52).

Parent Elements
headerFooter (§5.7.2.79)

5.7.2.127 ofPieChart (Pie of Pie or Bar of Pie Charts)

This element contains the pie of pie or bar of pie series on this chart. Only the first series shall be displayed. The splitType element shall determine whether the splitPos and custSplit elements apply.

Parent Elements

Parent Elements
plotArea (§5.7.2.146)

Child Elements	Subclause
custSplit (Custom Split)	§5.7.2.35
dLbls (Data Labels)	§5.7.2.49
extLst (Chart Extensibility)	§5.7.2.64
gapWidth (Gap Width)	§5.7.2.75
ofPieType (Pie of Pie or Bar of Pie Type)	§5.7.2.128
secondPieSize (Second Pie Size)	§5.7.2.165
ser (Pie Chart Series)	§5.7.2.170
serLines (Series Lines)	§5.7.2.177
splitPos (Split Position)	§5.7.2.196
splitType (Split Type)	§5.7.2.197
varyColors (Vary Colors by Point)	§5.7.2.228

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OfPieChart">
   <sequence>
      <element name="ofPieType" type="CT_OfPieType" minOccurs="1" maxOccurs="1"/>
      <group ref="EG_PieChartShared" minOccurs="1" maxOccurs="1"/>
      <element name="gapWidth" type="CT_GapAmount" minOccurs="0" maxOccurs="1"/>
      <element name="splitType" type="CT_SplitType" minOccurs="0" maxOccurs="1"/>
      <element name="splitPos" type="CT_Double" minOccurs="0" maxOccurs="1"/>
      <element name="custSplit" type="CT_CustSplit" minOccurs="0" maxOccurs="1"/>
      <element name="secondPieSize" type="CT_SecondPieSize" minOccurs="0" maxOccurs="1"/>
      <element name="serLines" type="CT_ChartLines" minOccurs="0" maxOccurs="unbounded"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.128 ofPieType (Pie of Pie or Bar of Pie Type)

This element specifies whether this chart is pie of pie or bar of pie.

Parent Elements
ofPieChart (§5.7.2.127)

Attributes	Description
val (Pie of Pie or Bar of Pie Type Value)	Specifies the type of pie of pie or bar of pie chart.

Attributes	Description
	The possible values for this attribute are defined by the ST_OfPieType simple type (§5.7.3.28).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OfPieType">
   <attribute name="val" type="ST_OfPieType" default="pie"/>
</complexType>
```

5.7.2.129 order (Order)

This element specifies the order of the series in the collection. It is 0 based.

Parent Elements
ser (§5.7.2.168); ser (§5.7.2.169); ser (§5.7.2.170); ser (§5.7.2.171); ser (§5.7.2.172); ser (§5.7.2.173); ser (§5.7.2.174); ser (§5.7.2.175)

Attributes	Description
val (Integer Value)	Specifies that the contents of this attribute will contain an integer number. The contents of this number are interpreted based on the context of the parent XML element. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_UnsignedInt">
   <attribute name="val" type="xsd:unsignedInt" use="required"/>
</complexType>
```

5.7.2.130 order (Polynomial Trendline Order)

This element specifies the order of the polynomial trend line. It is ignored for other trend line types.

Parent Elements
trendline (§5.7.2.212)

Attributes	Description
val (Order Value)	Specifies that the contents of this attribute will contain an integer between 2 and 6. The possible values for this attribute are defined by the ST_Order simple type (§5.7.3.29).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Order">
   <attribute name="val" type="ST_Order" default="2"/>
</complexType>
```

5.7.2.131 orientation (Axis Orientation)

This element specifies the stretching and stacking of the picture on the data point, series, wall, or floor.

Parent Elements
scaling (§5.7.2.161)

Attributes	Description
val (Orientation Value)	Specifies the orientation of the axis. The possible values for this attribute are defined by the ST_Orientation simple type (§5.7.3.30).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Orientation">
   <attribute name="val" type="ST_Orientation" default="minMax"/>
</complexType>
```

5.7.2.132 overlap (Overlap)

This element specifies how much bars and columns shall overlap on 2-D charts.

Parent Elements
barChart (§5.7.2.16)

Attributes	Description
val (Overlap Value)	Specifies the contents of this attribute will contain an integer between -100 and 100. The possible values for this attribute are defined by the ST_Overlap simple type (§5.7.3.31).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Overlap">
   <attribute name="val" type="ST_Overlap" default="0"/>
</complexType>
```

5.7.2.133 overlay (Overlay)

This element specifies that other chart elements shall be allowed to overlap this chart element.

Parent Elements
legend (§5.7.2.94); title (§5.7.2.211)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.134 pageMargins (Page Margins)

This element specifies the page margins for a chart.

Parent Elements
printSettings (§5.7.2.149)

Attributes	Description
b (Bottom)	Specifies the contents of this attribute will contain the bottom page margin in inches. The possible values for this attribute are defined by the XML Schema double datatype.
footer (Footer)	Specifies the contents of this attribute will contain the footer margin in inches. The possible values for this attribute are defined by the XML Schema double datatype.
header (Header)	Specifies the contents of this attribute will contain the header margin in inches. The possible values for this attribute are defined by the XML Schema double datatype.
l (Left)	Specifies the contents of this attribute will contain the left page margin in inches. The possible values for this attribute are defined by the XML Schema double datatype.
r (Right)	Specifies the contents of this attribute will contain the right page margin in inches. The possible values for this attribute are defined by the XML Schema double datatype.

Attributes	Description
t (Top)	Specifies the contents of this attribute will contain the top page margin in inches. The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PageMargins">
   <attribute name="l" type="xsd:double" use="required"/>
   <attribute name="r" type="xsd:double" use="required"/>
   <attribute name="t" type="xsd:double" use="required"/>
   <attribute name="b" type="xsd:double" use="required"/>
   <attribute name="header" type="xsd:double" use="required"/>
   <attribute name="footer" type="xsd:double" use="required"/>
</complexType>
```

5.7.2.135 pageSetup (Page Setup)

This element defines the page setup for the chart.

Parent Elements
printSettings (§5.7.2.149)

Attributes	Description
blackAndWhite (Black and White)	Specifies the page shall print in black and white. The possible values for this attribute are defined by the XML Schema boolean datatype.
copies (Copies)	Specifies the number of copies that shall be printed. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
draft (Draft)	Specifies the page shall be printed in draft mode. The possible values for this attribute are defined by the XML Schema boolean datatype.
firstPageNumber (First Page Number)	Specifies the page number. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
horizontalDpi (Horizontal DPI)	Specifies the horizontal resolution to print in dots per inch. The possible values for this attribute are defined by the XML Schema int datatype.
orientation (Orientation)	Specifies the orientation of the paper. The possible values for this attribute are defined by the ST_PageSetupOrientation simple type (§5.7.3.32).

Attributes	Description
paperSize (Page Size)	Specifies the paper size according to the following table.

Paper Size	Width	Height	Value
Letter paper	8.5 in.	11 in.	1
Legal paper	8.5 in.	14 in.	5
Standard paper	10 in.	11 in.	45
Standard paper	10 in.	14 in.	16
Standard paper	11 in.	17 in.	17
Standard paper	15 in.	11 in.	46
Standard paper	9 in.	11 in.	44
SuperA/SuperA/A4 paper	227 mm	356 mm	57
A2 paper	420 mm	594 mm	66
A3 paper	297 mm	420 mm	8
A3 extra paper	322 mm	445 mm	63
A3 extra transverse paper	322 mm	445 mm	68
A3 transverse paper	297 mm	420 mm	67
A4 paper	210 mm	297 mm	9
A4 extra paper	236 mm	322 mm	53
A4 plus paper	210 mm	330 mm	60
A4 transverse paper	210 mm	297 mm	55
A4 small paper	210 mm	297 mm	10
A5 paper	148 mm	210 mm	11
A5 extra paper	174 mm	235 mm	64
A5 transverse paper	148 mm	210 mm	61
SuperB/SuperB/A3 paper	305 mm	487 mm	58
B4 paper	250 mm	353 mm	12

Attributes	Description			
	B5 paper	176 mm	250 mm	13
	ISO B5 extra paper	201 mm	276 mm	65
	JIS B5 transverse paper	182 mm	257 mm	62
	C paper	17 in.	22 in.	24
	D paper	22 in.	34 in.	25
	#10 envelope	4.125 in.	9.5 in.	20
	#11 envelope	4.5 in.	10.375 in.	21
	#12 envelope	4.75 in.	11 in.	22
	#14 envelope	5 in.	11.5 in.	23
	#9 envelope	3.875 in.	8.875 in.	19
	B4 envelope	250 mm	353 mm	33
	B5 envelope	176 mm	250 mm	34
	B6 envelope	176 mm	125 mm	35
	C3 envelope	324 mm	458 mm	29
	C4 envelope	229 mm	324 mm	30
	C5 envelope	162 mm	229 mm	28
	C6 envelope	114 mm	162 mm	31
	C65 envelope	114 mm	229 mm	32
	DL envelope	110 mm	220 mm	27
	Invite envelope	220 mm	220 mm	47
	Italy envelope	110 mm	230 mm	36
	Monarch envelope	3.875 in.	7.5 in.).	37
	6 3/4 envelope	3.625 in.	6.5 in.	38
	E paper	34 in.	44 in.	26

Attributes	Description			
	Executive paper	7.25 in.	10.5 in.	7
	German legal fanfold	8.5 in.	13 in.	41
	German standard fanfold	8.5 in.	12 in.	40
	US standard fanfold	14.875 in.	11 in.	39
	Folio paper	8.5 in.	13 in.	14
	ISO B4	250 mm	353 mm	42
	Japanese double postcard	200 mm	148 mm	43
	Ledger paper	17 in.	11 in.	4
	Legal extra paper	9.275 in.	15 in.	51
	Letter extra paper	9.275 in.	12 in.	50
	Letter extra transverse paper	9.275 in.	12 in.	56
	Letter plus paper	8.5 in.	12.69 in.	59
	Letter transverse paper	8.275 in.	11 in.	54
	Letter small paper	8.5 in.	11 in.	2
	Note paper	8.5 in.	11 in.	18
	Quarto paper	215 mm	275 mm	15
	Statement paper	5.5 in.	8.5 in.	6
	Tabloid paper	11 in.	17 in.	3
	Tabloid extra paper	11.69 in.	18 in.	52
	The possible values for this attribute are defined by the XML Schema unsignedInt datatype.			
useFirstPageNumber (Use First Page Number)	Specifies to use the first page number instead of automatically generating a page number. The possible values for this attribute are defined by the XML Schema boolean datatype.			
verticalDpi	Specifies the vertical resolution to print in dots per inch.			

Attributes	Description
(Vertical DPI)	The possible values for this attribute are defined by the XML Schema int datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PageSetup">
   <attribute name="paperSize" type="xsd:unsignedInt" use="optional" default="1"/>
   <attribute name="firstPageNumber" type="xsd:unsignedInt" use="optional" default="1"/>
   <attribute name="orientation" type="ST_PageSetupOrientation" use="optional" default="default"/>
   <attribute name="blackAndWhite" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="draft" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="useFirstPageNumber" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="horizontalDpi" type="xsd:int" use="optional" default="600"/>
   <attribute name="verticalDpi" type="xsd:int" use="optional" default="600"/>
   <attribute name="copies" type="xsd:unsignedInt" use="optional" default="1"/>
</complexType>
```

5.7.2.136 period (Period)

This element specifies the period of the trend line for a moving average trend line. It is ignored for other trend line types.

Parent Elements
trendline (§5.7.2.212)

Attributes	Description
val (Period Value)	Specifies the contents of this attribute will contain an integer between 2 and 255. The possible values for this attribute are defined by the ST_Period simple type (§5.7.3.33).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Period">
   <attribute name="val" type="ST_Period" default="2"/>
</complexType>
```

5.7.2.137 perspective (Perspective)

This element specifies the field of view angle for the 3-D chart. This element is ignored if Right Angle Axes is true.

Parent Elements
view3D (§5.7.2.229)

Attributes	Description

Attributes	Description
val (Perspective Value)	Specifies the contents of this attribute will contain an integer between 0 and 100. The possible values for this attribute are defined by the ST_Perspective simple type (§5.7.3.34).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Perspective">
   <attribute name="val" type="ST_Perspective" default="30"/>
</complexType>
```

5.7.2.138 pictureFormat (Picture Format)

This element specifies the stretching and stacking of the picture on the data point, series, wall, or floor.

Parent Elements
pictureOptions (§5.7.2.139)

Attributes	Description
val (Picture Format Value)	Specifies the stretching and stacking of the picture. The possible values for this attribute are defined by the ST_PictureFormat simple type (§5.7.3.35).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PictureFormat">
   <attribute name="val" type="ST_PictureFormat" use="required"/>
</complexType>
```

5.7.2.139 pictureOptions (Picture Options)

This element specifies the picture to be used on the data point, series, wall, or floor.

Parent Elements
backWall (§5.7.2.11); dPt (§5.7.2.52); floor (§5.7.2.69); ser (§5.7.2.175); ser (§5.7.2.174); sideWall (§5.7.2.192)

Child Elements	Subclause
applyToEnd (Apply to End)	§5.7.2.1
applyToFront (Apply To Front)	§5.7.2.2
applyToSides (Apply To Sides)	§5.7.2.3
pictureFormat (Picture Format)	§5.7.2.138
pictureStackUnit (Picture Stack Unit)	§5.7.2.140

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PictureOptions">
   <sequence>
      <element name="applyToFront" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="applyToSides" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="applyToEnd" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="pictureFormat" type="CT_PictureFormat" minOccurs="0" maxOccurs="1"/>
      <element name="pictureStackUnit" type="CT_PictureStackUnit" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.140 pictureStackUnit (Picture Stack Unit)

This element specifies the unit for each picture on the chart. This element applies only if the Picture Format is Stack and Scale.

Parent Elements
pictureOptions (§5.7.2.139)

Attributes	Description
val (Picture Stack Unit)	Specifies the contents of this attribute will contain a floating point number. The possible values for this attribute are defined by the ST_PictureStackUnit simple type (§5.7.3.36).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PictureStackUnit">
   <attribute name="val" type="ST_PictureStackUnit" use="required"/>
</complexType>
```

5.7.2.141 pie3DChart (3D Pie Charts)

This element contains the 3-D pie series for this chart.

Parent Elements
plotArea (§5.7.2.146)

Child Elements	Subclause
dLbls (Data Labels)	§5.7.2.49
extLst (Chart Extensibility)	§5.7.2.64
ser (Pie Chart Series)	§5.7.2.170
varyColors (Vary Colors by Point)	§5.7.2.228

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Pie3DChart">
   <sequence>
      <group ref="EG_PieChartShared" minOccurs="1" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.142 pieChart (Pie Charts)

This element contains the 2-D pie series for this chart.

Parent Elements
plotArea (§5.7.2.146)

Child Elements	Subclause
dLbls (Data Labels)	§5.7.2.49
extLst (Chart Extensibility)	§5.7.2.64
firstSliceAng (First Slice Angle)	§5.7.2.68
ser (Pie Chart Series)	§5.7.2.170
varyColors (Vary Colors by Point)	§5.7.2.228

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PieChart">
   <sequence>
      <group ref="EG_PieChartShared" minOccurs="1" maxOccurs="1"/>
      <element name="firstSliceAng" type="CT_FirstSliceAng" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.143 pivotFmt (Pivot Format)

This element contains a set of formatting to be applied to the chart that is based on a pivotTable.

Parent Elements
pivotFmts (§5.7.2.144)

Child Elements	Subclause
dLbl (Data Label)	§5.7.2.47
extLst (Chart Extensibility)	§5.7.2.64
idx (Index)	§5.7.2.84
marker (Marker)	§5.7.2.106

Child Elements	Subclause
spPr (Shape Properties)	§5.7.2.198
txPr (Text Properties)	§5.7.2.217

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PivotFmt">
   <sequence>
      <element name="idx" type="CT_UnsignedInt" minOccurs="1" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
      <element name="txPr" type="a:CT_TextBody" minOccurs="0" maxOccurs="1"/>
      <element name="marker" type="CT_Marker" minOccurs="0" maxOccurs="1"/>
      <element name="dLbl" type="CT_DLbl" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.144 pivotFmts (Pivot Formats)

This element contains a collection of formatting bands for a surface chart indexed from low to high.

Parent Elements
chart (§5.7.2.27)

Child Elements	Subclause
pivotFmt (Pivot Format)	§5.7.2.143

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PivotFmts">
   <sequence>
      <element name="pivotFmt" type="CT_PivotFmt" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.7.2.145 pivotSource (Pivot Source)

This element specifies the source pivot table for a pivot chart.

Parent Elements
chartSpace (§5.7.2.29)

Child Elements	Subclause
extLst (Chart Extensibility)	§5.7.2.64
fmtId (Format ID)	§5.7.2.70
name (Pivot Name)	§5.7.2.118

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PivotSource">
   <sequence>
      <element name="name" type="ST_Xstring" minOccurs="1" maxOccurs="1"/>
      <element name="fmtId" type="CT_UnsignedInt" minOccurs="1" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.7.2.146 plotArea (Plot Area)

This element specifies the plot area of the chart.

Parent Elements
chart (§5.7.2.27)

Child Elements	Subclause
area3DChart (3D Area Charts)	§5.7.2.4
areaChart (Area Charts)	§5.7.2.5
bar3DChart (3D Bar Charts)	§5.7.2.15
barChart (Bar Charts)	§5.7.2.16
bubbleChart (Bubble Charts)	§5.7.2.20
catAx (Category Axis Data)	§5.7.2.25
dateAx (Date Axis)	§5.7.2.39
doughnutChart (Doughnut Charts)	§5.7.2.50
dTable (Data Table)	§5.7.2.54
extLst (Chart Extensibility)	§5.7.2.64
layout (Layout)	§5.7.2.88
line3DChart (3D Line Charts)	§5.7.2.97
lineChart (Line Charts)	§5.7.2.98
ofPieChart (Pie of Pie or Bar of Pie Charts)	§5.7.2.127
pie3DChart (3D Pie Charts)	§5.7.2.141
pieChart (Pie Charts)	§5.7.2.142
radarChart (Radar Charts)	§5.7.2.154
scatterChart (Scatter Charts)	§5.7.2.162
serAx (Series Axis)	§5.7.2.176
spPr (Shape Properties)	§5.7.2.198
stockChart (Stock Charts)	§5.7.2.199
surface3DChart (3D Surface Charts)	§5.7.2.204

Child Elements	Subclause
surfaceChart (Surface Charts)	§5.7.2.205
valAx (Value Axis)	§5.7.2.227

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PlotArea">
   <sequence>
      <element name="layout" type="CT_Layout" minOccurs="0" maxOccurs="1"/>
      <choice minOccurs="1" maxOccurs="unbounded">
         <element name="areaChart" type="CT_AreaChart" minOccurs="1" maxOccurs="1"/>
         <element name="area3DChart" type="CT_Area3DChart" minOccurs="1" maxOccurs="1"/>
         <element name="lineChart" type="CT_LineChart" minOccurs="1" maxOccurs="1"/>
         <element name="line3DChart" type="CT_Line3DChart" minOccurs="1" maxOccurs="1"/>
         <element name="stockChart" type="CT_StockChart" minOccurs="1" maxOccurs="1"/>
         <element name="radarChart" type="CT_RadarChart" minOccurs="1" maxOccurs="1"/>
         <element name="scatterChart" type="CT_ScatterChart" minOccurs="1" maxOccurs="1"/>
         <element name="pieChart" type="CT_PieChart" minOccurs="1" maxOccurs="1"/>
         <element name="pie3DChart" type="CT_Pie3DChart" minOccurs="1" maxOccurs="1"/>
         <element name="doughnutChart" type="CT_DoughnutChart" minOccurs="1" maxOccurs="1"/>
         <element name="barChart" type="CT_BarChart" minOccurs="1" maxOccurs="1"/>
         <element name="bar3DChart" type="CT_Bar3DChart" minOccurs="1" maxOccurs="1"/>
         <element name="ofPieChart" type="CT_OfPieChart" minOccurs="1" maxOccurs="1"/>
         <element name="surfaceChart" type="CT_SurfaceChart" minOccurs="1" maxOccurs="1"/>
         <element name="surface3DChart" type="CT_Surface3DChart" minOccurs="1" maxOccurs="1"/>
         <element name="bubbleChart" type="CT_BubbleChart" minOccurs="1" maxOccurs="1"/>
      </choice>
      <choice minOccurs="0" maxOccurs="unbounded">
         <element name="valAx" type="CT_ValAx" minOccurs="1" maxOccurs="1"/>
         <element name="catAx" type="CT_CatAx" minOccurs="1" maxOccurs="1"/>
         <element name="dateAx" type="CT_DateAx" minOccurs="1" maxOccurs="1"/>
         <element name="serAx" type="CT_SerAx" minOccurs="1" maxOccurs="1"/>
      </choice>
      <element name="dTable" type="CT_DTable" minOccurs="0" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.147 plotVisOnly (Plot Visible Only)

This element specifies that only visible cells should be plotted on the chart.

Parent Elements
chart (§5.7.2.27)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element.

Attributes	Description
	A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.148 plus (Plus)

This element specifies the error bar value in the positive direction. It shall be used only when the errValType is cust.

Parent Elements
errBars (§5.7.2.55)

Child Elements	Subclause
numLit (Number Literal)	§5.7.2.123
numRef (Number Reference)	§5.7.2.124

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NumDataSource">
   <sequence>
      <choice minOccurs="1" maxOccurs="1">
         <element name="numRef" type="CT_NumRef" minOccurs="1" maxOccurs="1"/>
         <element name="numLit" type="CT_NumData" minOccurs="1" maxOccurs="1"/>
      </choice>
   </sequence>
</complexType>
```

5.7.2.149 printSettings (Print Settings)

This element specifies the print settings for the chart.

Parent Elements
chartSpace (§5.7.2.29)

Child Elements	Subclause

Child Elements	Subclause
headerFooter (Header and Footer)	§5.7.2.79
legacyDrawingHF (Legacy Drawing for Headers and Footers)	§5.7.2.93
pageMargins (Page Margins)	§5.7.2.134
pageSetup (Page Setup)	§5.7.2.135

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PrintSettings">
   <sequence>
      <element name="headerFooter" type="CT_HeaderFooter" minOccurs="0" maxOccurs="1"/>
      <element name="pageMargins" type="CT_PageMargins" minOccurs="0" maxOccurs="1"/>
      <element name="pageSetup" type="CT_PageSetup" minOccurs="0" maxOccurs="1"/>
      <element name="legacyDrawingHF" type="CT_RelId" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.150 protection (Protection)

This element specifies protection for the chart. If the chart is on a protected worksheet or chart sheet, then these settings shall control how a user is able to interact with the chart.

Parent Elements
chartSpace (§5.7.2.29)

Child Elements	Subclause
chartObject (Chart Object)	§5.7.2.28
data (Data Cannot Be Changed)	§5.7.2.37
formatting (Formatting)	§5.7.2.72
selection (Selection)	§5.7.2.166
userInterface (User Interface)	§5.7.2.220

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Protection">
   <sequence>
      <element name="chartObject" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="data" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="formatting" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="selection" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="userInterface" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.151 pt (Numeric Point)

This element specifies data for a particular data point.

Parent Elements
numCache (§5.7.2.121); numLit (§5.7.2.123)

Child Elements	Subclause
v (Numeric Value)	§5.7.2.224

Attributes	Description
formatCode (Number Format)	A string representing the format code to apply. For more information see see the SpreadsheetML numFmt element's (§3.8.30) formatCode attribute. The possible values for this attribute are defined by the ST_Xstring simple type (§5.7.3.52).
idx (Index)	The index of the series in the collection The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NumVal">
   <sequence>
      <element name="v" type="ST_Xstring" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="idx" type="xsd:unsignedInt" use="required"/>
   <attribute name="formatCode" type="ST_Xstring" use="optional"/>
</complexType>
```

5.7.2.152 pt (String Point)

This element specifies string data for a specific data point.

Parent Elements
lvl (§5.7.2.100); strCache (§5.7.2.200); strLit (§5.7.2.201)

Child Elements	Subclause
v (Text Value)	§5.7.2.223

Attributes	Description
idx (Index)	A 0 based index into a set of points. Represents the data point number this data is for. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_StrVal">
   <sequence>
      <element name="v" type="ST_Xstring" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="idx" type="xsd:unsignedInt" use="required"/>
</complexType>
```

5.7.2.153 ptCount (Point Count)

This element contains the number of values in the cache.

Parent Elements
multiLvlStrCache (§5.7.2.115); numCache (§5.7.2.121); numLit (§5.7.2.123); strCache (§5.7.2.200); strLit (§5.7.2.201)

Attributes	Description
val (Integer Value)	Specifies that the contents of this attribute will contain an integer number. The contents of this number are interpreted based on the context of the parent XML element. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_UnsignedInt">
   <attribute name="val" type="xsd:unsignedInt" use="required"/>
</complexType>
```

5.7.2.154 radarChart (Radar Charts)

This element contains the radar chart series on this chart.

Parent Elements
plotArea (§5.7.2.146)

Child Elements	Subclause
axId (Axis ID)	§5.7.2.9
dLbls (Data Labels)	§5.7.2.49
extLst (Chart Extensibility)	§5.7.2.64
radarStyle (Radar Style)	§5.7.2.155
ser (Radar Chart Series)	§5.7.2.173

Child Elements	Subclause
varyColors (Vary Colors by Point)	§5.7.2.228

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RadarChart">
   <sequence>
      <element name="radarStyle" type="CT_RadarStyle" minOccurs="1" maxOccurs="1"/>
      <element name="varyColors" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="ser" type="CT_RadarSer" minOccurs="0" maxOccurs="unbounded"/>
      <element name="dLbls" type="CT_DLbls" minOccurs="0" maxOccurs="1"/>
      <element name="axId" type="CT_UnsignedInt" minOccurs="2" maxOccurs="2"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.155 radarStyle (Radar Style)

This element specifies what type of radar chart shall be drawn.

Parent Elements
radarChart (§5.7.2.154)

Attributes	Description
val (Radar Style Value)	Specifies the style of the radar chart. The possible values for this attribute are defined by the ST_RadarStyle simple type (§5.7.3.37).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RadarStyle">
   <attribute name="val" type="ST_RadarStyle" default="standard"/>
</complexType>
```

5.7.2.156 rAngAx (Right Angle Axes)

This element specifies that the chart axes are at right angles, rather than drawn in perspective. Applies only to 3-D charts.

Parent Elements
view3D (§5.7.2.229)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property is applied. This is the default value for

Attributes	Description
	this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.157 rich (Rich Text)

This element contains a string with rich text formatting.

Parent Elements
tx (§5.7.2.215)

Child Elements	Subclause
bodyPr (Body Properties)	§5.1.5.1.1
lstStyle (Text List Styles)	§5.1.5.4.12
p (Text Paragraphs)	§5.1.5.2.6

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextBody">
   <sequence>
      <element name="bodyPr" type="CT_TextBodyProperties" minOccurs="1" maxOccurs="1"/>
      <element name="lstStyle" type="CT_TextListStyle" minOccurs="0" maxOccurs="1"/>
      <element name="p" type="CT_TextParagraph" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.7.2.158 rotX (X Rotation)

This element specifies the amount a 3-D chart shall be rotated in the X direction.

Parent Elements
view3D (§5.7.2.229)

Attributes	Description
val (X Rotation Value)	Specifies the contents of this attribute will contain an integer between -90 and 90.

Attributes	Description
	The possible values for this attribute are defined by the ST_RotX simple type (§5.7.3.38).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RotX">
   <attribute name="val" type="ST_RotX" default="0"/>
</complexType>
```

5.7.2.159 rotY (Y Rotation)

This element specifies the amount a 3-D chart shall be rotated in the Y direction.

Parent Elements
view3D (§5.7.2.229)

Attributes	Description
val (Y Rotation Value)	Specifies the contents of this attribute will contain an integer between 0 and 360. The possible values for this attribute are defined by the ST_RotY simple type (§5.7.3.39).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RotY">
   <attribute name="val" type="ST_RotY" default="0"/>
</complexType>
```

5.7.2.160 roundedCorners (Rounded Corners)

This element specifies the chart area shall have rounded corners.

Parent Elements
chartSpace (§5.7.2.29)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.161 scaling (Scaling)

This element contains additional axis settings.

Parent Elements
catAx (§5.7.2.25); dateAx (§5.7.2.39); serAx (§5.7.2.176); valAx (§5.7.2.227)

Child Elements	Subclause
extLst (Chart Extensibility)	§5.7.2.64
logBase (Logarithmic Base)	§5.7.2.99
max (Maximum)	§5.7.2.108
min (Minimum)	§5.7.2.109
orientation (Axis Orientation)	§5.7.2.131

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Scaling">
   <sequence>
      <element name="logBase" type="CT_LogBase" minOccurs="0" maxOccurs="1"/>
      <element name="orientation" type="CT_Orientation" minOccurs="0" maxOccurs="1"/>
      <element name="max" type="CT_Double" minOccurs="0" maxOccurs="1"/>
      <element name="min" type="CT_Double" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.162 scatterChart (Scatter Charts)

This element contains the scatter chart series for this chart.

Parent Elements
plotArea (§5.7.2.146)

Child Elements	Subclause
axId (Axis ID)	§5.7.2.9
dLbls (Data Labels)	§5.7.2.49
extLst (Chart Extensibility)	§5.7.2.64
scatterStyle (Scatter Style)	§5.7.2.163

Child Elements	Subclause
ser (Scatter Chart Series)	§5.7.2.172
varyColors (Vary Colors by Point)	§5.7.2.228

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ScatterChart">
   <sequence>
      <element name="scatterStyle" type="CT_ScatterStyle" minOccurs="1" maxOccurs="1"/>
      <element name="varyColors" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="ser" type="CT_ScatterSer" minOccurs="0" maxOccurs="unbounded"/>
      <element name="dLbls" type="CT_DLbls" minOccurs="0" maxOccurs="1"/>
      <element name="axId" type="CT_UnsignedInt" minOccurs="2" maxOccurs="2"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.163 scatterStyle (Scatter Style)

This element specifies the type of lines for the scatter chart.

Parent Elements
scatterChart (§5.7.2.162)

Attributes	Description
val (Scatter Style Value)	Specifies the style of the scatter chart. The possible values for this attribute are defined by the ST_ScatterStyle simple type (§5.7.3.40).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ScatterStyle">
   <attribute name="val" type="ST_ScatterStyle" default="marker"/>
</complexType>
```

5.7.2.164 secondPiePt (Second Pie Point)

This element specifies a data point that shall be drawn in the second pie or bar in a pie of pie or bar of pie chart.

Parent Elements
custSplit (§5.7.2.35)

Attributes	Description
val (Integer Value)	Specifies that the contents of this attribute will contain an integer number.

Attributes	Description
	The contents of this number are interpreted based on the context of the parent XML element. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_UnsignedInt">
   <attribute name="val" type="xsd:unsignedInt" use="required"/>
</complexType>
```

5.7.2.165 secondPieSize (Second Pie Size)

This element specifies the size of the second pie or bar of a pie of pie chart or a bar of pie chart, as a percentage of the size of the first pie.

Parent Elements
ofPieChart (§5.7.2.127)

Attributes	Description
val (Second Pie Size Value)	Specifies the contents of this attribute will contain an integer between 5 and 200. The possible values for this attribute are defined by the ST_SecondPieSize simple type (§5.7.3.41).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SecondPieSize">
   <attribute name="val" type="ST_SecondPieSize" default="75"/>
</complexType>
```

5.7.2.166 selection (Selection)

This element specifies the chart elements are protected from selection.

Parent Elements
protection (§5.7.2.150)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted.

Attributes	Description
	A value of `off`, `0`, or `false` specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.167 separator (Separator)

This element specifies text that shall be used to separate the parts of a data label. The default is a comma, except for pie charts showing only category name and percentage, when a line break shall be used instead.

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
dLbl (§5.7.2.47); dLbls (§5.7.2.49)

5.7.2.168 ser (Bubble Chart Series)

This element specifies a series on a bubble chart.

Parent Elements
bubbleChart (§5.7.2.20)

Child Elements	Subclause
bubble3D (3D Bubble)	§5.7.2.19
bubbleSize (Bubble Size)	§5.7.2.22
dLbls (Data Labels)	§5.7.2.49
dPt (Data Point)	§5.7.2.52
errBars (Error Bars)	§5.7.2.55
extLst (Chart Extensibility)	§5.7.2.64
idx (Index)	§5.7.2.84
invertIfNegative (Invert if Negative)	§5.7.2.86
order (Order)	§5.7.2.129
spPr (Shape Properties)	§5.7.2.198
trendline (Trendlines)	§5.7.2.212
tx (Series Text)	§5.7.2.216

Child Elements	Subclause
xVal (X Values)	§5.7.2.235
yVal (Y Values)	§5.7.2.238

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BubbleSer">
   <sequence>
      <group ref="EG_SerShared" minOccurs="1" maxOccurs="1"/>
      <element name="invertIfNegative" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="dPt" type="CT_DPt" minOccurs="0" maxOccurs="unbounded"/>
      <element name="dLbls" type="CT_DLbls" minOccurs="0" maxOccurs="1"/>
      <element name="trendline" type="CT_Trendline" minOccurs="0" maxOccurs="unbounded"/>
      <element name="errBars" type="CT_ErrBars" minOccurs="0" maxOccurs="2"/>
      <element name="xVal" type="CT_AxDataSource" minOccurs="0" maxOccurs="1"/>
      <element name="yVal" type="CT_NumDataSource" minOccurs="0" maxOccurs="1"/>
      <element name="bubbleSize" type="CT_NumDataSource" minOccurs="0" maxOccurs="1"/>
      <element name="bubble3D" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.169 ser (Line Chart Series)

This element specifies a series on a line chart.

Parent Elements
line3DChart (§5.7.2.97); lineChart (§5.7.2.98); stockChart (§5.7.2.199)

Child Elements	Subclause
cat (Category Axis Data)	§5.7.2.24
dLbls (Data Labels)	§5.7.2.49
dPt (Data Point)	§5.7.2.52
errBars (Error Bars)	§5.7.2.55
extLst (Chart Extensibility)	§5.7.2.64
idx (Index)	§5.7.2.84
marker (Marker)	§5.7.2.106
order (Order)	§5.7.2.129
smooth (Smoothing)	§5.7.2.195
spPr (Shape Properties)	§5.7.2.198
trendline (Trendlines)	§5.7.2.212
tx (Series Text)	§5.7.2.216
val (Values)	§5.7.2.225

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LineSer">
   <sequence>
      <group ref="EG_SerShared" minOccurs="1" maxOccurs="1"/>
      <element name="marker" type="CT_Marker" minOccurs="0" maxOccurs="1"/>
      <element name="dPt" type="CT_DPt" minOccurs="0" maxOccurs="unbounded"/>
      <element name="dLbls" type="CT_DLbls" minOccurs="0" maxOccurs="1"/>
      <element name="trendline" type="CT_Trendline" minOccurs="0" maxOccurs="unbounded"/>
      <element name="errBars" type="CT_ErrBars" minOccurs="0" maxOccurs="1"/>
      <element name="cat" type="CT_AxDataSource" minOccurs="0" maxOccurs="1"/>
      <element name="val" type="CT_NumDataSource" minOccurs="0" maxOccurs="1"/>
      <element name="smooth" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.170 ser (Pie Chart Series)

This element specifies a series on a doughnut or pie chart.

Parent Elements
doughnutChart (§5.7.2.50); ofPieChart (§5.7.2.127); pie3DChart (§5.7.2.141); pieChart (§5.7.2.142)

Child Elements	Subclause
cat (Category Axis Data)	§5.7.2.24
dLbls (Data Labels)	§5.7.2.49
dPt (Data Point)	§5.7.2.52
explosion (Explosion)	§5.7.2.61
extLst (Chart Extensibility)	§5.7.2.64
idx (Index)	§5.7.2.84
order (Order)	§5.7.2.129
spPr (Shape Properties)	§5.7.2.198
tx (Series Text)	§5.7.2.216
val (Values)	§5.7.2.225

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PieSer">
   <sequence>
      <group ref="EG_SerShared" minOccurs="1" maxOccurs="1"/>
      <element name="explosion" type="CT_UnsignedInt" minOccurs="0" maxOccurs="1"/>
      <element name="dPt" type="CT_DPt" minOccurs="0" maxOccurs="unbounded"/>
      <element name="dLbls" type="CT_DLbls" minOccurs="0" maxOccurs="1"/>
      <element name="cat" type="CT_AxDataSource" minOccurs="0" maxOccurs="1"/>
      <element name="val" type="CT_NumDataSource" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.171 ser (Surface Chart Series)

This element specifies a series on a surface chart.

Parent Elements
surface3DChart (§5.7.2.204); surfaceChart (§5.7.2.205)

Child Elements	Subclause
cat (Category Axis Data)	§5.7.2.24
extLst (Chart Extensibility)	§5.7.2.64
idx (Index)	§5.7.2.84
order (Order)	§5.7.2.129
spPr (Shape Properties)	§5.7.2.198
tx (Series Text)	§5.7.2.216
val (Values)	§5.7.2.225

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SurfaceSer">
   <sequence>
      <group ref="EG_SerShared" minOccurs="1" maxOccurs="1"/>
      <element name="cat" type="CT_AxDataSource" minOccurs="0" maxOccurs="1"/>
      <element name="val" type="CT_NumDataSource" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.172 ser (Scatter Chart Series)

This element specifies a series on a scatter chart.

Parent Elements
scatterChart (§5.7.2.162)

Child Elements	Subclause
dLbls (Data Labels)	§5.7.2.49
dPt (Data Point)	§5.7.2.52
errBars (Error Bars)	§5.7.2.55
extLst (Chart Extensibility)	§5.7.2.64
idx (Index)	§5.7.2.84
marker (Marker)	§5.7.2.106
order (Order)	§5.7.2.129
smooth (Smoothing)	§5.7.2.195
spPr (Shape Properties)	§5.7.2.198
trendline (Trendlines)	§5.7.2.212
tx (Series Text)	§5.7.2.216
xVal (X Values)	§5.7.2.235
yVal (Y Values)	§5.7.2.238

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ScatterSer">
   <sequence>
      <group ref="EG_SerShared" minOccurs="1" maxOccurs="1"/>
      <element name="marker" type="CT_Marker" minOccurs="0" maxOccurs="1"/>
      <element name="dPt" type="CT_DPt" minOccurs="0" maxOccurs="unbounded"/>
      <element name="dLbls" type="CT_DLbls" minOccurs="0" maxOccurs="1"/>
      <element name="trendline" type="CT_Trendline" minOccurs="0" maxOccurs="unbounded"/>
      <element name="errBars" type="CT_ErrBars" minOccurs="0" maxOccurs="2"/>
      <element name="xVal" type="CT_AxDataSource" minOccurs="0" maxOccurs="1"/>
      <element name="yVal" type="CT_NumDataSource" minOccurs="0" maxOccurs="1"/>
      <element name="smooth" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.173 ser (Radar Chart Series)

This element specifies a series on a radar chart.

Parent Elements
radarChart (§5.7.2.154)

Child Elements	Subclause
cat (Category Axis Data)	§5.7.2.24

Child Elements	Subclause
dLbls (Data Labels)	§5.7.2.49
dPt (Data Point)	§5.7.2.52
extLst (Chart Extensibility)	§5.7.2.64
idx (Index)	§5.7.2.84
marker (Marker)	§5.7.2.106
order (Order)	§5.7.2.129
spPr (Shape Properties)	§5.7.2.198
tx (Series Text)	§5.7.2.216
val (Values)	§5.7.2.225

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RadarSer">
   <sequence>
      <group ref="EG_SerShared" minOccurs="1" maxOccurs="1"/>
      <element name="marker" type="CT_Marker" minOccurs="0" maxOccurs="1"/>
      <element name="dPt" type="CT_DPt" minOccurs="0" maxOccurs="unbounded"/>
      <element name="dLbls" type="CT_DLbls" minOccurs="0" maxOccurs="1"/>
      <element name="cat" type="CT_AxDataSource" minOccurs="0" maxOccurs="1"/>
      <element name="val" type="CT_NumDataSource" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.174 ser (Area Chart Series)

This element specifies a series on an area chart.

Parent Elements
area3DChart (§5.7.2.4); areaChart (§5.7.2.5)

Child Elements	Subclause
cat (Category Axis Data)	§5.7.2.24
dLbls (Data Labels)	§5.7.2.49
dPt (Data Point)	§5.7.2.52
errBars (Error Bars)	§5.7.2.55
extLst (Chart Extensibility)	§5.7.2.64
idx (Index)	§5.7.2.84
order (Order)	§5.7.2.129
pictureOptions (Picture Options)	§5.7.2.139
spPr (Shape Properties)	§5.7.2.198

Child Elements	Subclause
trendline (Trendlines)	§5.7.2.212
tx (Series Text)	§5.7.2.216
val (Values)	§5.7.2.225

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AreaSer">
   <sequence>
      <group ref="EG_SerShared" minOccurs="1" maxOccurs="1"/>
      <element name="pictureOptions" type="CT_PictureOptions" minOccurs="0" maxOccurs="1"/>
      <element name="dPt" type="CT_DPt" minOccurs="0" maxOccurs="unbounded"/>
      <element name="dLbls" type="CT_DLbls" minOccurs="0" maxOccurs="1"/>
      <element name="trendline" type="CT_Trendline" minOccurs="0" maxOccurs="unbounded"/>
      <element name="errBars" type="CT_ErrBars" minOccurs="0" maxOccurs="2"/>
      <element name="cat" type="CT_AxDataSource" minOccurs="0" maxOccurs="1"/>
      <element name="val" type="CT_NumDataSource" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.175 ser (Bar Chart Series)

This element specifies a series on a bar chart.

Parent Elements
bar3DChart (§5.7.2.15); barChart (§5.7.2.16)

Child Elements	Subclause
cat (Category Axis Data)	§5.7.2.24
dLbls (Data Labels)	§5.7.2.49
dPt (Data Point)	§5.7.2.52
errBars (Error Bars)	§5.7.2.55
extLst (Chart Extensibility)	§5.7.2.64
idx (Index)	§5.7.2.84
invertIfNegative (Invert if Negative)	§5.7.2.86
order (Order)	§5.7.2.129
pictureOptions (Picture Options)	§5.7.2.139
shape (Shape)	§5.7.2.178
spPr (Shape Properties)	§5.7.2.198
trendline (Trendlines)	§5.7.2.212
tx (Series Text)	§5.7.2.216

Child Elements	Subclause
val (Values)	§5.7.2.225

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BarSer">
   <sequence>
      <group ref="EG_SerShared" minOccurs="1" maxOccurs="1"/>
      <element name="invertIfNegative" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="pictureOptions" type="CT_PictureOptions" minOccurs="0" maxOccurs="1"/>
      <element name="dPt" type="CT_DPt" minOccurs="0" maxOccurs="unbounded"/>
      <element name="dLbls" type="CT_DLbls" minOccurs="0" maxOccurs="1"/>
      <element name="trendline" type="CT_Trendline" minOccurs="0" maxOccurs="unbounded"/>
      <element name="errBars" type="CT_ErrBars" minOccurs="0" maxOccurs="1"/>
      <element name="cat" type="CT_AxDataSource" minOccurs="0" maxOccurs="1"/>
      <element name="val" type="CT_NumDataSource" minOccurs="0" maxOccurs="1"/>
      <element name="shape" type="CT_Shape" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.176 serAx (Series Axis)

This element specifies a series axis for the chart.

Parent Elements
plotArea (§5.7.2.146)

Child Elements	Subclause
axId (Axis ID)	§5.7.2.9
axPos (Axis Position)	§5.7.2.10
crossAx (Crossing Axis ID)	§5.7.2.31
crosses (Crosses)	§5.7.2.33
crossesAt (Crossing Value)	§5.7.2.34
delete (Delete)	§5.7.2.40
extLst (Chart Extensibility)	§5.7.2.64
majorGridlines (Major Gridlines)	§5.7.2.101
majorTickMark (Major Tick Mark)	§5.7.2.102
minorGridlines (Minor Gridlines)	§5.7.2.110
minorTickMark (Minor Tick Mark)	§5.7.2.111
numFmt (Number Format)	§5.7.2.122
scaling (Scaling)	§5.7.2.161
spPr (Shape Properties)	§5.7.2.198

Child Elements	Subclause
tickLblPos (Tick Label Position)	§5.7.2.208
tickLblSkip (Tick Label Skip)	§5.7.2.209
tickMarkSkip (Tick Mark Skip)	§5.7.2.210
title (Title)	§5.7.2.211
txPr (Text Properties)	§5.7.2.217

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SerAx">
   <sequence>
      <group ref="EG_AxShared" minOccurs="1" maxOccurs="1"/>
      <element name="tickLblSkip" type="CT_Skip" minOccurs="0" maxOccurs="1"/>
      <element name="tickMarkSkip" type="CT_Skip" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.177 serLines (Series Lines)

This element specifies series lines for the chart.

Parent Elements
barChart (§5.7.2.16); ofPieChart (§5.7.2.127)

Child Elements	Subclause
spPr (Shape Properties)	§5.7.2.198

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ChartLines">
   <sequence>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.178 shape (Shape)

This element specifies the shape of a series or a 3-D bar chart.

Parent Elements
bar3DChart (§5.7.2.15); ser (§5.7.2.175)

Attributes	Description
val (Shape Value)	Specifies the shape of the series.

Attributes	Description
	The possible values for this attribute are defined by the ST_Shape simple type (§5.7.3.42).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Shape">
   <attribute name="val" type="ST_Shape" default="box"/>
</complexType>
```

5.7.2.179 showBubbleSize (Show Bubble Size)

This element specifies the bubble size shall be shown in a data label.

Parent Elements
dLbl (§5.7.2.47); dLbls (§5.7.2.49)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.180 showCatName (Show Category Name)

This element specifies that the category name shall be shown in the data label.

Parent Elements
dLbl (§5.7.2.47); dLbls (§5.7.2.49)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for

Attributes	Description
	this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.181 showDLblsOverMax (Show Data Labels over Maximum)

This element specifies data labels over the maximum of the chart shall be shown.

Parent Elements
chart (§5.7.2.27)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, `1`, or `true` specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.182 showHorzBorder (Show Horizontal Border)

This element specifies the horizontal borders shall be shown in a data table.

Parent Elements
dTable (§5.7.2.54)

Attributes	Description

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.183 showKeys (Show Legend Keys)

This element specifies the legend keys shall be shown in a data table.

Parent Elements
dTable (§5.7.2.54)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.184 showLeaderLines (Show Leader Lines)

This element specifies leader lines shall be shown for data labels.

Parent Elements
dLbls (§5.7.2.49)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.185 showLegendKey (Show Legend Key)

This element specifies legend keys shall be shown in data labels.

Parent Elements
dLbl (§5.7.2.47); dLbls (§5.7.2.49)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.186 showNegBubbles (Show Negative Bubbles)

This element specifies negative sized bubbles shall be shown on a bubble chart.

Parent Elements
bubbleChart (§5.7.2.20)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.187 showOutline (Show Outline Border)

This element specifies the outline shall be shown on a data table.

Parent Elements
dTable (§5.7.2.54)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.188 showPercent (Show Percent)

This element specifies that the percentage shall be shown in a data label.

Parent Elements
dLbl (§5.7.2.47); dLbls (§5.7.2.49)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.189 showSerName (Show Series Name)

This element specifies that the series name shall be shown in a data label.

Parent Elements
dLbl (§5.7.2.47); dLbls (§5.7.2.49)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.190 showVal (Show Value)

This element specifies that the value shall be shown in a data label.

Parent Elements
dLbl (§5.7.2.47); dLbls (§5.7.2.49)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.191 showVertBorder (Show Vertical Border)

This element specifies the vertical border shall be shown in a data table.

Parent Elements
dTable (§5.7.2.54)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.192 sideWall (Side Wall)

This element specifies the side wall.

Parent Elements
chart (§5.7.2.27)

Child Elements	Subclause
extLst (Chart Extensibility)	§5.7.2.64
pictureOptions (Picture Options)	§5.7.2.139
spPr (Shape Properties)	§5.7.2.198
thickness (Thickness)	§5.7.2.207

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Surface">
   <sequence>
      <element name="thickness" type="CT_UnsignedInt" minOccurs="0" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
      <element name="pictureOptions" type="CT_PictureOptions" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.193 size (Size)

This element specifies the size of the marker in points.

Parent Elements
marker (§5.7.2.106)

Attributes	Description
val (Marker Size Value)	Specifies the contents of this attribute will contain an integer between 2 and 72. The possible values for this attribute are defined by the ST_MarkerSize simple type (§5.7.3.26).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MarkerSize">
   <attribute name="val" type="ST_MarkerSize" default="5"/>
</complexType>
```

5.7.2.194 sizeRepresents (Size Represents)

This element specifies how the bubble size values are represented on the chart.

Parent Elements
bubbleChart (§5.7.2.20)

Attributes	Description
val (Size Represents	Specifies how the bubble sizes represent the values.

Attributes	Description
Value)	The possible values for this attribute are defined by the ST_SizeRepresents simple type (§5.7.3.43).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SizeRepresents">
   <attribute name="val" type="ST_SizeRepresents" default="area"/>
</complexType>
```

5.7.2.195 smooth (Smoothing)

This element specifies the line connecting the points on the chart shall be smoothed using Catmull-Rom splines.

Parent Elements
lineChart (§5.7.2.98); ser (§5.7.2.172); ser (§5.7.2.169)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.196 splitPos (Split Position)

This element specifies a value that shall be used to determine which data points are in the second pie or bar on a pie of pie or bar of pie chart.

Parent Elements
ofPieChart (§5.7.2.127)

Attributes	Description
val (Floating Point Value)	Specifies that the contents of this attribute will contain a floating point number.

Attributes	Description
	The contents of this number are interpreted based on the context of the parent XML element. The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Double">
   <attribute name="val" type="xsd:double" use="required"/>
</complexType>
```

5.7.2.197 splitType (Split Type)

This element specifies how to determine which data points are in the second pie or bar on a pie of pie or bar of pie chart.

Parent Elements
ofPieChart (§5.7.2.127)

Attributes	Description
val (Split Type Value)	Specifies how to split the data points between the first pie and second pie or bar. The possible values for this attribute are defined by the ST_SplitType simple type (§5.7.3.45).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SplitType">
   <attribute name="val" type="ST_SplitType" default="auto"/>
</complexType>
```

5.7.2.198 spPr (Shape Properties)

This element specifies the formatting for the parent chart element. The custGeom, prstGeom, scene3d, and xfrm elements are not supported. The bwMode attribute is not supported.

Parent Elements
backWall (§5.7.2.11); bandFmt (§5.7.2.13); catAx (§5.7.2.25); chartSpace (§5.7.2.29); dateAx (§5.7.2.39); dispUnitsLbl (§5.7.2.46); dLbl (§5.7.2.47); dLbls (§5.7.2.49); downBars (§5.7.2.51); dPt (§5.7.2.52); dropLines (§5.7.2.53); dTable (§5.7.2.54); errBars (§5.7.2.55); floor (§5.7.2.69); hiLowLines (§5.7.2.80); leaderLines (§5.7.2.92); legend (§5.7.2.94); majorGridlines (§5.7.2.101); marker (§5.7.2.106); minorGridlines (§5.7.2.110); pivotFmt (§5.7.2.143); plotArea (§5.7.2.146); ser (§5.7.2.168); ser (§5.7.2.169); ser (§5.7.2.170); ser (§5.7.2.171); ser (§5.7.2.172); ser (§5.7.2.173); ser (§5.7.2.174); ser (§5.7.2.175); serAx (§5.7.2.176); serLines (§5.7.2.177); sideWall (§5.7.2.192); title (§5.7.2.211); trendline (§5.7.2.212); trendlineLbl (§5.7.2.213); upBars (§5.7.2.218); valAx (§5.7.2.227)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
custGeom (Custom Geometry)	§5.1.11.8
effectDag (Effect Container)	§5.1.10.25
effectLst (Effect Container)	§5.1.10.26
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35
ln (Outline)	§5.1.2.1.24
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
prstGeom (Preset geometry)	§5.1.11.18
scene3d (3D Scene Properties)	§5.1.4.1.26
solidFill (Solid Fill)	§5.1.10.54
sp3d (Apply 3D shape properties)	§5.1.7.12
xfrm (2D Transform for Individual Objects)	§5.1.9.6

Attributes	Description
bwMode (Black and White Mode) Namespace: .../drawingml/2006/main	Specifies that the picture should be rendered using only black and white coloring. That is the coloring information for the picture should be converted to either black or white when rendering the picture. No gray is to be used in rendering this image, only stark black and stark white. [*Note*: This does not mean that the picture itself that is stored within the file is necessarily a black and white picture. This attribute instead sets the rendering mode that the picture will have applied to when rendering. *end note*] The possible values for this attribute are defined by the ST_BlackWhiteMode simple type (§5.1.12.10).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShapeProperties">
   <sequence>
      <element name="xfrm" type="CT_Transform2D" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_Geometry" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_FillProperties" minOccurs="0" maxOccurs="1"/>
      <element name="ln" type="CT_LineProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_EffectProperties" minOccurs="0" maxOccurs="1"/>
      <element name="scene3d" type="CT_Scene3D" minOccurs="0" maxOccurs="1"/>
      <element name="sp3d" type="CT_Shape3D" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="bwMode" type="ST_BlackWhiteMode" use="optional"/>
</complexType>
```

5.7.2.199 stockChart (Stock Charts)

This element contains the collection of stock chart series.

Parent Elements
plotArea (§5.7.2.146)

Child Elements	Subclause
axId (Axis ID)	§5.7.2.9
dLbls (Data Labels)	§5.7.2.49
dropLines (Drop Lines)	§5.7.2.53
extLst (Chart Extensibility)	§5.7.2.64
hiLowLines (High Low Lines)	§5.7.2.80
ser (Line Chart Series)	§5.7.2.169
upDownBars (Up/Down Bars)	§5.7.2.219

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_StockChart">
   <sequence>
      <element name="ser" type="CT_LineSer" minOccurs="3" maxOccurs="4"/>
      <element name="dLbls" type="CT_DLbls" minOccurs="0" maxOccurs="1"/>
      <element name="dropLines" type="CT_ChartLines" minOccurs="0" maxOccurs="1"/>
      <element name="hiLowLines" type="CT_ChartLines" minOccurs="0" maxOccurs="1"/>
      <element name="upDownBars" type="CT_UpDownBars" minOccurs="0" maxOccurs="1"/>
      <element name="axId" type="CT_UnsignedInt" minOccurs="2" maxOccurs="2"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.200 strCache (String Cache)

This element specifies the last string data used for a chart.

Parent Elements
strRef (§5.7.2.202)

Child Elements	Subclause
extLst (Chart Extensibility)	§5.7.2.64
pt (String Point)	§5.7.2.152
ptCount (Point Count)	§5.7.2.153

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_StrData">
   <sequence>
      <element name="ptCount" type="CT_UnsignedInt" minOccurs="0" maxOccurs="1"/>
      <element name="pt" type="CT_StrVal" minOccurs="0" maxOccurs="unbounded"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.201 strLit (String Literal)

This element specifies a set of strings used for a chart

Parent Elements
cat (§5.7.2.24); xVal (§5.7.2.235)

Child Elements	Subclause
extLst (Chart Extensibility)	§5.7.2.64
pt (String Point)	§5.7.2.152
ptCount (Point Count)	§5.7.2.153

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_StrData">
   <sequence>
      <element name="ptCount" type="CT_UnsignedInt" minOccurs="0" maxOccurs="1"/>
      <element name="pt" type="CT_StrVal" minOccurs="0" maxOccurs="unbounded"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.202 strRef (String Reference)

This element specifies a reference to data for a single data label or title with a cache of the last values used.

Parent Elements

Parent Elements
cat (§5.7.2.24); tx (§5.7.2.215); tx (§5.7.2.216); xVal (§5.7.2.235)

Child Elements	Subclause
extLst (Chart Extensibility)	§5.7.2.64
f (Formula)	§5.7.2.65
strCache (String Cache)	§5.7.2.200

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_StrRef">
   <sequence>
      <element name="f" type="xsd:string" minOccurs="1" maxOccurs="1"/>
      <element name="strCache" type="CT_StrData" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.203 style (Style)

This element specifies the style that shall be applied to the chart.

Parent Elements
chartSpace (§5.7.2.29)

Attributes	Description
val (Style Type)	Specifies the chart style. The possible values for this attribute are defined by the ST_Style simple type (§5.7.3.46).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Style">
   <attribute name="val" type="ST_Style" use="required"/>
</complexType>
```

5.7.2.204 surface3DChart (3D Surface Charts)

This element contains the set of 3-D surface series.

Parent Elements
plotArea (§5.7.2.146)

Child Elements	Subclause

Child Elements	Subclause
axId (Axis ID)	§5.7.2.9
bandFmts (Band Formats)	§5.7.2.14
extLst (Chart Extensibility)	§5.7.2.64
ser (Surface Chart Series)	§5.7.2.171
wireframe (Wireframe)	§5.7.2.231

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Surface3DChart">
   <sequence>
      <group ref="EG_SurfaceChartShared" minOccurs="1" maxOccurs="1"/>
      <element name="axId" type="CT_UnsignedInt" minOccurs="3" maxOccurs="3"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.205 surfaceChart (Surface Charts)

This element contains the set of 2-D contour charts.

Parent Elements
plotArea (§5.7.2.146)

Child Elements	Subclause
axId (Axis ID)	§5.7.2.9
bandFmts (Band Formats)	§5.7.2.14
extLst (Chart Extensibility)	§5.7.2.64
ser (Surface Chart Series)	§5.7.2.171
wireframe (Wireframe)	§5.7.2.231

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SurfaceChart">
   <sequence>
      <group ref="EG_SurfaceChartShared" minOccurs="1" maxOccurs="1"/>
      <element name="axId" type="CT_UnsignedInt" minOccurs="2" maxOccurs="3"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.206 symbol (Symbol)

This element specifies the marker that shall be used for the data points.

Parent Elements

Parent Elements
marker (§5.7.2.106)

Attributes	Description
val (Marker Style Value)	Specifies the marker style. The possible values for this attribute are defined by the ST_MarkerStyle simple type (§5.7.3.27).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MarkerStyle">
   <attribute name="val" type="ST_MarkerStyle" use="required"/>
</complexType>
```

5.7.2.207 thickness (Thickness)

This element specifies the thickness of the walls or floor as a percentage of the largest dimension of the plot volume.

Parent Elements
backWall (§5.7.2.11); floor (§5.7.2.69); sideWall (§5.7.2.192)

Attributes	Description
val (Integer Value)	Specifies that the contents of this attribute will contain an integer number. The contents of this number are interpreted based on the context of the parent XML element. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_UnsignedInt">
   <attribute name="val" type="xsd:unsignedInt" use="required"/>
</complexType>
```

5.7.2.208 tickLblPos (Tick Label Position)

This element specifies the position of the tick labels on the axis.

Parent Elements
catAx (§5.7.2.25); dateAx (§5.7.2.39); serAx (§5.7.2.176); valAx (§5.7.2.227)

Attributes	Description
val (Tick Label Position Value)	Specifies the tick label position. The possible values for this attribute are defined by the ST_TickLblPos simple type (§5.7.3.48).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TickLblPos">
   <attribute name="val" type="ST_TickLblPos" default="nextTo"/>
</complexType>
```

5.7.2.209 tickLblSkip (Tick Label Skip)

This element specifies how many tick labels to skip between label that is drawn.

Parent Elements
catAx (§5.7.2.25); serAx (§5.7.2.176)

Attributes	Description
val (Tick Skip Value)	Specifies the contents of this attribute will contain an integer greater than or equal to one. The possible values for this attribute are defined by the ST_Skip simple type (§5.7.3.44).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Skip">
   <attribute name="val" type="ST_Skip" use="required"/>
</complexType>
```

5.7.2.210 tickMarkSkip (Tick Mark Skip)

This element specifies how many tick marks shall be skipped before the next one shall be drawn.

Parent Elements
catAx (§5.7.2.25); serAx (§5.7.2.176)

Attributes	Description
val (Tick Skip Value)	Specifies the contents of this attribute will contain an integer greater than or equal to one. The possible values for this attribute are defined by the ST_Skip simple type (§5.7.3.44).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Skip">
   <attribute name="val" type="ST_Skip" use="required"/>
</complexType>
```

5.7.2.211 title (Title)

This element specifies a title.

Parent Elements
catAx (§5.7.2.25); chart (§5.7.2.27); dateAx (§5.7.2.39); serAx (§5.7.2.176); valAx (§5.7.2.227)

Child Elements	Subclause
extLst (Chart Extensibility)	§5.7.2.64
layout (Layout)	§5.7.2.88
overlay (Overlay)	§5.7.2.133
spPr (Shape Properties)	§5.7.2.198
tx (Chart Text)	§5.7.2.215
txPr (Text Properties)	§5.7.2.217

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Title">
   <sequence>
      <element name="tx" type="CT_Tx" minOccurs="0" maxOccurs="1"/>
      <element name="layout" type="CT_Layout" minOccurs="0" maxOccurs="1"/>
      <element name="overlay" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
      <element name="txPr" type="a:CT_TextBody" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.212 trendline (Trendlines)

This element specifies a trendline.

Parent Elements
ser (§5.7.2.175); ser (§5.7.2.172); ser (§5.7.2.169); ser (§5.7.2.168); ser (§5.7.2.174)

Child Elements	Subclause
backward (Backward)	§5.7.2.12
dispEq (Display Equation)	§5.7.2.43
dispRSqr (Display R Squared Value)	§5.7.2.44

Child Elements	Subclause
extLst (Chart Extensibility)	§5.7.2.64
forward (Forward)	§5.7.2.73
intercept (Intercept)	§5.7.2.85
name (Trendline Name)	§5.7.2.117
order (Polynomial Trendline Order)	§5.7.2.130
period (Period)	§5.7.2.136
spPr (Shape Properties)	§5.7.2.198
trendlineLbl (Trendline Label)	§5.7.2.213
trendlineType (Trendline Type)	§5.7.2.214

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Trendline">
   <sequence>
      <element name="name" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
      <element name="trendlineType" type="CT_TrendlineType" minOccurs="1" maxOccurs="1"/>
      <element name="order" type="CT_Order" minOccurs="0" maxOccurs="1"/>
      <element name="period" type="CT_Period" minOccurs="0" maxOccurs="1"/>
      <element name="forward" type="CT_Double" minOccurs="0" maxOccurs="1"/>
      <element name="backward" type="CT_Double" minOccurs="0" maxOccurs="1"/>
      <element name="intercept" type="CT_Double" minOccurs="0" maxOccurs="1"/>
      <element name="dispRSqr" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="dispEq" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="trendlineLbl" type="CT_TrendlineLbl" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.213 trendlineLbl (Trendline Label)

This element specifies the label for the trendline.

Parent Elements
trendline (§5.7.2.212)

Child Elements	Subclause
extLst (Chart Extensibility)	§5.7.2.64
layout (Layout)	§5.7.2.88
numFmt (Number Format)	§5.7.2.122
spPr (Shape Properties)	§5.7.2.198
tx (Chart Text)	§5.7.2.215

Child Elements	Subclause
txPr (Text Properties)	§5.7.2.217

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TrendlineLbl">
   <sequence>
      <element name="layout" type="CT_Layout" minOccurs="0" maxOccurs="1"/>
      <element name="tx" type="CT_Tx" minOccurs="0" maxOccurs="1"/>
      <element name="numFmt" type="CT_NumFmt" minOccurs="0" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
      <element name="txPr" type="a:CT_TextBody" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.214 trendlineType (Trendline Type)

This element specifies the type of the trendline.

Parent Elements
trendline (§5.7.2.212)

Attributes	Description
val (Trendline Type Value)	Specifies the trendline type. The possible values for this attribute are defined by the ST_TrendlineType simple type (§5.7.3.51).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TrendlineType">
   <attribute name="val" type="ST_TrendlineType" default="linear"/>
</complexType>
```

5.7.2.215 tx (Chart Text)

This element specifies text to use on a chart, including rich text formatting.

Parent Elements
dispUnitsLbl (§5.7.2.46); dLbl (§5.7.2.47); title (§5.7.2.211); trendlineLbl (§5.7.2.213)

Child Elements	Subclause
rich (Rich Text)	§5.7.2.157
strRef (String Reference)	§5.7.2.202

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Tx">
   <sequence>
      <choice minOccurs="1" maxOccurs="1">
         <element name="strRef" type="CT_StrRef" minOccurs="1" maxOccurs="1"/>
         <element name="rich" type="a:CT_TextBody" minOccurs="1" maxOccurs="1"/>
      </choice>
   </sequence>
</complexType>
```

5.7.2.216 tx (Series Text)

This element specifies text for a series name, without rich text formatting.

Parent Elements
ser (§5.7.2.168); ser (§5.7.2.169); ser (§5.7.2.170); ser (§5.7.2.171); ser (§5.7.2.172); ser (§5.7.2.173); ser (§5.7.2.174); ser (§5.7.2.175)

Child Elements	Subclause
strRef (String Reference)	§5.7.2.202
v (Text Value)	§5.7.2.223

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SerTx">
   <sequence>
      <choice minOccurs="1" maxOccurs="1">
         <element name="strRef" type="CT_StrRef" minOccurs="1" maxOccurs="1"/>
         <element name="v" type="ST_Xstring" minOccurs="1" maxOccurs="1"/>
      </choice>
   </sequence>
</complexType>
```

5.7.2.217 txPr (Text Properties)

This element specifies text formatting. The lstStyle element is not supported.

Parent Elements
catAx (§5.7.2.25); chartSpace (§5.7.2.29); dateAx (§5.7.2.39); dispUnitsLbl (§5.7.2.46); dLbl (§5.7.2.47); dLbls (§5.7.2.49); dTable (§5.7.2.54); legend (§5.7.2.94); legendEntry (§5.7.2.95); pivotFmt (§5.7.2.143); serAx (§5.7.2.176); title (§5.7.2.211); trendlineLbl (§5.7.2.213); valAx (§5.7.2.227)

Child Elements	Subclause
bodyPr (Body Properties)	§5.1.5.1.1
lstStyle (Text List Styles)	§5.1.5.4.12
p (Text Paragraphs)	§5.1.5.2.6

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextBody">
   <sequence>
      <element name="bodyPr" type="CT_TextBodyProperties" minOccurs="1" maxOccurs="1"/>
      <element name="lstStyle" type="CT_TextListStyle" minOccurs="0" maxOccurs="1"/>
      <element name="p" type="CT_TextParagraph" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.7.2.218 upBars (Up Bars)

This element specifies the up bars on the chart.

Parent Elements
upDownBars (§5.7.2.219)

Child Elements	Subclause
spPr (Shape Properties)	§5.7.2.198

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_UpDownBar">
   <sequence>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.219 upDownBars (Up/Down Bars)

This element specifies the up and down bars.

Parent Elements
lineChart (§5.7.2.98); stockChart (§5.7.2.199)

Child Elements	Subclause
downBars (Down Bars)	§5.7.2.51
extLst (Chart Extensibility)	§5.7.2.64
gapWidth (Gap Width)	§5.7.2.75
upBars (Up Bars)	§5.7.2.218

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_UpDownBars">
   <sequence>
      <element name="gapWidth" type="CT_GapAmount" minOccurs="0" maxOccurs="1"/>
      <element name="upBars" type="CT_UpDownBar" minOccurs="0" maxOccurs="1"/>
      <element name="downBars" type="CT_UpDownBar" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.220 userInterface (User Interface)

This element specifies that the protection applies to the user interface only, and not to changes made through the object model.

Parent Elements
protection (§5.7.2.150)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of off, 0, or false specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.221 userShapes (User Shapes)

This element shall specify the shapes drawn on top of the chart.

Child Elements	Subclause
absSizeAnchor (Absolute Anchor Shape Size)	§5.8.2.1
relSizeAnchor (Relative Anchor Shape Size)	§5.8.2.21

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Drawing">
   <sequence>
      <group ref="EG_Anchor" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.7.2.222 userShapes (Reference to Chart Drawing Part)

This element specifies a relationship to a separate part which contains a drawing to be drawn on top of the chart.

Parent Elements
chartSpace (§5.7.2.29)

Attributes	Description
id (Relationship Reference) Namespace: .../officeDocument/2006/relationships	Specifies the relationship ID for the relationship for this Chart, Chart Drawing, or VML Drawing part. The type of relationship needed is specified by the parent element. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RelId">
   <attribute ref="r:id" use="required"/>
</complexType>
```

5.7.2.223 v (Text Value)

This element specifies a text value for a category axis label or a series name.

The possible values for this element are defined by the ST_Xstring simple type (§5.7.3.52).

Parent Elements
pt (§5.7.2.152); tx (§5.7.2.216)

5.7.2.224 v (Numeric Value)

This element specifies a numeric value.

The possible values for this element are defined by the ST_Xstring simple type (§5.7.3.52).

Parent Elements
pt (§5.7.2.151)

5.7.2.225 val (Values)

This element specifies the data values which shall be used to define the location of data markers on a chart.

Parent Elements
ser (§5.7.2.174); ser (§5.7.2.171); ser (§5.7.2.175); ser (§5.7.2.169); ser (§5.7.2.170); ser (§5.7.2.173)

Child Elements	Subclause
numLit (Number Literal)	§5.7.2.123
numRef (Number Reference)	§5.7.2.124

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NumDataSource">
   <sequence>
      <choice minOccurs="1" maxOccurs="1">
         <element name="numRef" type="CT_NumRef" minOccurs="1" maxOccurs="1"/>
         <element name="numLit" type="CT_NumData" minOccurs="1" maxOccurs="1"/>
      </choice>
   </sequence>
</complexType>
```

5.7.2.226 val (Error Bar Value)

This element specifies a value which is used with the Error Bar Type to determine the length of the error bars.

Parent Elements
errBars (§5.7.2.55)

Attributes	Description
val (Floating Point Value)	Specifies that the contents of this attribute will contain a floating point number. The contents of this number are interpreted based on the context of the parent XML element. The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Double">
   <attribute name="val" type="xsd:double" use="required"/>
</complexType>
```

5.7.2.227 valAx (Value Axis)

This element specifies a value axis.

Parent Elements
plotArea (§5.7.2.146)

Child Elements	Subclause
axId (Axis ID)	§5.7.2.9
axPos (Axis Position)	§5.7.2.10
crossAx (Crossing Axis ID)	§5.7.2.31
crossBetween (Cross Between)	§5.7.2.32
crosses (Crosses)	§5.7.2.33
crossesAt (Crossing Value)	§5.7.2.34
delete (Delete)	§5.7.2.40
dispUnits (Display Units)	§5.7.2.45
extLst (Chart Extensibility)	§5.7.2.64
majorGridlines (Major Gridlines)	§5.7.2.101
majorTickMark (Major Tick Mark)	§5.7.2.102
majorUnit (Major Unit)	§5.7.2.104
minorGridlines (Minor Gridlines)	§5.7.2.110
minorTickMark (Minor Tick Mark)	§5.7.2.111
minorUnit (Minor Unit)	§5.7.2.113
numFmt (Number Format)	§5.7.2.122
scaling (Scaling)	§5.7.2.161
spPr (Shape Properties)	§5.7.2.198
tickLblPos (Tick Label Position)	§5.7.2.208
title (Title)	§5.7.2.211
txPr (Text Properties)	§5.7.2.217

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ValAx">
   <sequence>
      <group ref="EG_AxShared" minOccurs="1" maxOccurs="1"/>
      <element name="crossBetween" type="CT_CrossBetween" minOccurs="0" maxOccurs="1"/>
      <element name="majorUnit" type="CT_AxisUnit" minOccurs="0" maxOccurs="1"/>
      <element name="minorUnit" type="CT_AxisUnit" minOccurs="0" maxOccurs="1"/>
      <element name="dispUnits" type="CT_DispUnits" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.228 varyColors (Vary Colors by Point)

This element specifies that each data marker in the series shall have a different color.

Parent Elements
area3DChart (§5.7.2.4); areaChart (§5.7.2.5); bar3DChart (§5.7.2.15); barChart (§5.7.2.16); bubbleChart (§5.7.2.20); doughnutChart (§5.7.2.50); line3DChart (§5.7.2.97); lineChart (§5.7.2.98); ofPieChart (§5.7.2.127); pie3DChart (§5.7.2.141); pieChart (§5.7.2.142); radarChart (§5.7.2.154); scatterChart (§5.7.2.162)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or `true` specifies that the property is applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.229 view3D (View In 3D)

This element specifies the 3-D view of the chart.

Parent Elements
chart (§5.7.2.27)

Child Elements	Subclause
depthPercent (Depth Percent)	§5.7.2.41
extLst (Chart Extensibility)	§5.7.2.64
hPercent (Height Percent)	§5.7.2.83
perspective (Perspective)	§5.7.2.137
rAngAx (Right Angle Axes)	§5.7.2.156
rotX (X Rotation)	§5.7.2.158
rotY (Y Rotation)	§5.7.2.159

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_View3D">
   <sequence>
      <element name="rotX" type="CT_RotX" minOccurs="0" maxOccurs="1"/>
      <element name="hPercent" type="CT_HPercent" minOccurs="0" maxOccurs="1"/>
      <element name="rotY" type="CT_RotY" minOccurs="0" maxOccurs="1"/>
      <element name="depthPercent" type="CT_DepthPercent" minOccurs="0" maxOccurs="1"/>
      <element name="rAngAx" type="CT_Boolean" minOccurs="0" maxOccurs="1"/>
      <element name="perspective" type="CT_Perspective" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_ExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.7.2.230 w (Width)

This element specifies the width (if Width Mode is Factor) or right (if Width Mode is Edge) of the chart element as a fraction of the width of the chart.

Parent Elements
manualLayout (§5.7.2.105)

Attributes	Description
val (Floating Point Value)	Specifies that the contents of this attribute will contain a floating point number. The contents of this number are interpreted based on the context of the parent XML element. The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Double">
   <attribute name="val" type="xsd:double" use="required"/>
</complexType>
```

5.7.2.231 wireframe (Wireframe)

This element specifies the surface chart is drawn as a wireframe.

Parent Elements
surface3DChart (§5.7.2.204); surfaceChart (§5.7.2.205)

Attributes	Description
val (Boolean Value)	Specifies a boolean value for the property defined by the parent XML element. A value of on, 1, or true specifies that the property is applied. This is the default value for

Attributes	Description
	this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `off`, `0`, or `false` specifies that the property is not applied. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Boolean">
   <attribute name="val" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.7.2.232 wMode (Width Mode)

This element specifies how to interpret the Width element for this manual layout.

Parent Elements
manualLayout (§5.7.2.105)

Attributes	Description
val (Layout Mode Value)	Specifies the layout mode for the width. The possible values for this attribute are defined by the ST_LayoutMode simple type (§5.7.3.20).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LayoutMode">
   <attribute name="val" type="ST_LayoutMode" default="factor"/>
</complexType>
```

5.7.2.233 x (Left)

This element specifies the x location (left) of the chart element as a fraction of the width of the chart. If Left Mode is Factor, then the position is relative to the default position for the chart element.

Parent Elements
manualLayout (§5.7.2.105)

Attributes	Description
val (Floating Point Value)	Specifies that the contents of this attribute will contain a floating point number. The contents of this number are interpreted based on the context of the parent XML element.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Double">
   <attribute name="val" type="xsd:double" use="required"/>
</complexType>
```

5.7.2.234 xMode (Left Mode)

This element specifies how to interpret the Left element for this manual layout.

Parent Elements
manualLayout (§5.7.2.105)

Attributes	Description
val (Layout Mode Value)	Specifies the layout mode for the width. The possible values for this attribute are defined by the ST_LayoutMode simple type (§5.7.3.20).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LayoutMode">
   <attribute name="val" type="ST_LayoutMode" default="factor"/>
</complexType>
```

5.7.2.235 xVal (X Values)

This element specifies the x values which shall be used to define the location of data markers on a chart.

Parent Elements
ser (§5.7.2.168); ser (§5.7.2.172)

Child Elements	Subclause
multiLvlStrRef (Multi Level String Reference)	§5.7.2.116
numLit (Number Literal)	§5.7.2.123
numRef (Number Reference)	§5.7.2.124
strLit (String Literal)	§5.7.2.201
strRef (String Reference)	§5.7.2.202

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AxDataSource">
   <sequence>
      <choice minOccurs="1" maxOccurs="1">
         <element name="multiLvlStrRef" type="CT_MultiLvlStrRef" minOccurs="1" maxOccurs="1"/>
         <element name="numRef" type="CT_NumRef" minOccurs="1" maxOccurs="1"/>
         <element name="numLit" type="CT_NumData" minOccurs="1" maxOccurs="1"/>
         <element name="strRef" type="CT_StrRef" minOccurs="1" maxOccurs="1"/>
         <element name="strLit" type="CT_StrData" minOccurs="1" maxOccurs="1"/>
      </choice>
   </sequence>
</complexType>
```

5.7.2.236 y (Top)

This element specifies the top of the chart element as a fraction of the height of the chart. If Top Mode is Factor, then the position is relative to the default position for the chart element.

Parent Elements
manualLayout (§5.7.2.105)

Attributes	Description
val (Floating Point Value)	Specifies that the contents of this attribute will contain a floating point number. The contents of this number are interpreted based on the context of the parent XML element. The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Double">
   <attribute name="val" type="xsd:double" use="required"/>
</complexType>
```

5.7.2.237 yMode (Top Mode)

This element specifies how to interpret the Top element for this manual layout.

Parent Elements
manualLayout (§5.7.2.105)

Attributes	Description
val (Layout Mode Value)	Specifies the layout mode for the width. The possible values for this attribute are defined by the ST_LayoutMode simple type

Attributes	Description
	(§5.7.3.20).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LayoutMode">
   <attribute name="val" type="ST_LayoutMode" default="factor"/>
</complexType>
```

5.7.2.238 yVal (Y Values)

This element specifies the y values which shall be used to define the location of data markers on a chart.

Parent Elements
ser (§5.7.2.168); ser (§5.7.2.172)

Child Elements	Subclause
numLit (Number Literal)	§5.7.2.123
numRef (Number Reference)	§5.7.2.124

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NumDataSource">
   <sequence>
      <choice minOccurs="1" maxOccurs="1">
         <element name="numRef" type="CT_NumRef" minOccurs="1" maxOccurs="1"/>
         <element name="numLit" type="CT_NumData" minOccurs="1" maxOccurs="1"/>
      </choice>
   </sequence>
</complexType>
```

5.7.3 Simple Types

This is the complete list of simple types in the http://schemas.openxmlformats.org/drawingml/2006/chart namespace.

5.7.3.1 ST_AxisUnit (Axis Unit)

This simple type specifies that its contents will contain a positive floating point number.

This simple type's contents are a restriction of the XML Schema double datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than 0.

Referenced By
majorUnit@val (§5.7.2.104); minorUnit@val (§5.7.2.113)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_AxisUnit">
   <restriction base="xsd:double">
      <minExclusive value="0"/>
   </restriction>
</simpleType>
```

5.7.3.2 ST_AxPos (Axis Position)

This simple type specifies the possible positions for an axis.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
b (Bottom)	Specifies that the axis shall be displayed at the bottom of the plot area.
l (Left)	Specifies that the axis shall be displayed at the left of the plot area.
r (Right)	Specifies that the axis shall be displayed at the right of the plot area.
t (Top)	Specifies that the axis shall be displayed at the top of the plot area.

Referenced By
axPos@val (§5.7.2.10)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_AxPos">
   <restriction base="xsd:string">
      <enumeration value="b"/>
      <enumeration value="l"/>
      <enumeration value="r"/>
      <enumeration value="t"/>
   </restriction>
</simpleType>
```

5.7.3.3 ST_BarDir (Bar Direction)

This simple type specifies the possible directions for a bar chart.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description

Enumeration Value	Description
bar (Bar)	Specifies that the chart is a bar chart - the data markers are horizontal rectangles.
col (Column)	Specifies that the chart is a column chart - the data markers are vertical rectangles.

Referenced By
barDir@val (§5.7.2.17)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_BarDir">
   <restriction base="xsd:string">
      <enumeration value="bar"/>
      <enumeration value="col"/>
   </restriction>
</simpleType>
```

5.7.3.4 ST_BarGrouping (Bar Grouping)

This simple type specifies the possible groupings for a bar chart.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
clustered (Clustered)	Specifies that the chart series are drawn next to each other along the category axis.
percentStacked (100% Stacked)	Specifies that the chart series are drawn next to each other along the value axis and scaled to total 100%.
stacked (Stacked)	Specifies that the chart series are drawn next to each other on the value axis.
standard (Standard)	Specifies that the chart series are drawn next to each other on the depth axis.

Referenced By
grouping@val (§5.7.2.77)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_BarGrouping">
   <restriction base="xsd:string">
      <enumeration value="percentStacked"/>
      <enumeration value="clustered"/>
      <enumeration value="standard"/>
      <enumeration value="stacked"/>
   </restriction>
</simpleType>
```

5.7.3.5 ST_BubbleScale (Bubble Scale)

This simple type specifies that its contents will contain an integer between 0 and 300.

This simple type's contents are a restriction of the XML Schema unsignedInt datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 0.
- This simple type has a maximum value of less than or equal to 300.

Referenced By
bubbleScale@val (§5.7.2.21)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_BubbleScale">
   <restriction base="xsd:unsignedInt">
      <minInclusive value="0"/>
      <maxInclusive value="300"/>
   </restriction>
</simpleType>
```

5.7.3.6 ST_BuiltInUnit (Built-In Unit)

This simple type specifies the built in display units for an axis.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
billions (Billions)	Specifies the values on the chart shall be divided by 1,000,000,000.
hundredMillions (Hundred Millions)	Specifies the values on the chart shall be divided by 100,000,000.
hundreds (Hundreds)	Specifies the values on the chart shall be divided by 100.
hundredThousands (Hundred Thousands)	Specifies the values on the chart shall be divided by

Enumeration Value	Description
	100,000.
millions (Millions)	Specifies the values on the chart shall be divided by 1,000,000.
tenMillions (Ten Millions)	Specifies the values on the chart shall be divided by 10,000,000.
tenThousands (Ten Thousands)	Specifies the values on the chart shall be divided by 10,000.
thousands (Thousands)	Specifies the values on the chart shall be divided by 1,000.
trillions (Trillions)	Specifies the values on the chart shall be divided by 1,000,000,000.

Referenced By
builtInUnit@val (§5.7.2.23)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_BuiltInUnit">
   <restriction base="xsd:string">
      <enumeration value="hundreds"/>
      <enumeration value="thousands"/>
      <enumeration value="tenThousands"/>
      <enumeration value="hundredThousands"/>
      <enumeration value="millions"/>
      <enumeration value="tenMillions"/>
      <enumeration value="hundredMillions"/>
      <enumeration value="billions"/>
      <enumeration value="trillions"/>
   </restriction>
</simpleType>
```

5.7.3.7 ST_CrossBetween (Cross Between)

This simple type specifies the possible crossing states of an axis.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
between (Between)	Specifies the value axis shall cross the category axis between data markers.
midCat (Midpoint of Category)	Specifies the value axis shall cross the category axis at the midpoint of a category.

Referenced By
crossBetween@val (§5.7.2.32)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_CrossBetween">
   <restriction base="xsd:string">
      <enumeration value="between"/>
      <enumeration value="midCat"/>
   </restriction>
</simpleType>
```

5.7.3.8 ST_Crosses (Crosses)

This simple type specifies the possible crossing points for an axis.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
autoZero (Axis Crosses at Zero)	The category axis crosses at the zero point of the value axis (if possible), or the minimum value (if the minimum is greater than zero) or the maximum (if the maximum is less than zero).
max (Maximum)	The axis crosses at the maximum value
min (Minimum)	Axis crosses at the minimum value of the chart.

Referenced By
crosses@val (§5.7.2.33)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Crosses">
   <restriction base="xsd:string">
      <enumeration value="autoZero"/>
      <enumeration value="max"/>
      <enumeration value="min"/>
   </restriction>
</simpleType>
```

5.7.3.9 ST_DepthPercent (Depth Percent)

This simple type specifies that its contents will contain a whole number between 20 and 2000, whose contents are a percentage.

This simple type's contents are a restriction of the XML Schema unsignedShort datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 20.
- This simple type has a maximum value of less than or equal to 2000.

Referenced By
depthPercent@val (§5.7.2.41)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DepthPercent">
   <restriction base="xsd:unsignedShort">
      <minInclusive value="20"/>
      <maxInclusive value="2000"/>
   </restriction>
</simpleType>
```

5.7.3.10 ST_DispBlanksAs (Display Blanks As)

This simple type specifies the possible ways to display blanks.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
gap (Gap)	Specifies that blank values shall be left as a gap.
span (Span)	Specifies that blank values shall be spanned with a line.
zero (Zero)	Specifies that blank values shall be treated as zero.

Referenced By
dispBlanksAs@val (§5.7.2.42)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DispBlanksAs">
   <restriction base="xsd:string">
      <enumeration value="span"/>
      <enumeration value="gap"/>
      <enumeration value="zero"/>
   </restriction>
</simpleType>
```

5.7.3.11 ST_DLblPos (Data Label Position)

This simple type specifies the possible positions for a data label.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
b (Bottom)	Specifies that data labels shall be displayed below the data marker.
bestFit (Best Fit)	Specifies that data labels shall be displayed in the best position.
ctr (Center)	Specifies that data labels shall be displayed centered on the data marker.
inBase (Inside Base)	Specifies that data labels shall be displayed inside the base of the data marker.
inEnd (Inside End)	Specifies that data labels shall be displayed inside the end of the data marker.
l (Left)	Specifies that data labels shall be displayed to the left of the data marker.
outEnd (Outside End)	Specifies that data labels shall be displayed outside the end of the data marker.
r (Right)	Specifies that data labels shall be displayed to the right of the data marker.
t (Top)	Specifies that data labels shall be displayed above the data marker.

Referenced By
dLblPos@val (§5.7.2.48)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_DLblPos">
   <restriction base="xsd:string">
      <enumeration value="bestFit"/>
      <enumeration value="b"/>
      <enumeration value="ctr"/>
      <enumeration value="inBase"/>
      <enumeration value="inEnd"/>
      <enumeration value="l"/>
      <enumeration value="outEnd"/>
      <enumeration value="r"/>
      <enumeration value="t"/>
   </restriction>
</simpleType>
```

5.7.3.12 ST_ErrBarType (Error Bar Type)

This simple type specifies the possible ways to draw an error bar.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
both (Both)	Specifies that error bars shall be shown in the positive and negative directions.
minus (Minus)	Specifies that error bars shall be shown in the negative direction only.
plus (Plus)	Specifies that error bars shall be shown in the positive direction only.

Referenced By
errBarType@val (§5.7.2.56)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ErrBarType">
   <restriction base="xsd:string">
      <enumeration value="both"/>
      <enumeration value="minus"/>
      <enumeration value="plus"/>
   </restriction>
</simpleType>
```

5.7.3.13 ST_ErrDir (Error Bar Direction)

This simple type specifies the possible directions for error bars.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
x (X)	Specifies that error bars shall be shown in the x direction.
y (Y)	Specifies that error bars shall be shown in the y direction.

Referenced By
errDir@val (§5.7.2.57)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ErrDir">
   <restriction base="xsd:string">
      <enumeration value="x"/>
      <enumeration value="y"/>
   </restriction>
</simpleType>
```

5.7.3.14 ST_ErrValType (Error Value Type)

This simple type specifies the possible ways to determine the length of the error bars

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
cust (Custom Error Bars)	Specifies that the length of the error bars shall be determined by the Plus and Minus elements.
fixedVal (Fixed Value)	Specifies that the length of the error bars shall be the fixed value determined by Error Bar Value.
percentage (Percentage)	Specifies that the length of the error bars shall be Error Bar Value percent of the data.
stdDev (Standard Deviation)	Specifies that the length of the error bars shall be Error Bar Value standard deviations of the data.
stdErr (Standard Error)	Specifies that the length of the error bars shall be Error Bar Value standard errors of the data.

Referenced By
errValType@val (§5.7.2.58)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ErrValType">
   <restriction base="xsd:string">
      <enumeration value="cust"/>
      <enumeration value="fixedVal"/>
      <enumeration value="percentage"/>
      <enumeration value="stdDev"/>
      <enumeration value="stdErr"/>
   </restriction>
</simpleType>
```

5.7.3.15 ST_FirstSliceAng (First Slice Angle)

This simple type specifies that its contents will contain an integer between 0 and 360.

This simple type's contents are a restriction of the XML Schema unsignedShort datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 0.
- This simple type has a maximum value of less than or equal to 360.

Referenced By
firstSliceAng@val (§5.7.2.68)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FirstSliceAng">
   <restriction base="xsd:unsignedShort">
      <minInclusive value="0"/>
      <maxInclusive value="360"/>
   </restriction>
</simpleType>
```

5.7.3.16 ST_GapAmount (Gap Amount)

This simple type specifies that its contents will contain an integer between 0 and 500, whose contents are a percentage.

This simple type's contents are a restriction of the XML Schema unsignedShort datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 0.
- This simple type has a maximum value of less than or equal to 500.

Referenced By
gapDepth@val (§5.7.2.74); gapWidth@val (§5.7.2.75)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_GapAmount">
   <restriction base="xsd:unsignedShort">
      <minInclusive value="0"/>
      <maxInclusive value="500"/>
   </restriction>
</simpleType>
```

5.7.3.17 ST_Grouping (Grouping)

This simple type specifies the possible groupings for a bar chart.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description

Enumeration Value	Description
percentStacked (100% Stacked)	Specifies that the chart series are drawn next to each other along the value axis and scaled to total 100%.
stacked (Stacked)	Specifies that the chart series are drawn next to each other on the value axis.
standard (Standard)	Specifies that the chart series are drawn on the value axis.

Referenced By
grouping@val (§5.7.2.76)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Grouping">
   <restriction base="xsd:string">
      <enumeration value="percentStacked"/>
      <enumeration value="standard"/>
      <enumeration value="stacked"/>
   </restriction>
</simpleType>
```

5.7.3.18 ST_HoleSize (Hole Size)

This simple type specifies that its contents will contain an integer between 10 and 90, whose contents are a percentage.

This simple type's contents are a restriction of the XML Schema unsignedByte datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 10.
- This simple type has a maximum value of less than or equal to 90.

Referenced By
holeSize@val (§5.7.2.82)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_HoleSize">
   <restriction base="xsd:unsignedByte">
      <minInclusive value="10"/>
      <maxInclusive value="90"/>
   </restriction>
</simpleType>
```

5.7.3.19 ST_HPercent (Height Percent)

This simple type specifies that its contents will contain an integer between 5 and 500, whose contents are a percentage.

This simple type's contents are a restriction of the XML Schema unsignedShort datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 5.
- This simple type has a maximum value of less than or equal to 500.

Referenced By
hPercent@val (§5.7.2.83)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_HPercent">
   <restriction base="xsd:unsignedShort">
      <minInclusive value="5"/>
      <maxInclusive value="500"/>
   </restriction>
</simpleType>
```

5.7.3.20 ST_LayoutMode (Layout Mode)

This simple type specifies the possible ways to store a chart element's position.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
edge (Edge)	Specifies that the Width or Height shall be interpreted as the Right or Bottom of the chart element.
factor (Factor)	Specifies that the Width or Height shall be interpreted as the Width or Height of the chart element.

Referenced By
hMode@val (§5.7.2.81); wMode@val (§5.7.2.232); xMode@val (§5.7.2.234); yMode@val (§5.7.2.237)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_LayoutMode">
   <restriction base="xsd:string">
      <enumeration value="edge"/>
      <enumeration value="factor"/>
   </restriction>
</simpleType>
```

5.7.3.21 ST_LayoutTarget (Layout Target)

This simple type specifies the possible ways to layout the plot area.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
inner (Inner)	Specifies that the plot area size shall determine the size of the plot area, not including the tick marks and axis labels.
outer (Outer)	Specifies that the plot area size shall determine the size of the plot area, the tick marks, and the axis labels.

Referenced By
layoutTarget@val (§5.7.2.89)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_LayoutTarget">
   <restriction base="xsd:string">
      <enumeration value="inner"/>
      <enumeration value="outer"/>
   </restriction>
</simpleType>
```

5.7.3.22 ST_LblAlgn (Label Alignment)

This simple type specifies the possible ways to align the tick labels.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
ctr (Center)	Specifies that the text shall be centered.
l (Left)	Specifies that the text shall be left justified.
r (Right)	Specifies that the text shall be right justified.

Referenced By
lblAlgn@val (§5.7.2.90)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_LblAlgn">
   <restriction base="xsd:string">
      <enumeration value="ctr"/>
      <enumeration value="l"/>
      <enumeration value="r"/>
   </restriction>
</simpleType>
```

5.7.3.23 ST_LblOffset (Label Offset)

This simple type specifies that its contents will contain an integer between 0 and 1000, whose contents are a percentage of the default value.

This simple type's contents are a restriction of the XML Schema unsignedShort datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 0.
- This simple type has a maximum value of less than or equal to 1000.

Referenced By
lblOffset@val (§5.7.2.91)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_LblOffset">
   <restriction base="xsd:unsignedShort">
      <minInclusive value="0"/>
      <maxInclusive value="1000"/>
   </restriction>
</simpleType>
```

5.7.3.24 ST_LegendPos (Legend Position)

This simple type specifies the possible positions for a legend.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
b (Bottom)	Specifies that the legend shall be drawn at the bottom of the chart.
l (Left)	Specifies that the legend shall be drawn at the left of the chart.
r (Right)	Specifies that the legend shall be drawn at the right of the chart.
t (Top)	Specifies that the legend shall be drawn at the top of

Enumeration Value	Description
	the chart.
tr (Top Right)	Specifies that the legend shall be drawn at the top right of the chart.

Referenced By
legendPos@val (§5.7.2.96)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_LegendPos">
   <restriction base="xsd:string">
      <enumeration value="b"/>
      <enumeration value="tr"/>
      <enumeration value="l"/>
      <enumeration value="r"/>
      <enumeration value="t"/>
   </restriction>
</simpleType>
```

5.7.3.25 ST_LogBase (Logarithmic Base)

This simple type specifies that its contents will contain an floating point number greater than or equal to two.

This simple type's contents are a restriction of the XML Schema double datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 2.
- This simple type has a maximum value of less than or equal to 1000.

Referenced By
logBase@val (§5.7.2.99)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_LogBase">
   <restriction base="xsd:double">
      <minInclusive value="2"/>
      <maxInclusive value="1000"/>
   </restriction>
</simpleType>
```

5.7.3.26 ST_MarkerSize (Marker Size)

This simple type specifies that its contents will contain an integer between 2 and 72, whose contents are a size in points.

This simple type's contents are a restriction of the XML Schema unsignedByte datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 2.
- This simple type has a maximum value of less than or equal to 72.

Referenced By
size@val (§5.7.2.193)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_MarkerSize">
   <restriction base="xsd:unsignedByte">
      <minInclusive value="2"/>
      <maxInclusive value="72"/>
   </restriction>
</simpleType>
```

5.7.3.27 ST_MarkerStyle (Marker Style)

This picture shows each of the marker styles. Black is used as the line color, while red is used as the fill color. The height of the dash and the dot are 1/5th of the height of the marker. The width of the dot is 1/2 the width of the marker. The dash and dot have fills as well, but the markers need to be made quite large before these are visible.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
circle (Circle)	Specifies a circle shall be drawn at each data point.
dash (Dash)	Specifies a dash shall be drawn at each data point.
diamond (Diamond)	Specifies a diamond shall be drawn at each data point.
dot (Dot)	Specifies a dot shall be drawn at each data point.
none (None)	Specifies nothing shall be drawn at each data point.

Enumeration Value	Description
picture (Picture)	Specifies a picture shall be drawn at each data point.
plus (Plus)	Specifies a plus shall be drawn at each data point.
square (Square)	Specifies a square shall be drawn at each data point.
star (Star)	Specifies a star shall be drawn at each data point.
triangle (Triangle)	Specifies a triangle shall be drawn at each data point.
x (X)	Specifies an X shall be drawn at each data point.

Referenced By
symbol@val (§5.7.2.206)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_MarkerStyle">
   <restriction base="xsd:string">
      <enumeration value="circle"/>
      <enumeration value="dash"/>
      <enumeration value="diamond"/>
      <enumeration value="dot"/>
      <enumeration value="none"/>
      <enumeration value="picture"/>
      <enumeration value="plus"/>
      <enumeration value="square"/>
      <enumeration value="star"/>
      <enumeration value="triangle"/>
      <enumeration value="x"/>
   </restriction>
</simpleType>
```

5.7.3.28 ST_OfPieType (Pie of Pie or Bar of Pie Type)

This simple type specifies the possible types of Pie or Pie or Bar of Pie charts.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
bar (Bar)	Specifies that the chart is a bar of pie chart, not a pie of pie chart.
pie (Pie)	Specifies that the chart is pie of pie chart, not a bar of pie chart.

Referenced By

Referenced By
ofPieType@val (§5.7.2.128)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_OfPieType">
   <restriction base="xsd:string">
      <enumeration value="pie"/>
      <enumeration value="bar"/>
   </restriction>
</simpleType>
```

5.7.3.29 ST_Order (Order)

This simple type specifies that its contents will contain an integer between 2 and 6, whose contents are the order of the trendline polynomial.

This simple type's contents are a restriction of the XML Schema unsignedByte datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 2.
- This simple type has a maximum value of less than or equal to 6.

Referenced By
order@val (§5.7.2.130)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Order">
   <restriction base="xsd:unsignedByte">
      <minInclusive value="2"/>
      <maxInclusive value="6"/>
   </restriction>
</simpleType>
```

5.7.3.30 ST_Orientation (Orientation)

This simple type specifies the possible ways to place a picture on a data point, series, wall, or floor.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
maxMin (Maximum to Minimum)	Specifies that the values on the axis shall be reversed so they go from maximum to minimum.
minMax (Minimum to Maximum)	Specifies that the axis values shall be in the usual order, minimum to maximum.

Referenced By
orientation@val (§5.7.2.131)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Orientation">
   <restriction base="xsd:string">
      <enumeration value="maxMin"/>
      <enumeration value="minMax"/>
   </restriction>
</simpleType>
```

5.7.3.31 ST_Overlap (Overlap)

This simple type specifies that its contents will contain an integer between -100 and 100, whose contents are a percentage.

This simple type's contents are a restriction of the XML Schema byte datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to -100.
- This simple type has a maximum value of less than or equal to 100.

Referenced By
overlap@val (§5.7.2.132)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Overlap">
   <restriction base="xsd:byte">
      <minInclusive value="-100"/>
      <maxInclusive value="100"/>
   </restriction>
</simpleType>
```

5.7.3.32 ST_PageSetupOrientation (Printed Page Orientation)

This simple type specifies the page orientation of the printed page(s) on which this chart shall appear.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
default (Default Page Orientation)	Specifies that the page orientation shall be the default orientation of the system.
landscape (Landscape Page)	Specifies that the printed page shall have landscape orientation.

Enumeration Value	Description
portrait (Portrait Page)	Specifies that the printed page shall have portrait orientation.

Referenced By
pageSetup@orientation (§5.7.2.135)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PageSetupOrientation">
   <restriction base="xsd:string">
      <enumeration value="default"/>
      <enumeration value="portrait"/>
      <enumeration value="landscape"/>
   </restriction>
</simpleType>
```

5.7.3.33 ST_Period (Period)

This simple type specifies that its contents will contain an integer between 2 and 255.

This simple type's contents are a restriction of the XML Schema unsignedByte datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 2.
- This simple type has a maximum value of less than or equal to 255.

Referenced By
period@val (§5.7.2.136)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Period">
   <restriction base="xsd:unsignedByte">
      <minInclusive value="2"/>
      <maxInclusive value="255"/>
   </restriction>
</simpleType>
```

5.7.3.34 ST_Perspective (Perspective)

This simple type specifies that its contents will contain an integer between 0 and 100, whose contents are a percentage.

This simple type's contents are a restriction of the XML Schema unsignedByte datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 0.
- This simple type has a maximum value of less than or equal to 240.

Referenced By
perspective@val (§5.7.2.137)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Perspective">
   <restriction base="xsd:unsignedByte">
      <minInclusive value="0"/>
      <maxInclusive value="240"/>
   </restriction>
</simpleType>
```

5.7.3.35 ST_PictureFormat (Picture Format)

This simple type specifies the possible ways to place a picture on a data point, series, wall, or floor.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
stack (Stack)	Specifies that the picture shall be stacked.
stackScale (Stack and Scale)	Specifies that the picture shall be stacked after being scaled so that it's height is one Picture Stack Unit. Does not apply to walls or floor.
stretch (Stretch)	Specifies that the picture shall be anisotropic stretched to fill the data point, series, wall or floor.

Referenced By
pictureFormat@val (§5.7.2.138)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PictureFormat">
   <restriction base="xsd:string">
      <enumeration value="stretch"/>
      <enumeration value="stack"/>
      <enumeration value="stackScale"/>
   </restriction>
</simpleType>
```

5.7.3.36 ST_PictureStackUnit (Picture Stack Unit)

This simple type specifies that its contents will contain an floating point number greater than zero.

This simple type's contents are a restriction of the XML Schema double datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than 0.

Referenced By
pictureStackUnit@val (§5.7.2.140)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PictureStackUnit">
   <restriction base="xsd:double">
      <minExclusive value="0"/>
   </restriction>
</simpleType>
```

5.7.3.37 ST_RadarStyle (Radar Style)

This simple type specifies the possible styles of radar chart.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
filled (Filled)	Specifies that the radar chart shall be filled and have lines but no markers.
marker (Marker)	Specifies that the radar chart shall have lines and markers but no fill.
standard (Standard)	Specifies that the radar chart shall have lines but no markers and no fill.

Referenced By
radarStyle@val (§5.7.2.155)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_RadarStyle">
   <restriction base="xsd:string">
      <enumeration value="standard"/>
      <enumeration value="marker"/>
      <enumeration value="filled"/>
   </restriction>
</simpleType>
```

5.7.3.38 ST_RotX (X Rotation)

This simple type specifies that its contents will contain an integer between -90 and 90, whose contents are an angle in degrees.

This simple type's contents are a restriction of the XML Schema byte datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to -90.
- This simple type has a maximum value of less than or equal to 90.

Referenced By
rotX@val (§5.7.2.158)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_RotX">
   <restriction base="xsd:byte">
      <minInclusive value="-90"/>
      <maxInclusive value="90"/>
   </restriction>
</simpleType>
```

5.7.3.39 ST_RotY (Y Rotation)

This simple type specifies that its contents will contain an integer between 0 and 360, whose contents are an angle in degrees.

This simple type's contents are a restriction of the XML Schema unsignedShort datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 0.
- This simple type has a maximum value of less than or equal to 360.

Referenced By
rotY@val (§5.7.2.159)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_RotY">
   <restriction base="xsd:unsignedShort">
      <minInclusive value="0"/>
      <maxInclusive value="360"/>
   </restriction>
</simpleType>
```

5.7.3.40 ST_ScatterStyle (Scatter Style)

This simple type specifies the possible styles of scatter chart.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
line (Line)	Specifies the points on the scatter chart shall be connected with straight lines but markers shall not be drawn.
lineMarker (Line with Markers)	Specifies the points on the scatter chart shall be connected with straight lines and markers shall be drawn.
marker (Marker)	Specifies the points on the scatter chart shall not be connected with lines and markers shall be drawn.
none (None)	Specifies the points on the scatter chart shall not be connected with straight lines and markers shall not be drawn.
smooth (Smooth)	Specifies the the points on the scatter chart shall be connected with smoothed lines and markers shall not be drawn.
smoothMarker (Smooth with Markers)	Specifies the the points on the scatter chart shall be connected with smoothed lines and markers shall be drawn.

Referenced By
scatterStyle@val (§5.7.2.163)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ScatterStyle">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="line"/>
      <enumeration value="lineMarker"/>
      <enumeration value="marker"/>
      <enumeration value="smooth"/>
      <enumeration value="smoothMarker"/>
   </restriction>
</simpleType>
```

5.7.3.41 ST_SecondPieSize (Second Pie Size)

This simple type specifies that its contents will contain an integer between 5 and 200, whose contents consist of a percentage.

This simple type's contents are a restriction of the XML Schema unsignedShort datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 5.
- This simple type has a maximum value of less than or equal to 200.

Referenced By
secondPieSize@val (§5.7.2.165)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SecondPieSize">
   <restriction base="xsd:unsignedShort">
      <minInclusive value="5"/>
      <maxInclusive value="200"/>
   </restriction>
</simpleType>
```

5.7.3.42 ST_Shape (Shape)

This simple type specifies the possible shapes for a 3-D data marker.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
box (Box)	Specifies the chart shall be drawn with a box shape.
cone (Cone)	Specifies the chart shall be drawn as a cone, with the base of the cone on the floor and the point of the cone at the top of the data marker.
coneToMax (Cone to Max)	Specifies the chart shall be drawn with truncated cones such that the point of the cone would be the maximum data value.
cylinder (Cylinder)	Specifies the chart shall be drawn as a cylinder.
pyramid (Pyramid)	Specifies the chart shall be drawn as a rectangular pyramid, with the base of the pyramid on the floor and the point of the pyramid at the top of the data marker.
pyramidToMax (Pyramid to Maximum)	Specifies the chart shall be drawn with truncated cones such that the point of the cone would be the maximum data value.

Referenced By
shape@val (§5.7.2.178)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Shape">
   <restriction base="xsd:string">
      <enumeration value="cone"/>
      <enumeration value="coneToMax"/>
      <enumeration value="box"/>
      <enumeration value="cylinder"/>
      <enumeration value="pyramid"/>
      <enumeration value="pyramidToMax"/>
   </restriction>
</simpleType>
```

5.7.3.43 ST_SizeRepresents (Size Represents)

This simple type specifies the possible ways to represent data as bubble chart sizes.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
area (Bubble Size Represents Area)	Specifies the area of the bubbles shall be proportional to the bubble size value.
w (Bubble Size Represents Width)	Specifies the radius of the bubbles shall be proportional to the bubble size value.

Referenced By
sizeRepresents@val (§5.7.2.194)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SizeRepresents">
   <restriction base="xsd:string">
      <enumeration value="area"/>
      <enumeration value="w"/>
   </restriction>
</simpleType>
```

5.7.3.44 ST_Skip (Skip)

This simple type specifies that its contents will contain an integer greater than or equal to one.

This simple type's contents are a restriction of the XML Schema unsignedShort datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 1.

Referenced By

Referenced By
tickLblSkip@val (§5.7.2.209); tickMarkSkip@val (§5.7.2.210)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Skip">
   <restriction base="xsd:unsignedShort">
      <minInclusive value="1"/>
   </restriction>
</simpleType>
```

5.7.3.45 ST_SplitType (Split Type)

This simple type specifies the possible ways to split a pie of pie or bar of pie chart.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
auto (Default Split)	Specifies the data points shall be split using the default mechanism for this chart type.
cust (Custom Split)	Specifies the data points shall be split between the pie and the second chart according to the Custom Split values.
percent (Split by Percentage)	Specifies the data points shall be split between the pie and the second chart by putting the points with percentage less than Split Position percent in the second chart.
pos (Split by Position)	Specifies the data points shall be split between the pie and the second chart by putting the last Split Position of the data points in the second chart
val (Split by Value)	Specifies the data points shall be split between the pie and the second chart by putting the data points with value less than Split Position in the second chart

Referenced By
splitType@val (§5.7.2.197)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SplitType">
   <restriction base="xsd:string">
      <enumeration value="auto"/>
      <enumeration value="cust"/>
      <enumeration value="percent"/>
      <enumeration value="pos"/>
      <enumeration value="val"/>
   </restriction>
</simpleType>
```

5.7.3.46 ST_Style (Style)

This simple type specifies that its contents will contain an integer between 1 and 48. The value determines the default formatting for all chart elements through the tables described below.

The default font is the minor font as defined by the document's theme. The default font size for each element is the font size of the chart, except for the title which is always 120% the font size of the chart. If the chart does not have a font size set, then the default font size is 10. Axis titles and chart titles are bold by default, while all other chart elements are normal. The default font color is the same as the Axis & Major Gridlines Line Color.

The default line style, fill style, and effect style are determined by the tables below. Each of the default includes a themed line, fill, or effect (None, Subtle, Moderate, or Intense) and a color to be used when applying that line, fill, or effect. In some cases, both the themed formatting and the color vary per style, in other cases they do not. The default line width is determined by the theme, except for lines for data points it is multipled by the line width value given in the table.

Table 1: Chart element defaults

This table lists whether the default is constant or whether it depends on the ST_Style using one of the later tables.

	Line		Fill		Effect	
Chart Element	Themed Line	Color	Themed Fill	Color	Themed Effect	Color
Axis	Subtle	Table 2	No Fill		No Effect	
Axis Title	No Line		No Fill		No Effect	
Chart Area	Table 2		Subtle	Table 3	No Effect	
Chart Title	No Line		No Fill		No Effect	
Data Labels	No Line		No Fill		No Effect	
Data Table	Subtle	Table 2	No Fill		No Effect	
Down Bars	Table 4		Table 4		Table 4	dk1
Fills for Data Points (2-D)	Subtle	Table 5	Table 5		Table 5	dk1
Fills for Data Points (3-D)	Subtle	Table 5	Table 5		Table 5	dk1
Floor	Table 2		Table 3		No Effect	

	Line		Fill		Effect	
Legend	No Line		No Fill		No Effect	
Lines for Data Points	Subtle	Table 5	N/A		No Effect	
Major Gridlines	Subtle	Table 2	N/A		No Effect	
Markers for Data Points	Subtle	Table 5	Table 5		Table 5	dk1
Minor Gridlines	Subtle	Table 2	N/A		No Effect	
Other Lines	Subtle	Table 2	N/A		No Effect	
Plot Area (2-D charts)	No Line		Subtle	Table 3	No Effect	
Plot Area (3-D charts)	No Line		No Fill		No Effect	
Trendline Labels	No Line		No Fill		No Effect	
Up Bars	Table 4		Table 4		Table 4	dk1
Walls	No Line		Table 3		No Effect	

Other Lines includes Drop Lines, Error Bars, High Low Lines, Leader Lines, Series Lines, and Trendlines.

Fills for Data Points (2-D) includes 2-D bar, filled radar, stock, bubble, pie, doughnut and area charts.

Fills for Data Points (3-D) includes all 3-D charts.

Lines for Data Points includes lines on 2-D line, scatter, bubble, and radar charts.

Markers for Data Points includes markers on 2-D line, scatter, and radar charts.

Table 2: Default line formatting per chart style

This table lists line formatting for several chart elements by style.

Style	Axis & Major Gridlines	Minor Gridlines	Chart Area, Data Table, & Floor	Other Lines	Floor & Chart Area
	Color	Color	Color	Color	Themed Line
1-32	tx1	50% tint of tx1	75% tint of tx1	tx1	Subtle
33-34	dk1	50% tint of tx1	75% tint of dk1	dk1	Subtle
35-40	dk1	50% tint of tx1	75% tint of dk1	25% shade of dk1	Subtle
41-48	dk1	90% tint of tx1	lt1	lt1	No Line

Table 3: Default fill formatting per chart style

This table lists fill formatting for several chart elements by style.

Style	Chart Area	Floor, Walls & Plot Area (2-D)	Floor & Walls
	Color	Color	Themed Fill
1-32	bg1	bg1	No Fill
33-34	lt1	20% tint of dk1	Subtle
35-40	lt1	accent1-6	Subtle
41-48	dk1	95% tint of dk1	Subtle

Table 4: Up and down bars default formatting per chart style

This table lists line, fill, and effect formatting for up and down bars by style. The color listed as accent1-6 means that the first style uses accent1, the next uses accent2, up to the sixth uses accent6.

	Up Bars	Down Bars	Up & Down Bars			
			Themed			
Style	Fill Color	Fill Color	Fill	Effect	Line	Line Color
1	25% tint of dk1	85% tint of dk1	Subtle	None	Subtle	tx1
2	5% tint of dk1	95% tint of dk1	Subtle	None	Subtle	tx1
3-8	25% tint of accent1-6	25% shade of accent1-6	Subtle	None	Subtle	tx1
9	25% tint of dk1	85% tint of dk1	Subtle	Subtle	Subtle	tx1
10	5% tint of dk1	95% tint of dk1	Subtle	Subtle	Subtle	tx1
11-16	25% tint of accent1-6	25% shade of accent1-6	Subtle	Subtle	Subtle	tx1
17	25% tint of dk1	85% tint of dk1	Intense	Moderate	No ne	None
18	5% tint of dk1	95% tint of dk1	Intense	Moderate	None	None
19-24	25% tint of accent1-6	25% shade of accent1-6	Intense	Moderate	None	None
25	25% tint of dk1	85% tint of dk1	Intense	Intense	None	None

	Up Bars	Down Bars	Up & Down Bars			
26	5% tint of dk1	95% tint of dk1	Intense	Intense	None	None
27-32	25% tint of accent1-6	25% shade of accent1-6	Intense	Intense	None	None
33	lt1	85% tint of dk1	Subtle	None	Subtle	dk1
34	lt1	95% tint of dk1	Subtle	None	Subtle	dk1
35-40	lt1	25% shade of accent1-6	Subtle	None	Subtle	25% shade of accent1-6
41	25% tint of dk1	85% tint of dk1	Intense	Intense	None	None
42	lt1	dk1	Intense	Intense	None	None
43-48	25% tint of accent1-6	25% shade of accent1-6	Intense	Intense	None	None

Table 5: Default data point formatting per chart style

This table lists line, fill, and effect formatting for data points by style. Some of the formatting is a repeating pattern described in the next table (denoted by "Pattern"). Other use a fade pattern in which the first series is a certain% shade of the color listed and the last series is an certain% tint of the color listed. The intermediate colors are linearly interpolated by shade and tint to the listed color. The starting shade% and ending tint% are determined by the spreadsheet application, where shades are always darker than tints.

Note: A suggested way to implement this percentage is by using the formula: shade/tint percent = -70 + 140 * (SeriesFormattingIndex / (HighestFormattingIndexOfAllSeriesOnAllGraphs + 1)). In this case the series formatting index is the `idx` attribute value for the series. Negative outputs are shades, and positive outputs are tints.

Note: The arrows across the top of the table indicate which columns apply for that row: i.e., for Fills, Lines, or Markers. So, for example, only the last two columns for Lines for Data Points apply.

Fills for Data Points (2-D)	↓	↓		↓	↓		
Fills for Data Points (3-D)		↓	↓	↓	↓		
Lines for Data Points						↓	↓
Markers for Data Points	↓	↓					↓
Effect	Fill	Fill	Fill	Line	Line	Line	Line

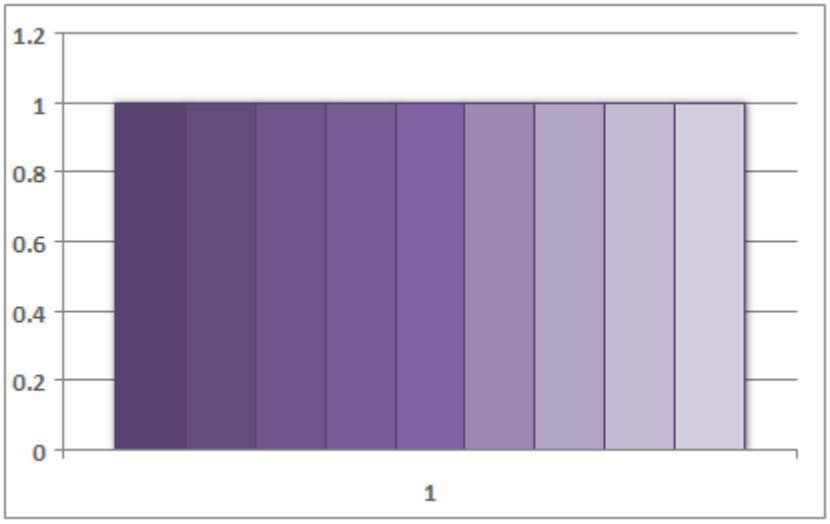
Style	Themed Effect	Themed Fill	Pattern	Themed Fill	Themed Line	Color or Pattern	Width	Patt
1	No Effect	Subtle	1	Subtle	No Line		3	1
2	No Effect	Subtle	2	Subtle	No Line		3	2
3	No Effect	Subtle	accent1 fade	Subtle	No Line		3	acce
4	No Effect	Subtle	accent2 fade	Subtle	No Line		3	acce
5	No Effect	Subtle	accent3 fade	Subtle	No Line		3	acce
6	No Effect	Subtle	accent4 fade	Subtle	No Line		3	acce
7	No Effect	Subtle	accent5 fade	Subtle	No Line		3	acce
8	No Effect	Subtle	accent6 fade	Subtle	No Line		3	acce
9	Subtle	Subtle	1	Subtle	Subtle	lt1	5	1
10	Subtle	Subtle	2	Subtle	Subtle	lt1	5	2
11	Subtle	Subtle	accent1 fade	Subtle	Subtle	lt1	5	acce
12	Subtle	Subtle	accent2 fade	Subtle	Subtle	lt1	5	acce
13	Subtle	Subtle	accent3 fade	Subtle	Subtle	lt1	5	acce
14	Subtle	Subtle	accent4 fade	Subtle	Subtle	lt1	5	acce
15	Subtle	Subtle	accent5 fade	Subtle	Subtle	lt1	5	acce
16	Subtle	Subtle	accent6 fade	Subtle	Subtle	lt1	5	acce
17	Moderate	Intense	1	Subtle	No Line		5	1
18	Moderate	Intense	2	Intense	No Line		5	2
19	Moderate	Intense	accent1 fade	Subtle	No Line		5	acce
20	Moderate	Intense	accent2 fade	Subtle	No Line		5	acce
21	Moderate	Intense	accent3 fade	Subtle	No Line		5	acce
22	Moderate	Intense	accent4 fade	Subtle	No Line		5	acce
23	Moderate	Intense	accent5 fade	Subtle	No Line		5	acce
24	Moderate	Intense	accent6 fade	Subtle	No Line		5	acce

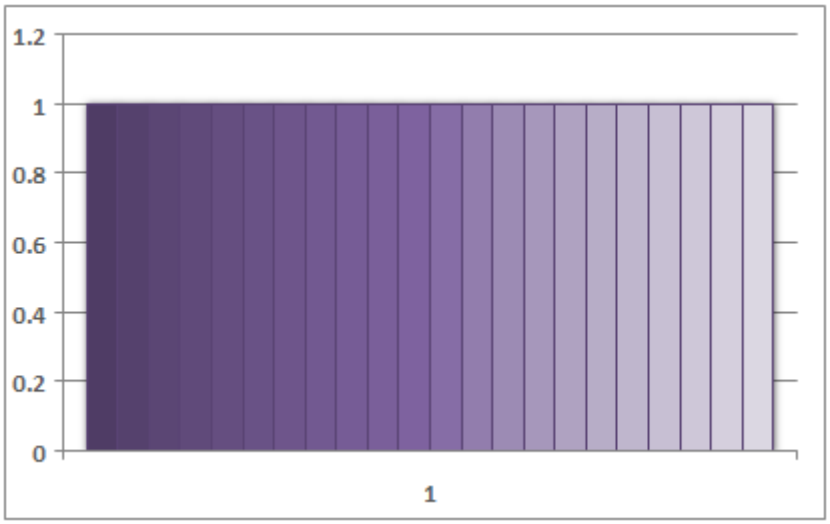
25	Intense	Intense	1	Subtle	No Line		7	1
26	Intense	Intense	2	Intense	No Line		7	2
27	Intense	Intense	accent1 fade	Subtle	No Line		7	acce
28	Intense	Intense	accent2 fade	Subtle	No Line		7	acce
29	Intense	Intense	accent3 fade	Subtle	No Line		7	acce
30	Intense	Intense	accent4 fade	Subtle	No Line		7	acce
31	Intense	Intense	accent5 fade	Subtle	No Line		7	acce
32	Intense	Intense	accent6 fade	Subtle	No Line		7	acce
33	No Effect	Subtle	1	Subtle	Subtle	50% shade of dk1	5	1
34	No Effect	Subtle	2	Subtle	Subtle	Pattern 3	5	2
35	No Effect	Subtle	accent1 fade	Subtle	Subtle	50% shade of accent1	5	acce
36	No Effect	Subtle	accent2 fade	Subtle	Subtle	50% shade of accent2	5	acce
37	No Effect	Subtle	accent3 fade	Subtle	Subtle	50% shade of accent3	5	acce
38	No Effect	Subtle	accent4 fade	Subtle	Subtle	50% shade of accent4	5	acce
39	No Effect	Subtle	accent5 fade	Subtle	Subtle	50% shade of accent5	5	acce
40	No Effect	Subtle	accent6 fade	Subtle	Subtle	50% shade of accent6	5	acce
41	Intense	Intense	4	Subtle	No Line		5	4
42	Intense	Intense	2	Intense	No Line		5	2
43	Intense	Intense	accent1 fade	Subtle	No Line		5	acce
44	Intense	Intense	accent2 fade	Subtle	No Line		5	acce
45	Intense	Intense	accent3 fade	Subtle	No Line		5	acce
46	Intense	Intense	accent4 fade	Subtle	No Line		5	acce
47	Intense	Intense	accent5 fade	Subtle	No Line		5	acce
48	Intense	Intense	accent6 fade	Subtle	No Line		5	acce

Table 6: Default data point formatting per data point

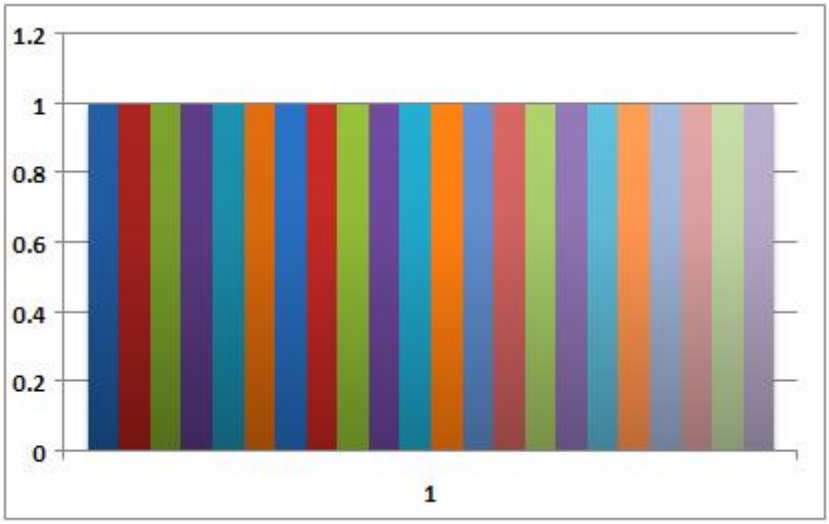
This table contains the formatting patterns used for each data point when there are is one series, or each series when there is just one series. The patterns in this table can repeat, see the pictures below the table for an illustration.

Pattern	Data Point 1	Data Point 2	Data Point 3	Data Point 4	Data Point 5	Data Point 6
1	88.5% tint of dk1	55% tint of dk1	78% tint of dk1	92.5% tint of dk1	70% tint of dk1	30% tint of dk1
2	accent1	accent2	accent3	accent4	Accent5	accent6
3	50% shade of accent1	50% shade of accent2	50% shade of accent3	50% shade of accent4	50% shade of accent5	50% shade of accent6
4	5% tint of dk1	55% tint of dk1	78% tint of dk1	15% tint of dk1	70% tint of dk1	30% tint of dk1





Above are two charts showing the same monochromatic pattern, just with a different number of series. Note that these charts just have tint and shade adjusted for each series beginning and ending at the same values.



Above is a pattern that has 6 distinct accent colors. The colors repeat every 6 data points, but the tint/shade is changed for each set.

This simple type's contents are a restriction of the XML Schema unsignedByte datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 1.
- This simple type has a maximum value of less than or equal to 48.

Referenced By
style@val (§5.7.2.203)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Style">
   <restriction base="xsd:unsignedByte">
      <minInclusive value="1"/>
      <maxInclusive value="48"/>
   </restriction>
</simpleType>
```

5.7.3.47 ST_TextLanguageID (Chart Language Tag)

This simple type specifies a language code value. *Language tags* are a standard mechanism defined by RFC 3066 to specify a language, and consist of a combination of:

- A two-letter ISO 639-1 language code
- A country code

This simple type's contents are a restriction of the XML Schema string datatype.

Referenced By
lang@val (§5.7.2.87)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextLanguageID">
   <restriction base="xsd:string"/>
</simpleType>
```

5.7.3.48 ST_TickLblPos (Tick Label Position)

This simple type specifies the possible positions for tick labels.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
high (High)	Specifies the axis labels shall be at the high end of the perpendicular axis.
low (Low)	Specifies the axis labels shall be at the low end of the perpendicular axis.
nextTo (Next To)	Specifies the axis labels shall be next to the axis.
none (None)	Specifies the axis labels are not drawn.

Referenced By
tickLblPos@val (§5.7.2.208)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TickLblPos">
   <restriction base="xsd:string">
      <enumeration value="high"/>
      <enumeration value="low"/>
      <enumeration value="nextTo"/>
      <enumeration value="none"/>
   </restriction>
</simpleType>
```

5.7.3.49 ST_TickMark (Tick Mark)

This simple type specifies the possible positions for tick marks.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
cross (Cross)	Specifies the tick marks shall cross the axis.
in (Inside)	Specifies the tick marks shall be inside the plot area.
none (None)	Specifies there shall be no tick marks.
out (Outside)	Specifies the tick marks shall be outside the plot area.

Referenced By
majorTickMark@val (§5.7.2.102); minorTickMark@val (§5.7.2.111)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TickMark">
   <restriction base="xsd:string">
      <enumeration value="cross"/>
      <enumeration value="in"/>
      <enumeration value="none"/>
      <enumeration value="out"/>
   </restriction>
</simpleType>
```

5.7.3.50 ST_TimeUnit (Time Unit)

This simple type specifies a unit of time.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
days (Days)	Specifies the chart data shall be shown in days.

Enumeration Value	Description
months (Months)	Specifies the chart data shall be shown in months.
years (Years)	Specifies the chart data shall be shown in years.

Referenced By
baseTimeUnit@val (§5.7.2.18); majorTimeUnit@val (§5.7.2.103); minorTimeUnit@val (§5.7.2.112)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TimeUnit">
   <restriction base="xsd:string">
      <enumeration value="days"/>
      <enumeration value="months"/>
      <enumeration value="years"/>
   </restriction>
</simpleType>
```

5.7.3.51 ST_TrendlineType (Trendline Type)

This simple type specifies all types of trendline which are available for series in a chart.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
exp (Exponential)	Specifies the trendline shall be an exponential curve in the form $y = ab^x$.
linear (Linear)	Specifies the trendline shall be a line in the form $y = mx + b$.
log (Logarithmic)	Specifies the trendline shall be a logarithmic curve in the form $y = a \log x + b$, where log is the natural logarithm.
movingAvg (Moving Average)	Specifies the trendline shall be a moving average of period Period.
poly (Polynomial)	Specifies the trendline shall be a polynomial curve of order Order in the form $y = ax^6 + bx^5 + cx^4 + dx^3 + ex^2 + fx + g$.
power (Power)	Specifies the trendline shall be a power curve in the form $y = ax^b$.

Referenced By
trendlineType@val (§5.7.2.214)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TrendlineType">
   <restriction base="xsd:string">
      <enumeration value="exp"/>
      <enumeration value="linear"/>
      <enumeration value="log"/>
      <enumeration value="movingAvg"/>
      <enumeration value="poly"/>
      <enumeration value="power"/>
   </restriction>
</simpleType>
```

5.7.3.52 ST_Xstring (String With Encoded Characters)

This simple type specifies that its contents will contain a string. Any character in the string may be encoded by use of a \ followed by the hexadecimal representation of the Unicode code points.

This simple type's contents are a restriction of the XML Schema string datatype.

Referenced By
evenFooter (§5.7.2.59); evenHeader (§5.7.2.60); firstFooter (§5.7.2.66); firstHeader (§5.7.2.67); formatCode (§5.7.2.71); name (§5.7.2.118); numFmt@formatCode (§5.7.2.122); oddFooter (§5.7.2.125); oddHeader (§5.7.2.126); pt@formatCode (§5.7.2.151); v (§5.7.2.223); v (§5.7.2.224)

5.8 DrawingML - Chart Drawings

Within a chart, it is sometimes necessary to include DrawingML elements (shapes or pictures) which should be a child object within the parent chart. This relationship allows those elements to optionally be resized with the chart, automatically moved with the chart, etc.

The Chart Drawing namespace acts in this capacity, specifying all information necessary to anchor and display DrawingML objects within a chart.

5.8.1 Table of Contents

This subclause is informative.

End of informative text.

5.8.2 Elements

The following element define the contents of the ChartDrawing namespace:

5.8.2.1 absSizeAnchor (Absolute Anchor Shape Size)

This element specifies that the shape described here to reside within a chart should be sized based on relative anchor points. This is achieved via two elements. The from element specifies the top left corner of the shape bounding box in a RTL(right-to-left) implementation. The ext element then specifies the bottom right corner of the shape bounding box in a RTL(right-to-left) implementation and thus the size of the shape.

Parent Elements
userShapes (§5.7.2.221)

Child Elements	Subclause
cxnSp (Connection Shape)	§5.8.2.9
ext (Shape Extent)	§5.8.2.10
from (Starting Anchor Point)	§5.8.2.11
graphicFrame (Graphic Frame)	§5.8.2.12
grpSp (Group Shape)	§5.8.2.13

Child Elements	Subclause
pic (Picture)	§5.8.2.20
sp (Shape)	§5.8.2.22

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AbsSizeAnchor">
   <sequence>
      <element name="from" type="CT_Marker"/>
      <element name="ext" type="a:CT_PositiveSize2D"/>
      <group ref="EG_ObjectChoices"/>
   </sequence>
</complexType>
```

5.8.2.2 blipFill (Picture Fill)

This element specifies the type of picture fill that the picture object will have. Because a picture has a picture fill already by default, it is possible to have two fills specified for a picture object. An example of this is shown below.

[*Example*: Consider the picture below that has a blip fill applied to it. The image used to fill this picture object has transparent pixels instead of white pixels.

```
<p:pic>
  ..
  <p:blipFill>
    <a:blip r:embed="rId2"/>
    <a:stretch>
      <a:fillRect/>
    </a:stretch>
  </p:blipFill>
  ..
</p:pic>
```

The above picture object is shown as an example of this fill type. End example]

[Example: Consider now the same picture object but with an additional gradient fill applied within the shape properties portion of the picture.

```
<p:pic>
  ..
  <p:blipFill>
    <a:blip r:embed="rId2"/>
    <a:stretch>
      <a:fillRect/>
    </a:stretch>
  </p:blipFill>
  <p:spPr>
    <a:gradFill>
      <a:gsLst>
        <a:gs pos="0">
          <a:schemeClr val="tx2">
            <a:shade val="50000"/>
          </a:schemeClr>
        </a:gs>
        <a:gs pos="39999">
          <a:schemeClr val="tx2">
            <a:tint val="20000"/>
          </a:schemeClr>
        </a:gs>
        <a:gs pos="70000">
          <a:srgbClr val="C4D6EB"/>
        </a:gs>
        <a:gs pos="100000">
          <a:schemeClr val="bg1"/>
        </a:gs>
      </a:gsLst>
    </a:gradFill>
  </p:spPr>
  ..
</p:pic>
```

The above picture object is shown as an example of this double fill type. *End example*]

Parent Elements
pic (§5.8.2.20)

Child Elements	Subclause
blip (Blip)	§5.1.10.13
srcRect (Source Rectangle)	§5.1.10.55
stretch (Stretch)	§5.1.10.56
tile (Tile)	§5.1.10.58

Attributes	Description
dpi (DPI Setting) Namespace: .../drawingml/2006/main	Specifies the DPI (dots per inch) used to calculate the size of the blip. If not present or zero, the DPI in the blip is used. [*Note*: This attribute is primarily used to keep track of the picture quality within a document. There are different levels of quality needed for print than on-screen viewing and thus a need to track this information. *end note*] The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
rotWithShape (Rotate With Shape) Namespace: .../drawingml/2006/main	Specifies that the fill should rotate with the shape. That is, when the shape that has been filled with a picture and the containing shape (say a rectangle) is transformed with a rotation then the fill will be transformed with the same rotation. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BlipFillProperties">
   <sequence>
      <element name="blip" type="CT_Blip" minOccurs="0" maxOccurs="1"/>
      <element name="srcRect" type="CT_RelativeRect" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_FillModeProperties" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="dpi" type="xsd:unsignedInt" use="optional"/>
   <attribute name="rotWithShape" type="xsd:boolean" use="optional"/>
</complexType>
```

5.8.2.3 cNvCxnSpPr (Non-Visual Connection Shape Drawing Properties)

This element specifies the non-visual drawing properties for a connector shape. These non-visual properties are properties that the generating application would utilize when rendering the parent chart.

Parent Elements
nvCxnSpPr (§5.8.2.15)

Child Elements	Subclause
cxnSpLocks (Connection Shape Locks)	§5.1.2.1.11
endCxn (Connection End)	§5.1.2.1.13
extLst (Extension List)	§5.1.2.1.15
stCxn (Connection Start)	§5.1.2.1.36

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualConnectorProperties">
   <sequence>
      <element name="cxnSpLocks" type="CT_ConnectorLocking" minOccurs="0" maxOccurs="1"/>
      <element name="stCxn" type="CT_Connection" minOccurs="0" maxOccurs="1"/>
      <element name="endCxn" type="CT_Connection" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.8.2.4 cNvGraphicFramePr (Non-Visual Graphic Frame Drawing Properties)

This element specifies the non-visual drawing properties for a graphic frame. These non-visual properties are properties that the generating application would utilize when rendering the chart.

Parent Elements
nvGraphicFramePr (§5.8.2.16)

Child Elements	Subclause

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
graphicFrameLocks (Graphic Frame Locks)	§5.1.2.1.19

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualGraphicFrameProperties">
   <sequence>
      <element name="graphicFrameLocks" type="CT_GraphicalObjectFrameLocking" minOccurs="0"
        maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.8.2.5 cNvGrpSpPr (Non-Visual Group Shape Drawing Properties)

This element specifies the non-visual drawing properties for a group shape. These non-visual properties are properties that the generating application would utilize when rendering the chart.

Parent Elements
nvGrpSpPr (§5.8.2.17)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
grpSpLocks (Group Shape Locks)	§5.1.2.1.21

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualGroupDrawingShapeProps">
   <sequence>
      <element name="grpSpLocks" type="CT_GroupLocking" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.8.2.6 cNvPicPr (Non-Visual Picture Drawing Properties)

This element specifies the non-visual properties for the picture canvas. These properties are to be used by the generating application to determine how certain properties are to be changed for the picture object in question.

[*Example*: Consider the following DrawingML.

```
<p:pic>
  ..
  <p:nvPicPr>
    <p:cNvPr id="4" name="Lilly_by_Lisher.jpg"/>
    <p:cNvPicPr>
      <a:picLocks noChangeAspect="1"/>
```

```
      </p:cNvPicPr>
      <p:nvPr/>
    </p:nvPicPr>
    ..
  </p:pic>
```

End example]

Parent Elements
nvPicPr (§5.8.2.18)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
picLocks (Picture Locks)	§5.1.2.1.31

Attributes	Description
preferRelativeResize (Relative Resize Preferred) Namespace: .../drawingml/2006/main	Specifies if the user interface should show the resizing of the picture based on the picture's current size or its original size. If this attribute is set to true, then scaling will be relative to the original picture size as opposed to the current picture size. [*Example*: Consider the case where a picture has been resized within a document and is now 50% of the originally inserted picture size. Now if the user chooses to make a later adjustment to the size of this picture within the generating application, then the value of this attribute should be checked. If this attribute is set to true then a value of 50% will be shown. Similarly, if this attribute is set to false, then a value of 100% should be shown because the picture has not yet been resized from its current (smaller) size. *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualPictureProperties">
   <sequence>
      <element name="picLocks" type="CT_PictureLocking" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="preferRelativeResize" type="xsd:boolean" use="optional" default="true"/>
</complexType>
```

5.8.2.7 cNvPr (Non-Visual Drawing Properties)

This element specifies non-visual canvas properties. This allows for additional information that does not affect the appearance of the picture to be stored.

[Example: Consider the following ChartDrawingML.

```
<cdr:pic>
  ..
  <cdr:nvPicPr>
    <cdr:cNvPr id="4" name="Lilly_by_Lisher.jpg"/>
  </cdr:nvPicPr>
  ..
</cdr:pic>
```

End example]

Parent Elements
nvCxnSpPr (§5.8.2.15); nvGraphicFramePr (§5.8.2.16); nvGrpSpPr (§5.8.2.17); nvPicPr (§5.8.2.18); nvSpPr (§5.8.2.19)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
hlinkClick (Click Hyperlink)	§5.1.5.3.5
hlinkHover (Hyperlink for Hover)	§5.1.2.1.23

Attributes	Description
descr (Alternative Text for Object) Namespace: .../drawingml/2006/main	Specifies alternative text for the current DrawingML object, for use by assistive technologies or applications which will not display the current object. If this element is omitted, then no alternative text is present for the parent object. [*Example*: Consider a DrawingML object defined as follows: `<… descr="A picture of a bowl of fruit">` The descr attribute contains alternative text which may be used in place of the actual DrawingML object. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
hidden (Hidden) Namespace: .../drawingml/2006/main	Specifies whether this DrawingML object shall be displayed. When a DrawingML object is displayed within a document, that object may be hidden (i.e., present, but not visible). This attribute shall determine whether the object shall be rendered or made hidden. [*Note*: An application may have settings which allow this object to be viewed. *end note*] If this attribute is omitted, then the parent DrawingML object shall be displayed (i.e., not hidden).

Attributes	Description
	[*Example*: Consider an inline DrawingML object which shall be hidden within the document's content. This setting would be specified as follows: `<… hidden="true" />` The hidden attribute has a value of `true`, which specifies that the DrawingML object is hidden and not displayed when the document is displayed. *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
id (Unique Identifier) Namespace: …/drawingml/2006/main	Specifies a unique identifier for the current DrawingML object within the current document. This ID may be used to assist in uniquely identifying this object so that it can be referred to by other parts of the document. If multiple objects within the same document share the same id attribute value, then the document shall be considered non-conformant. [*Example*: Consider a DrawingML object defined as follows: `<… id="10" … >` The id attribute has a value of `10`, which is the unique identifier for this DrawingML object. *end example*] The possible values for this attribute are defined by the ST_DrawingElementId simple type (§5.1.12.19).
name (Name) Namespace: …/drawingml/2006/main	Specifies the name of the object. [*Note*: Typically, this will be used to store the original file name of a picture object. *end note*] [*Example*: Consider a DrawingML object defined as follows: `< … name="foo.jpg" >` The name attribute has a value of `foo.jpg`, which is the name of this DrawingML object. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualDrawingProps">
   <sequence>
      <element name="hlinkClick" type="CT_Hyperlink" minOccurs="0" maxOccurs="1"/>
      <element name="hlinkHover" type="CT_Hyperlink" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="id" type="ST_DrawingElementId" use="required"/>
   <attribute name="name" type="xsd:string" use="required"/>
   <attribute name="descr" type="xsd:string" use="optional" default=""/>
   <attribute name="hidden" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.8.2.8 cNvSpPr (Non-Visual Shape Drawing Properties)

This element specifies the non-visual drawing properties for a shape. These properties are to be used by the generating application to determine how the shape should be dealt with.

[Example: Consider the shape that has a shape lock applied to it.

```
<cdr:sp>
  <cdr:nvSpPr>
    <cdr:cNvPr id="2" name="Rectangle 1"/>
    <cdr:cNvSpPr>
      <a:spLocks noGrp="1"/>
    </cdr:cNvSpPr>
   </cdr:nvSpPr>
   ..
</cdr:sp>
```

This shape lock is stored within the non-visual drawing properties for this shape. End example]

Parent Elements
nvSpPr (§5.8.2.19)

Child Elements	Subclause
extLst (Extension List)	§5.1.2.1.15
spLocks (Shape Locks)	§5.1.2.1.34

Attributes	Description
txBox (Text Box) Namespace: .../drawingml/200	Specifies that the corresponding shape is a text box and thus should be treated as such by the generating application. If this attribute is omitted then it is assumed that the corresponding shape is not specifically a text box.

Attributes	Description
6/main	[*Note*: Because a shape is not specified to be a text box does not mean that it cannot have text attached to it. A text box is merely a specialized shape with specific properties. *end note*] The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NonVisualDrawingShapeProps">
   <sequence>
      <element name="spLocks" type="CT_ShapeLocking" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="txBox" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.8.2.9 cxnSp (Connection Shape)

This element specifies a connection shape that is used to connect two sp elements. Once a connection is specified using a cxnSp, it is left to the generating application to determine the exact path the connector will take. That is the connector routing algorithm is left up to the generating application as the desired path might be different depending on the specific needs of the application.

[Example: Consider the following connector shape that connects two regular shapes.

```
<cp:grpSp>
  ..
  <cp:sp>
    <cp:nvSpPr>
      <cp:cNvPr id="1" name="Rectangle 1"/>
      <cp:cNvSpPr/>
      <cp:nvPr/>
    </cp:nvSpPr>
    ..
  </cp:sp>
  <cp:sp>
    <cp:nvSpPr>
      <cp:cNvPr id="2" name="Rectangle 2"/>
      <cp:cNvSpPr/>
      <cp:nvPr/>
    </cp:nvSpPr>
```

```
    ..
  </cp:sp>
  <cp:cxnSp>
    <cp:nvCxnSpPr>
      <cp:cNvPr id="3" name="Elbow Connector 3"/>
      <cp:cNvCxnSpPr>
        <a:stCxn id="1" idx="3"/>
        <a:endCxn id="2" idx="1"/>
      </cp:cNvCxnSpPr>
      <cp:nvPr/>
    </cp:nvCxnSpPr>
    …
  </cp:cxnSp>
</cp:grpSp>
```

End example]

Parent Elements
absSizeAnchor (§5.8.2.1); grpSp (§5.8.2.13); relSizeAnchor (§5.8.2.21)

Child Elements	Subclause
nvCxnSpPr (Connector Non Visual Properties)	§5.8.2.15
spPr (Shape Properties)	§5.8.2.23
style (Shape Style)	§5.8.2.24

Attributes	Description

fPublished (Publish to Server)	Specifies whether the shape shall be published with the worksheet when sent to the spreadsheet server. This is for use when interfacing with a document server.
	
	[*Example*: Consider the following shape that will not be published with the worksheet when it is published back on the spreadsheet server.
	`<cdr:relSizeAnchor>`
	`..`
	`<cdr:sp fPublished="0">`
	`..`
	`</cdr:sp>`
	`..`
	`</cdr:relSizeAnchor>`
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

macro (Reference	This element specifies the custom function associated with the chart. [*Example*: A macro

Attributes	Description
to Custom Function)	script, add-in function, and so on. *end example*] The format of this string shall be application-defined, and should be ignored if not understood. [*Example*: `<cdr:... macro="DoWork()" >` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Connector">
   <sequence>
      <element name="nvCxnSpPr" type="CT_ConnectorNonVisual" minOccurs="1" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="1" maxOccurs="1"/>
      <element name="style" type="a:CT_ShapeStyle" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="macro" type="xsd:string" use="optional"/>
   <attribute name="fPublished" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.8.2.10 ext (Shape Extent)

This element describes the length and width properties for how far a drawing element should extend for.

Parent Elements
absSizeAnchor (§5.8.2.1)

Attributes	Description
cx (Extent Length) Namespace: .../drawingml/2006/main	Specifies the length of the extents rectangle in EMUs. This rectangle shall dictate the size of the object as displayed (the result of any scaling to the original object). [*Example*: Consider a DrawingML object specified as follows: `<… cx="1828800" cy="200000"/>` The cx attributes specifies that this object has a height of 1828800 EMUs (English Metric Units). *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).
cy (Extent Width)	Specifies the width of the extents rectangle in EMUs. This rectangle shall dictate the size

Attributes	Description
Namespace: .../drawingml/2006/main	of the object as displayed (the result of any scaling to the original object). [*Example*: Consider a DrawingML object specified as follows: `< … cx="1828800" cy="200000"/>` The cy attribute specifies that this object has a width of 200000 EMUs (English Metric Units). *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PositiveSize2D">
   <attribute name="cx" type="ST_PositiveCoordinate" use="required"/>
   <attribute name="cy" type="ST_PositiveCoordinate" use="required"/>
</complexType>
```

5.8.2.11 from (Starting Anchor Point)

This element specifies the first anchor point for the drawing element. This will be used to anchor the top and left sides of the shape within the chart. That is when the corresponding chart is adjusted, the shape will also be adjusted.

[Example: Consider the following Chart Drawing content:

```
<cdr:relSizeAnchor>
  <cdr:from>
    <cdr:x>0.04583</cdr:x>
    <cdr:y>0.53125</cdr:y>
  </cdr:from>
  <cdr:to>
    <cdr:x>0.24583</cdr:x>
    <cdr:y>0.86458</cdr:y>
  </cdr:to>
  <cdr:sp macro="" textlink="">
  ...
  </cdr:sp>
</cdr:relSizeAnchor>
```

The above example shows the first anchor point being specified via the from element. End example]

Parent Elements
absSizeAnchor (§5.8.2.1); relSizeAnchor (§5.8.2.21)

Child Elements	Subclause
x (Relative X Coordinate)	§5.8.2.27
y (Relative Y Coordinate)	§5.8.2.29

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Marker">
   <sequence>
      <element name="x" type="ST_MarkerCoordinate" minOccurs="1" maxOccurs="1"/>
      <element name="y" type="ST_MarkerCoordinate" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.8.2.12 graphicFrame (Graphic Frame)

This element specifies the existence of a graphics frame. This frame contains a graphic that was generated by an external source and needs a container in which to be displayed on the slide surface.

Parent Elements
absSizeAnchor (§5.8.2.1); grpSp (§5.8.2.13); relSizeAnchor (§5.8.2.21)

Child Elements	Subclause
graphic (Graphic Object)	§5.1.2.1.16
nvGraphicFramePr (Non-Visual Graphic Frame Properties)	§5.8.2.16
xfrm (Graphic Frame Transform)	§5.8.2.28

Attributes	Description

fPublished (Publish To Server)	Specifies whether the shape shall be published with the worksheet when sent to the spreadsheet server. This is for use when interfacing with a document server.
	
	[*Example*: Consider the following shape that will not be published with the worksheet when it is published back on the spreadsheet server.
	`<cdr:relSizeAnchor>`
	`..`
	`<cdr:sp fPublished="0">`
	`..`
	`</cdr:sp>`
	`..`
	`</cdr:relSizeAnchor>`
	end example]
	
	The possible values for this attribute are defined by the XML Schema boolean datatype.

macro (Reference to Custom Function)	This element specifies the custom function associated with the chart. [*Example*: A macro script, add-in function, and so on. *end example*]

Attributes	Description
	The format of this string shall be application-defined, and should be ignored if not understood. [*Example*: `<cdr:... macro="DoWork()" >` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GraphicFrame">
   <sequence>
      <element name="nvGraphicFramePr" type="CT_GraphicFrameNonVisual" minOccurs="1" maxOccurs="1"/>
      <element name="xfrm" type="a:CT_Transform2D" minOccurs="1" maxOccurs="1"/>
      <element ref="a:graphic" minOccurs="1" maxOccurs="1"/>
   </sequence>
   <attribute name="macro" type="xsd:string" use="optional"/>
   <attribute name="fPublished" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.8.2.13 grpSp (Group Shape)

This element specifies a group shape that represents many shapes grouped together. This shape is to be treated just as if it were a regular shape but instead of being described by a single geometry it is made up of all the shape geometries encompassed within it. Within a group shape each of the shapes that make up the group are specified just as they normally would. The idea behind grouping elements however is that a single transform can apply to many shapes at the same time.

[Example: Consider the following group shape.

```
<cdr:grpSp>
  <cdr:nvGrpSpPr>
    <cdr:cNvPr id="10" name="Group 9"/>
    <cdr:cNvGrpSpPr/>
    <cdr:nvPr/>
  </cdr:nvGrpSpPr>
  <cdr:grpSpPr>
    <a:xfrm>
      <a:off x="838200" y="990600"/>
      <a:ext cx="2426208" cy="978408"/>
      <a:chOff x="838200" y="990600"/>
      <a:chExt cx="2426208" cy="978408"/>
   </a:xfrm>
```

```
    </cdr:grpSpPr>
    <cdr:sp>
    ..
    </cdr:sp>
    <cdr:sp>
    ..
    </cdr:sp>
    <cdr:sp>
    ..
    </cdr:sp>
  </cdr:grpSp>
```

In the above example we see three shapes specified within a single group. These three shapes have their position and sizes specified just as they normally would within the shape tree. The generating application should apply the transformation after the bounding box for the group shape has been calculated. end example]

Parent Elements
absSizeAnchor (§5.8.2.1); grpSp (§5.8.2.13); relSizeAnchor (§5.8.2.21)

Child Elements	Subclause
cxnSp (Connection Shape)	§5.8.2.9
graphicFrame (Graphic Frame)	§5.8.2.12
grpSp (Group Shape)	§5.8.2.13
grpSpPr (Group Shape Properties)	§5.8.2.14
nvGrpSpPr (Non-Visual Group Shape Properties)	§5.8.2.17
pic (Picture)	§5.8.2.20
sp (Shape)	§5.8.2.22

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GroupShape">
   <sequence>
      <element name="nvGrpSpPr" type="CT_GroupShapeNonVisual" minOccurs="1" maxOccurs="1"/>
      <element name="grpSpPr" type="a:CT_GroupShapeProperties" minOccurs="1" maxOccurs="1"/>
      <choice minOccurs="0" maxOccurs="unbounded">
         <element name="sp" type="CT_Shape"/>
         <element name="grpSp" type="CT_GroupShape"/>
         <element name="graphicFrame" type="CT_GraphicFrame"/>
         <element name="cxnSp" type="CT_Connector"/>
         <element name="pic" type="CT_Picture"/>
      </choice>
   </sequence>
</complexType>
```

5.8.2.14 grpSpPr (Group Shape Properties)

This element specifies the properties that are to be common across all of the shapes within the corresponding group. If there are any conflicting properties within the group shape properties and the individual shape properties then the individual shape properties should take precedence.

Parent Elements
grpSp (§5.8.2.13)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
effectDag (Effect Container)	§5.1.10.25
effectLst (Effect Container)	§5.1.10.26
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
scene3d (3D Scene Properties)	§5.1.4.1.26
solidFill (Solid Fill)	§5.1.10.54
xfrm (2D Transform for Grouped Objects)	§5.1.9.5

Attributes	Description
bwMode (Black and White Mode) Namespace: .../drawingml/2006/main	Specifies that the group shape should be rendered using only black and white coloring. That is the coloring information for the group shape should be converted to either black or white when rendering the corresponding shapes. No gray is to be used in rendering this image, only stark black and stark white. [*Note*: This does not mean that the group shapes themselves are stored with only black and white color information. This attribute instead sets the rendering mode that the shapes will use when rendering. *end note*] The possible values for this attribute are defined by the ST_BlackWhiteMode simple type (§5.1.12.10).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GroupShapeProperties">
   <sequence>
      <element name="xfrm" type="CT_GroupTransform2D" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_FillProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_EffectProperties" minOccurs="0" maxOccurs="1"/>
      <element name="scene3d" type="CT_Scene3D" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="bwMode" type="ST_BlackWhiteMode" use="optional"/>
</complexType>
```

5.8.2.15 nvCxnSpPr (Connector Non Visual Properties)

This element specifies all non-visual properties for a connection shape. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a connection shape. This allows for additional information that does not affect the appearance of the connection shape to be stored.

Parent Elements
cxnSp (§5.8.2.9)

Child Elements	Subclause
cNvCxnSpPr (Non-Visual Connection Shape Drawing Properties)	§5.8.2.3
cNvPr (Non-Visual Drawing Properties)	§5.8.2.7

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ConnectorNonVisual">
   <sequence>
      <element name="cNvPr" type="a:CT_NonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
      <element name="cNvCxnSpPr" type="a:CT_NonVisualConnectorProperties" minOccurs="1"
        maxOccurs="1"/>
   </sequence>
</complexType>
```

5.8.2.16 nvGraphicFramePr (Non-Visual Graphic Frame Properties)

This element specifies all non-visual properties for a graphic frame. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a graphic frame. This allows for additional information that does not affect the appearance of the graphic frame to be stored.

Parent Elements
graphicFrame (§5.8.2.12)

Child Elements	Subclause
cNvGraphicFramePr (Non-Visual Graphic Frame Drawing Properties)	§5.8.2.4
cNvPr (Non-Visual Drawing Properties)	§5.8.2.7

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GraphicFrameNonVisual">
   <sequence>
      <element name="cNvPr" type="a:CT_NonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
      <element name="cNvGraphicFramePr" type="a:CT_NonVisualGraphicFrameProperties" minOccurs="1"
        maxOccurs="1"/>
   </sequence>
</complexType>
```

5.8.2.17 nvGrpSpPr (Non-Visual Group Shape Properties)

This element specifies all non-visual properties for a group shape. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a group shape. This allows for additional information that does not affect the appearance of the group shape to be stored.

Parent Elements
grpSp (§5.8.2.13)

Child Elements	Subclause
cNvGrpSpPr (Non-Visual Group Shape Drawing Properties)	§5.8.2.5
cNvPr (Non-Visual Drawing Properties)	§5.8.2.7

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GroupShapeNonVisual">
   <sequence>
      <element name="cNvPr" type="a:CT_NonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
      <element name="cNvGrpSpPr" type="a:CT_NonVisualGroupDrawingShapeProps" minOccurs="1"
        maxOccurs="1"/>
   </sequence>
</complexType>
```

5.8.2.18 nvPicPr (Non-Visual Picture Properties)

This element specifies the non visual properties for a picture. This allows for additional information that does not affect the appearance of the picture to be stored.

[*Example*: Consider the following DrawingML.

```
<p:pic>
  ..
  <p:nvPicPr>
```

```
      ..
    </p:nvPicPr>
    ..
  </p:pic>
```

End example]

Parent Elements
pic (§5.8.2.20)

Child Elements	Subclause
cNvPicPr (Non-Visual Picture Drawing Properties)	§5.8.2.6
cNvPr (Non-Visual Drawing Properties)	§5.8.2.7

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PictureNonVisual">
   <sequence>
      <element name="cNvPr" type="a:CT_NonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
      <element name="cNvPicPr" type="a:CT_NonVisualPictureProperties" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.8.2.19 nvSpPr (Non-Visual Shape Properties)

This element specifies all non-visual properties for a shape. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a shape. This allows for additional information that does not affect the appearance of the shape to be stored.

Parent Elements
sp (§5.8.2.22)

Child Elements	Subclause
cNvPr (Non-Visual Drawing Properties)	§5.8.2.7
cNvSpPr (Non-Visual Shape Drawing Properties)	§5.8.2.8

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShapeNonVisual">
   <sequence>
      <element name="cNvPr" type="a:CT_NonVisualDrawingProps" minOccurs="1" maxOccurs="1"/>
      <element name="cNvSpPr" type="a:CT_NonVisualDrawingShapeProps" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.8.2.20 pic (Picture)

This element specifies the existence of a picture object within the document.

[Example: Consider the following ChartDrawingML that specifies the existence of a picture within a document. This picture can have non-visual properties, a picture fill as well as shape properties attached to it.

```
<cdr:pic>
  <cdr:nvPicPr>
    <cdr:cNvPr id="4" name="lake.JPG" descr="Picture of a Lake" />
    <cdr:cNvPicPr>
      <a:picLocks noChangeAspect="1"/>
    </cdr:cNvPicPr>
    <cdr:nvPr/>
  </cdr:nvPicPr>
  <cdr:blipFill>
  ...
  </cdr:blipFill>
  <cdr:spPr>
  ...
  </cdr:spPr>
</cdr:pic>
```

End example]

Parent Elements
absSizeAnchor (§5.8.2.1); grpSp (§5.8.2.13); relSizeAnchor (§5.8.2.21)

Child Elements	Subclause
blipFill (Picture Fill)	§5.8.2.2
nvPicPr (Non-Visual Picture Properties)	§5.8.2.18
spPr (Shape Properties)	§5.8.2.23
style (Shape Style)	§5.8.2.24

Attributes	Description

fPublished (Publish to Server)	Specifies whether the shape shall be published with the worksheet when sent to the spreadsheet server. This is for use when interfacing with a document server.
	
	[*Example*: Consider the following shape that will not be published with the worksheet when it is published back on the spreadsheet server.
	`<cdr:relSizeAnchor>`
	`..`

Attributes	Description
	<cdr:sp fPublished="0"> .. </cdr:sp> .. </cdr:relSizeAnchor> *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
macro (Reference to Custom Function)	This element specifies the custom function associated with the chart. [*Example*: A macro script, add-in function, and so on. *end example*] The format of this string shall be application-defined, and should be ignored if not understood. [*Example*: <cdr:... macro="DoWork()" > *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Picture">
   <sequence>
      <element name="nvPicPr" type="CT_PictureNonVisual" minOccurs="1" maxOccurs="1"/>
      <element name="blipFill" type="a:CT_BlipFillProperties" minOccurs="1" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="1" maxOccurs="1"/>
      <element name="style" type="a:CT_ShapeStyle" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="macro" type="xsd:string" use="optional" default=""/>
   <attribute name="fPublished" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.8.2.21 relSizeAnchor (Relative Anchor Shape Size)

This element specifies that the shape described here to reside within a chart should be sized based on relative anchor points. This is achieved via two elements. The from element specifies the top left corner of the shape bounding box in a RTL(right-to-left) implementation. The to element then specifies the bottom right corner of the shape bounding box in a RTL(right-to-left) implementation and thus the size of the shape.

Parent Elements
userShapes (§5.7.2.221)

Child Elements	Subclause

Child Elements	Subclause
cxnSp (Connection Shape)	§5.8.2.9
from (Starting Anchor Point)	§5.8.2.11
graphicFrame (Graphic Frame)	§5.8.2.12
grpSp (Group Shape)	§5.8.2.13
pic (Picture)	§5.8.2.20
sp (Shape)	§5.8.2.22
to (Ending Anchor Point)	§5.8.2.25

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RelSizeAnchor">
   <sequence>
      <element name="from" type="CT_Marker"/>
      <element name="to" type="CT_Marker"/>
      <group ref="EG_ObjectChoices"/>
   </sequence>
</complexType>
```

5.8.2.22 sp (Shape)

This element specifies the existence of a single shape. A shape can either be a preset or a custom geometry, defined using the DrawingML framework. In addition to geometry, each shape can have both visual and non-visual properties attached. Text and corresponding styling information can also be attached to a shape. This shape is specified along with all other shapes within either the shape tree or group shape elements.

Parent Elements
absSizeAnchor (§5.8.2.1); grpSp (§5.8.2.13); relSizeAnchor (§5.8.2.21)

Child Elements	Subclause
nvSpPr (Non-Visual Shape Properties)	§5.8.2.19
spPr (Shape Properties)	§5.8.2.23
style (Shape Style)	§5.8.2.24
txBody (Shape Text Body)	§5.8.2.26

Attributes	Description
fLocksText (Lock Text)	Specifies whether to allow for the editing of text within this shape when the worksheet on which the shape resides has been protected as defined by SpreadsheetML. This allows for the specifying of locked or "protected" text on a per-shape basis within a spreadsheet document. If this attribute is not specified then a value of 0, or false will be assumed. [*Example*: Consider the following shape that does not have locked text on.

Attributes	Description
	<cdr:relSizeAnchor> .. <cdr:sp fLocksText="0"> .. </cdr:sp> .. </cdr:relSizeAnchor> *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
fPublished (Publish to Server)	Specifies whether the shape shall be published with the worksheet when sent to the spreadsheet server. This is for use when interfacing with a document server. [*Example*: Consider the following shape that will not be published with the worksheet when it is published back on the spreadsheet server. <cdr:relSizeAnchor> .. <cdr:sp fPublished="0"> .. </cdr:sp> .. </cdr:relSizeAnchor> *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
macro (Reference to Custom Function)	This element specifies the custom function associated with the chart. [*Example*: A macro script, add-in function, and so on. *end example*] The format of this string shall be application-defined, and should be ignored if not understood. [*Example*: <cdr:... macro="DoWork()" > *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
textlink (Text Link)	Specifies whether the text contained within this shape is linked to a cell within the spreadsheet. That is the text within the shape will have the value defined in the referenced spreadsheet cell. [*Example*: Consider the following shape with text linked to cell A1. <cdr:relSizeAnchor> .. <cdr:sp macro="" textlink="A1">

Attributes	Description
	.. </cdr:sp> .. </cdr:relSizeAnchor> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Shape">
   <sequence>
      <element name="nvSpPr" type="CT_ShapeNonVisual" minOccurs="1" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="1" maxOccurs="1"/>
      <element name="style" type="a:CT_ShapeStyle" minOccurs="0" maxOccurs="1"/>
      <element name="txBody" type="a:CT_TextBody" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="macro" type="xsd:string" use="optional"/>
   <attribute name="textlink" type="xsd:string" use="optional"/>
   <attribute name="fLocksText" type="xsd:boolean" use="optional" default="true"/>
   <attribute name="fPublished" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.8.2.23 spPr (Shape Properties)

This element specifies the visual shape properties that can be applied to a special shape such as a connector shape or picture. These are the same properties that are allowed to describe the visual properties of a shape but are used here to describe additional object-specific properties within a document. This allows for these shapes to have both the properties of a shape as well as specific properties that are unique to only them.

Parent Elements
cxnSp (§5.8.2.9); pic (§5.8.2.20); sp (§5.8.2.22)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
custGeom (Custom Geometry)	§5.1.11.8
effectDag (Effect Container)	§5.1.10.25
effectLst (Effect Container)	§5.1.10.26
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35
ln (Outline)	§5.1.2.1.24
noFill (No Fill)	§5.1.10.44

Child Elements	Subclause
pattFill (Pattern Fill)	§5.1.10.47
prstGeom (Preset geometry)	§5.1.11.18
scene3d (3D Scene Properties)	§5.1.4.1.26
solidFill (Solid Fill)	§5.1.10.54
sp3d (Apply 3D shape properties)	§5.1.7.12
xfrm (2D Transform for Individual Objects)	§5.1.9.6

Attributes	Description
bwMode (Black and White Mode) Namespace: .../drawingml/2006/main	Specifies that the picture should be rendered using only black and white coloring. That is the coloring information for the picture should be converted to either black or white when rendering the picture. No gray is to be used in rendering this image, only stark black and stark white. [*Note*: This does not mean that the picture itself that is stored within the file is necessarily a black and white picture. This attribute instead sets the rendering mode that the picture will have applied to when rendering. *end note*] The possible values for this attribute are defined by the ST_BlackWhiteMode simple type (§5.1.12.10).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShapeProperties">
   <sequence>
      <element name="xfrm" type="CT_Transform2D" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_Geometry" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_FillProperties" minOccurs="0" maxOccurs="1"/>
      <element name="ln" type="CT_LineProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_EffectProperties" minOccurs="0" maxOccurs="1"/>
      <element name="scene3d" type="CT_Scene3D" minOccurs="0" maxOccurs="1"/>
      <element name="sp3d" type="CT_Shape3D" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="bwMode" type="ST_BlackWhiteMode" use="optional"/>
</complexType>
```

5.8.2.24 style (Shape Style)

The element specifies the style that will be applied to a shape and the corresponding references for each of the style components such as lines and fills.

Parent Elements
cxnSp (§5.8.2.9); pic (§5.8.2.20); sp (§5.8.2.22)

Child Elements	Subclause
effectRef (Effect Reference)	§5.1.4.2.8
fillRef (Fill Reference)	§5.1.4.2.10
fontRef (Font Reference)	§5.1.4.1.17
lnRef (Line Reference)	§5.1.4.2.19

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShapeStyle">
   <sequence>
      <element name="lnRef" type="CT_StyleMatrixReference" minOccurs="1" maxOccurs="1"/>
      <element name="fillRef" type="CT_StyleMatrixReference" minOccurs="1" maxOccurs="1"/>
      <element name="effectRef" type="CT_StyleMatrixReference" minOccurs="1" maxOccurs="1"/>
      <element name="fontRef" type="CT_FontReference" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.8.2.25 to (Ending Anchor Point)

This element specifies the second anchor point for the drawing element. This will be used to anchor the bottom and right sides of the shape within the spreadsheet. That is when the corresponding chart is adjusted, the shape will also be adjusted.

[Example: Consider the following ChartDrawingML

```
<cdr:relSizeAnchor>
  <cdr:from>
    <cdr:x>0.04583</cdr:x>
    <cdr:y>0.53125</cdr:y>
  </cdr:from>
  <cdr:to>
    <cdr:x>0.24583</cdr:x>
    <cdr:y>0.86458</cdr:y>
  </cdr:to>
  <cdr:sp macro="" textlink="">
  ...
  </cdr:sp>
</cdr:relSizeAnchor>
```

The above example shows the second anchor point being specified via the to element. End example]

Parent Elements
relSizeAnchor (§5.8.2.21)

Child Elements	Subclause

Child Elements	Subclause
x (Relative X Coordinate)	§5.8.2.27
y (Relative Y Coordinate)	§5.8.2.29

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Marker">
   <sequence>
      <element name="x" type="ST_MarkerCoordinate" minOccurs="1" maxOccurs="1"/>
      <element name="y" type="ST_MarkerCoordinate" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.8.2.26 txBody (Shape Text Body)

This element specifies the existence of text to be contained within the corresponding shape. All visible text and visible text related properties are contained within this element. There can be multiple paragraphs and within paragraphs multiple runs of text.

Parent Elements
sp (§5.8.2.22)

Child Elements	Subclause
bodyPr (Body Properties)	§5.1.5.1.1
lstStyle (Text List Styles)	§5.1.5.4.12
p (Text Paragraphs)	§5.1.5.2.6

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextBody">
   <sequence>
      <element name="bodyPr" type="CT_TextBodyProperties" minOccurs="1" maxOccurs="1"/>
      <element name="lstStyle" type="CT_TextListStyle" minOccurs="0" maxOccurs="1"/>
      <element name="p" type="CT_TextParagraph" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.8.2.27 x (Relative X Coordinate)

This element specifies the relative x coordinate that is used to define the percentage-based horizontal position for a shape within a chart drawing object. The coordinate boundaries are specified within the corresponding simple type listed below.

The possible values for this element are defined by the ST_MarkerCoordinate simple type (§5.8.3.1).

Parent Elements
from (§5.8.2.11); to (§5.8.2.25)

5.8.2.28 xfrm (Graphic Frame Transform)

This element specifies a 2-D transform to be applied to a Graphic Frame.

Parent Elements
graphicFrame (§5.8.2.12)

Child Elements	Subclause
ext (Extents)	§5.1.9.3
off (Offset)	§5.1.9.4

Attributes	Description
flipH (Horizontal Flip) Namespace: .../drawingml/2006/main	Specifies a horizontal flip. When true, this attribute defines that the shape will be flipped horizontally about the center of its bounding box. [*Example*: The following illustrates the effect of a horizontal flip.  *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
flipV (Vertical Flip) Namespace: .../drawingml/2006/main	Specifies a vertical flip. When true, this attribute defines that the group will be flipped vertically about the center of its bounding box. [*Example*: The following illustrates the effect of a vertical flip.  *end example*] The possible values for this attribute are defined by the XML Schema boolean datatype.
rot (Rotation) Namespace: .../drawingml/2006/main	Specifies the rotation of the Graphic Frame. The units for which this attribute is specified in reside within the simple type definition referenced below. The possible values for this attribute are defined by the ST_Angle simple type (§5.1.12.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Transform2D">
   <sequence>
      <element name="off" type="CT_Point2D" minOccurs="0" maxOccurs="1"/>
      <element name="ext" type="CT_PositiveSize2D" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="rot" type="ST_Angle" use="optional" default="0"/>
   <attribute name="flipH" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="flipV" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.8.2.29 y (Relative Y Coordinate)

This element specifies the relative y coordinate that is used to define the percentage-based vertical position for a shape within a chart drawing object. The coordinate boundaries are specified within the corresponding simple type listed below.

The possible values for this element are defined by the ST_MarkerCoordinate simple type (§5.8.3.1).

Parent Elements
from (§5.8.2.11); to (§5.8.2.25)

5.8.3 Simple Types

This is the complete list of simple types in the http://schemas.openxmlformats.org/drawingml/2006/chartDrawing namespace.

5.8.3.1 ST_MarkerCoordinate (Chart Marker Coordinate Value)

This simple type specifies the chart marker coordinate value. It is to be represented as a fractional position between 0.0 and 1.0 of the chart width or height with 0.0 being the left or top edge.

This simple type's contents are a restriction of the XML Schema double datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 0.0.
- This simple type has a maximum value of less than or equal to 1.0.

Referenced By
x (§5.8.2.27); y (§5.8.2.29)

5.9 DrawingML - Diagrams

A DrawingML diagram allows the definition of diagrams using DrawingML objects and constructs. This namespace defines the contents of a DrawingML diagram.

5.9.1 Table of Contents

This subclause is informative.

End of informative text.

5.9.2 Diagram Definition

This section specifies the elements which define the layout and hierarchy of a diagram based on its constituent nodes and connections.

5.9.2.1 adj (Shape Adjust)

Shape adjust value. These can be used to modify the adjust handles supported on various auto shapes. It is only possible to set the initial value, not to modify it using constraints and rules.

[*Example*: Consider the following example of the adj element in a DrawingML diagram:

```
<adjLst>
    <adj idx="2" val=".35" />
</adjLst>
```

In this example we have a single adjust handle being modified by setting its value to `0.35`. *end example*]

Parent Elements
adjLst (§5.9.2.2)

Attributes	Description
idx (Adjust Handle Index)	Adjust value index. Different shapes support different adjust handles. The possible values for this attribute are defined by the ST_Index1 simple type (§5.9.7.38).
val (Value)	An absolute value. The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Adj">
   <attribute name="idx" type="ST_Index1" use="required"/>
   <attribute name="val" type="xsd:double" use="required"/>
</complexType>
```

5.9.2.2 adjLst (Shape Adjust List)

This element is simply a list of shape adjusts.

[*Example*: Consider the following example of the adjLst element in a DrawingML diagram:

```
<adjLst>
   <adj idx="1" val="1.35" />
    <adj idx="2" val=".35" />
</adjLst>
```

In this example we have a two adjust handle being modified for the containing shape. *end example*]

Parent Elements
shape (§5.9.2.27)

Child Elements	Subclause
adj (Shape Adjust)	§5.9.2.1

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AdjLst">
   <sequence>
      <element name="adj" type="CT_Adj" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.9.2.3 alg (Algorithm)

The algorithm used by the containing layout node. The algorithm defines the behavior of the layout node along with the behavior and layout of the nested layout nodes.

[*Example*: Consider the following example of alg being used in a DrawingML diagram:

```
<layoutNode name="arrow">
    <varLst/>
    <alg type="tx" />
    <shape type="upArrow">
          <adjLst>
                <adj idx="2" val=".35" />
          </adjLst>
    </shape>
    <presOf axis="desOrSelf" ptType="node" />
    <ruleLst/>
</layoutNode>
```

In this example, the tx algorithm is being used to layout text within the containing layout node. *end example*]

Parent Elements
else (§5.9.2.12); forEach (§5.9.2.14); if (§5.9.2.15); layoutNode (§5.9.2.19)

Child Elements	Subclause
extLst (Extension List)	§5.9.2.13
param (Parameter)	§5.9.2.20

Attributes	Description
rev (Revision Number)	The revision number of an algorithm. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
type (Algorithm Type)	Specifies the algorithm type. The possible values for this attribute are defined by the ST_AlgorithmType simple type (§5.9.7.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Algorithm">
   <sequence>
      <element name="param" type="CT_Parameter" minOccurs="0" maxOccurs="unbounded"/>
      <element name="extLst" type="a:CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="type" type="ST_AlgorithmType" use="required"/>
   <attribute name="rev" type="xsd:unsignedInt" use="optional" default="0"/>
</complexType>
```

5.9.2.4 cat (Category)

This element specifies a category in the user interface where this layout definition will display to the user.

[*Example*: Consider the following example of a cat in a DrawingML diagram:

```
<catLst>
    <cat type="relationship" pri="19000" />
</catLst>
```

In this example we define a single category called `relationship` which has a priority of `19000`. *end example*]

Parent Elements
catLst (§5.9.2.5)

Attributes	Description
pri (Priority)	The priority within the category for this diagram determines the order in which it will display in the user interface. Lower numbers are displayed at the beginning of the list. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
type (Category Type)	Specifies the category type associated with the element. The possible values for this attribute are defined by the XML Schema anyURI datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Category">
   <attribute name="type" type="xsd:anyURI" use="required"/>
   <attribute name="pri" type="xsd:unsignedInt" use="required"/>
</complexType>
```

5.9.2.5 catLst (Category List)

This element is simply a list of cat elements.

[*Example*: Consider the following example of a catLst element in a DrawingML diagram:

```
<catLst>
    <cat type="list" pri="18000" />
    <cat type="relationship" pri="19000" />
</catLst>
```

In this example we define two different categories which are to be displayed in the user interface. *end example*]

Parent Elements
layoutDef (§5.9.2.16); layoutDefHdr (§5.9.2.17)

Child Elements	Subclause
cat (Category)	§5.9.2.4

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Categories">
   <sequence>
      <element name="cat" type="CT_Category" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.9.2.6 choose (Choose Element)

The choose element wraps if/else blocks into a choose block.

[*Example*: Consider the following example of a choose element in a DrawingML diagram:

```
<choose name="Name1">
    <if name="Name2" func="var" arg="dir" op="equ" val="norm">
          <alg type="snake">
                <param type="grDir" val="tL"/>
                <param type="flowDir" val="row"/>
                <param type="contDir" val="sameDir"/>
                <param type="off" val="ctr"/>
          </alg>
    </if>
    <else name="Name3">
          <alg type="snake">
                <param type="grDir" val="tR"/>
                <param type="flowDir" val="row"/>
                <param type="contDir" val="sameDir"/>
                <param type="off" val="ctr"/>
          </alg>
    </else>
</choose>
```

In this example, a choose element is used to define two different sets of parameters associated with a snake algorithm depending upon the direction in which the user wants the algorithm to flow (RTL or LTR*). end example*]

Parent Elements
else (§5.9.2.12); forEach (§5.9.2.14); if (§5.9.2.15); layoutNode (§5.9.2.19)

Child Elements	Subclause
else (Else)	§5.9.2.12
if (If)	§5.9.2.15

Attributes	Description

name (Name)	A unique name associated with the choose statement.
	
	[*Example*: Consider the following example of a choose element in a DrawingML diagram:
	
	`<choose name="Name1">`
	
	`...`

Attributes	Description
	</choose> In this example, the choose element is named Name1. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Choose">
   <sequence>
      <element name="if" type="CT_When" maxOccurs="unbounded"/>
      <element name="else" type="CT_Otherwise" minOccurs="0"/>
   </sequence>
   <attribute name="name" type="xsd:string" use="optional" default=""/>
</complexType>
```

5.9.2.7 clrData (Color Transform Sample Data)

This element defines the sample data that is to be used in the user interface controls regarding displaying color transforms for a given diagram. This sample data predefines a data model to be combined with a layout definition in order to create a diagram which a color transform can be applied and displayed to the user as an example of the color transform.

[*Example*: Consider the following example of a clrData element in a DrawingML diagram:

```
<clrData>
    <dataModel>
          <ptLst>
                <pt modelId="0" type="doc"/>
                <pt modelId="1"/>
                <pt modelId="2"/>
                <pt modelId="3"/>
                <pt modelId="4"/>
                <pt modelId="5"/>
                <pt modelId="6"/>
          </ptLst>
          <cxnLst>
                <cxn modelId="7" srcId="0" destId="1" srcOrd="0" destOrd="0"/>
                <cxn modelId="8" srcId="0" destId="2" srcOrd="1" destOrd="0"/>
                <cxn modelId="9" srcId="0" destId="3" srcOrd="2" destOrd="0"/>
                <cxn modelId="10" srcId="0" destId="4" srcOrd="3" destOrd="0"/>
                <cxn modelId="11" srcId="0" destId="5" srcOrd="4" destOrd="0"/>
                <cxn modelId="12" srcId="0" destId="6" srcOrd="5" destOrd="0"/>
          </cxnLst>
          <bg/>
```

```
            <whole/>
        </dataModel>
    </clrData>
```

In this example we define 6 points which all connect back to a seventh document type point. *end example*]

Parent Elements
layoutDef (§5.9.2.16)

Child Elements	Subclause
dataModel (Data Model)	§5.9.2.10

Attributes	Description
useDef (Use Default)	If the value of this attribute is `true`, the data model defined in the clrData element is ignored and a default data model is used instead. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SampleData">
   <sequence>
      <element name="dataModel" type="CT_DataModel" minOccurs="0"/>
   </sequence>
   <attribute name="useDef" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.9.2.8 constr (Constraint)

This element is used to specify size, position of nodes, text values, and layout dependencies between nodes in a layout definition.

[*Example*: Consider the following example of a constraint list which contains some example constraints being defined and applied to layout nodes in the layout definition:

```
<constrLst>
    <constr type="w" for="ch" forName="node1" refType="w" refForName=""/>
    <constr type="h" for="ch" forName="node1" refType="w" refFor="ch"
refForName="node1" op="equ" fact="0.6"/>
    <constr type="w" for="ch" forName="transition1" refType="w" refFor="ch"
refForName="node1" op="equ" fact="0.1"/>
    <constr type="primFontSz" for="ch" forName="node1" refForName="" op="equ"
val="100"/>
</constrLst>
```

In this example we can see constraints being defined for the width and height along with the primary font size for a layout node referenced by node1. The width for a transition is also specified. *end example*]

Parent Elements
constrLst (§5.9.2.9)

Child Elements	Subclause
extLst (Extension List)	§5.9.2.13

Attributes	Description
fact (Factor)	Factor used in a reference constraint or a rule in order to modify a referenced value by the factor defined. [*Example*: Consider the following example of fact in use in a DrawingML diagram: `<constr type="w" for="ch" forName="transition1" refType="w" refFor="ch" refForName="node1" op="equ" fact="0.1"/>` In this example, the width for transition1 is being defined as one-tenth the width of node1. *end example*] The possible values for this attribute are defined by the XML Schema double datatype.
for (For)	Specifies the axis of layout nodes to apply a constraint or rule to. [*Example*: Consider the following example of for in use in a DrawingML diagram: `<constr type="w" for="ch" forName="transition1" refType="w" refFor="ch" refForName="node1" op="equ" fact="0.1"/>` In this example, the for attribute is specifying that node1 is a child node to the current layout node. *end example*] The possible values for this attribute are defined by the ST_ConstraintRelationship simple type (§5.9.7.20).
forName (For Name)	Specifies the name of the layout node to apply a constraint or rule to. [*Example*: Consider the following example of forName in use in a DrawingML diagram: `<constr type="w" for="ch" forName="transition1" refType="w" refFor="ch" refForName="node1" op="equ" fact="0.1"/>` In this example, forName is specifying the layout node named transition1 for its reference. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the XML Schema string datatype.
op (Operator)	The operator constraint used to evaluate the condition. [*Example*: Consider the following example of op in use in a DrawingML diagram: `<constr type="w" for="ch" forName="transition1" refType="w" refFor="ch" refForName="node1" op="equ" fact="0.1"/>` In this example, op is specifying an equality defined between the two referencing values. *end example*] The possible values for this attribute are defined by the ST_BoolOperator simple type (§5.9.7.10).
ptType (Data Point Type)	Specifies the type of data point to select. The possible values for this attribute are defined by the ST_ElementType simple type (§5.9.7.25).
refFor (Reference For)	The for value of the referenced constraint. [*Example*: Consider the following example of refFor in use in a DrawingML diagram: `<constr type="w" for="ch" forName="transition1" refType="w" refFor="ch" refForName="node1" op="equ" fact="0.1"/>` In this example, refFor is specifying the reference constraint is a child of the current layout node. *end example*] The possible values for this attribute are defined by the ST_ConstraintRelationship simple type (§5.9.7.20).
refForName (Reference For Name)	The name of the layout node referenced by a reference constraint. [*Example*: Consider the following example of refForName in use in a DrawingML diagram: `<constr type="w" for="ch" forName="transition1" refType="w" refFor="ch" refForName="node1" op="equ" fact="0.1"/>` In this example, refForName is specifying the layout node named node1 for its reference. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
refPtType (Reference Point Type)	The point type used int he referenced constraint.

Attributes	Description
	The possible values for this attribute are defined by the ST_ElementType simple type (§5.9.7.25).
refType (Reference Type)	Specifies the type of a reference constraint. [*Example*: Consider the following example of refType in use in a DrawingML diagram: `<constr type="w" for="ch" forName="transition1" refType="w" refFor="ch" refForName="node1" op="equ" fact="0.1"/>` In this example, refType is specifying referencing the width attribute of `node1`. *end example*] The possible values for this attribute are defined by the ST_ConstraintType simple type (§5.9.7.21).
type (Constraint Type)	Specifies the constraint to apply to this layout node. [*Example*: Consider the following example of type in use in a DrawingML diagram: `<constr type="w" for="ch" forName="transition1" refType="w" refFor="ch" refForName="node1" op="equ" fact="0.1"/>` In this example, type is specifying the width attribute of `transition1`. *end example*] The possible values for this attribute are defined by the ST_ConstraintType simple type (§5.9.7.21).
val (Value)	Specifies an absolute value instead of reference another constraint. [*Example*: Consider the following example of forName in use in a DrawingML diagram: `<constr type="w" for="ch" forName="transition1" val="10"/>` In this example, val is specifying the absolute value of the width of `transition1`. *end example*] The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Constraint">
   <sequence>
      <element name="extLst" type="a:CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attributeGroup ref="AG_ConstraintAttributes"/>
   <attributeGroup ref="AG_ConstraintRefAttributes"/>
   <attribute name="op" type="ST_BoolOperator" use="optional" default="none"/>
   <attribute name="val" type="xsd:double" use="optional" default="0"/>
   <attribute name="fact" type="xsd:double" use="optional" default="1"/>
</complexType>
```

5.9.2.9 constrLst (Constraint List)

This element is simply a list of constraints.

[*Example*: Consider the following example of a constraint list which contains some example constraints which are being defined and applied to layout nodes in the layout definition:

```
<constrLst>
   <constr type="w" for="ch" forName="node1" refType="w" refForName=""/>
   <constr type="h" for="ch" forName="node1" refType="w" refFor="ch"
refForName="node1" op="equ" fact="0.6"/>
   <constr type="w" for="ch" forName="transition1" refType="w" refFor="ch"
refForName="node1" op="equ" fact="0.1"/>
   <constr type="primFontSz" for="ch" forName="node1" refForName="" op="equ"
val="100"/>
</constrLst>
```

In this example we can see constraints being defined for the width and height along with the primary font size for a layout node referenced by `node1`. The width for a transition is also specified. *end example*]

Parent Elements
else (§5.9.2.12); forEach (§5.9.2.14); if (§5.9.2.15); layoutNode (§5.9.2.19)

Child Elements	Subclause
constr (Constraint)	§5.9.2.8

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Constraints">
   <sequence>
      <element name="constr" type="CT_Constraint" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.9.2.10 dataModel (Data Model)

The data for this instance of the diagram. Either a sample data model, or the data the user has entered.

Parent Elements
Root element of DrawingML Diagram Data partclrData (§5.9.2.7); sampData (§5.9.2.26); styleData (§5.9.2.29)

Child Elements	Subclause
bg (Background Formatting)	§5.9.3.1
cxnLst (Connection List)	§5.9.3.3
extLst (Extension List)	§5.9.2.13
ptLst (Point List)	§5.9.3.6
whole (Whole E2O Formatting)	§5.9.3.9

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DataModel">
   <sequence>
      <element name="ptLst" type="CT_PtList"/>
      <element name="cxnLst" type="CT_CxnList" minOccurs="0" maxOccurs="1"/>
      <element name="bg" type="a:CT_BackgroundFormatting" minOccurs="0"/>
      <element name="whole" type="a:CT_WholeE2oFormatting" minOccurs="0"/>
      <element name="extLst" type="a:CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.9.2.11 desc (Description)

This element holds a description for a layout definition. The description can be used to describe the qualities associated with a particular layout definition.

Parent Elements
layoutDef (§5.9.2.16); layoutDefHdr (§5.9.2.17)

Attributes	Description
lang (Language)	The natural language of the title or description of this layout definition. The possible values for this attribute are defined by the XML Schema string datatype.
val (Value)	The string which is used as the description of the layout definition. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Description">
   <attribute name="lang" type="xsd:string" use="optional" default=""/>
   <attribute name="val" type="xsd:string" use="required"/>
</complexType>
```

5.9.2.12 else (Else)

This element is similar to an else statement in a programming language in that it wraps elements which are to be used when the if conditionals are not true.

[*Example*: Consider the following example of an else element in a DrawingML diagram within the context of a choose statement:

```
<choose name="Name1">
    <if name="Name2" func="var" arg="dir" op="equ" val="norm">
          <alg type="snake">
                <param type="grDir" val="tL"/>
                <param type="flowDir" val="row"/>
                <param type="contDir" val="sameDir"/>
                <param type="off" val="ctr"/>
          </alg>
    </if>
    <else name="Name3">
          <alg type="snake">
                <param type="grDir" val="tR"/>
                <param type="flowDir" val="row"/>
                <param type="contDir" val="sameDir"/>
                <param type="off" val="ctr"/>
          </alg>
    </else>
</choose>
```

In this example, a else element is used to define a set of parameters associated with the snake algorithm when the diagram is reversed. *end example*]

Parent Elements
choose (§5.9.2.6)

Child Elements	Subclause
alg (Algorithm)	§5.9.2.3
choose (Choose Element)	§5.9.2.6

Child Elements	Subclause
constrLst (Constraint List)	§5.9.2.9
extLst (Extension List)	§5.9.2.13
forEach (For Each)	§5.9.2.14
layoutNode (Layout Node)	§5.9.2.19
presOf (Presentation Of)	§5.9.2.21
ruleLst (Rule List)	§5.9.2.25
shape (Shape)	§5.9.2.27

Attributes	Description

name (Name)	A unique name associated with the choose statement.
	
	[*Example*: Consider the following example of a else element in a DrawingML diagram:
	
	`<else name="Name1">`
	
	`...`
	
	`</else>`
	
	In this example, the else element is named `Name1`. *end example*].
	
	The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Otherwise">
   <choice minOccurs="0" maxOccurs="unbounded">
      <element name="alg" type="CT_Algorithm" minOccurs="0" maxOccurs="1"/>
      <element name="shape" type="CT_Shape" minOccurs="0" maxOccurs="1"/>
      <element name="presOf" type="CT_PresentationOf" minOccurs="0" maxOccurs="1"/>
      <element name="constrLst" type="CT_Constraints" minOccurs="0" maxOccurs="1"/>
      <element name="ruleLst" type="CT_Rules" minOccurs="0" maxOccurs="1"/>
      <element name="forEach" type="CT_ForEach"/>
      <element name="layoutNode" type="CT_LayoutNode"/>
      <element name="choose" type="CT_Choose"/>
      <element name="extLst" type="a:CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </choice>
   <attribute name="name" type="xsd:string" use="optional" default=""/>
</complexType>
```

5.9.2.13 extLst (Extension List)

This element specifies an extension list, within which all future extensions will be defined within ext elements.

The extension list along with corresponding future extensions is used to extend the storage capabilities of the DrawingML framework. This allows for various new types of data to be stored natively within the existing diagram syntax.

Parent Elements
alg (§5.9.2.3); colorsDef (§5.9.4.3); colorsDefHdr (§5.9.4.4); constr (§5.9.2.8); cxn (§5.9.3.2); dataModel (§5.9.2.10); else (§5.9.2.12); forEach (§5.9.2.14); if (§5.9.2.15); layoutDef (§5.9.2.16); layoutDefHdr (§5.9.2.17); layoutNode (§5.9.2.19); presOf (§5.9.2.21); pt (§5.9.3.5); rule (§5.9.2.24); shape (§5.9.2.27); styleDef (§5.9.5.7); styleDefHdr (§5.9.5.8); styleLbl (§5.9.5.10); styleLbl (§5.9.4.10)

Child Elements	Subclause
ext (Extension)	§5.1.2.1.14

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OfficeArtExtensionList">
   <sequence>
      <group ref="EG_OfficeArtExtensionList" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.9.2.14 forEach (For Each)

A looping structure, similar to a `for` loop in a programming language, which defines what data model points will use this layout node.

[*Example*: Consider the following example of a forEach being used within a DrawingML diagram:

```
<forEach name="Name5" ref="" axis="ch" ptType="node">
    <layoutNode name="node1" styleLbl="" moveWith="">
          <alg type="sp"/>
          <shape
xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
r:blip="">
                <adjLst/>
          </shape>
          <constrLst/>
    </layoutNode>
</forEach>
```

In this example, the forEach element will create a layout node, referenced by the name `node1`, for every associated data model point in the diagram. In this particular instance the forEach will create the layout node for every child of the current point node. *end example*]

Parent Elements

Parent Elements
else (§5.9.2.12); forEach (§5.9.2.14); if (§5.9.2.15); layoutNode (§5.9.2.19)

Child Elements	Subclause
alg (Algorithm)	§5.9.2.3
choose (Choose Element)	§5.9.2.6
constrLst (Constraint List)	§5.9.2.9
extLst (Extension List)	§5.9.2.13
forEach (For Each)	§5.9.2.14
layoutNode (Layout Node)	§5.9.2.19
presOf (Presentation Of)	§5.9.2.21
ruleLst (Rule List)	§5.9.2.25
shape (Shape)	§5.9.2.27

Attributes	Description
axis (Axis)	Specifies the axis on which to select data from the data model. [*Example*: For example, `axis="ch"` will select children of the current point node and `axis="des"` will select all descendants. *end example*] The possible values for this attribute are defined by the ST_AxisTypes simple type (§5.9.7.7).

cnt (Count)	Specifies the count of items to use in a data set.
	
	[*Example*: Consider the following example of a forEach in a DrawingML diagram:
	
	`<forEach name="Name5" ref="" axis="ch" ptType="node" cnt="2">`
	
	`...`
	
	`</forEach>`
	
	In this example, up to two children will be obtained through this forEach. *end example*]
	
	The possible values for this attribute are defined by the ST_UnsignedInts simple type (§5.9.7.63).

hideLastTrans (Hide Last Transition)	In algorithms that support transitions, this attribute specifies that the last transition will not be rendered. This allows for diagrams that start and end with a node. The possible values for this attribute are defined by the ST_Booleans simple type

Attributes	Description
	(§5.9.7.9).
name (Name)	A unique identifier for the layout node. The possible values for this attribute are defined by the XML Schema string datatype.

ptType (Data Point Type)	Specifies the type of data point to select.
	
	[*Example*: Consider the following example of a forEach in a DrawingML diagram:
	
	`<forEach name="Name5" ref="" axis="ch" ptType="node" cnt="2">`
	
	`...`
	
	`</forEach>`
	
	In this example, the forEach will select all node type points in the set. *end example*]
	
	The possible values for this attribute are defined by the ST_ElementTypes simple type (§5.9.7.26).

ref (Reference)	When used on a for-each element, causes the specified for-each element to be used instead. The possible values for this attribute are defined by the XML Schema string datatype.
st (Start)	Specifies where to start in a data set. [*Example*: Consider the following example of a forEach in a DrawingML diagram: `<presOf axis="desOrSelf" ptType="node" st="2"/>` In this example, the second element in the set will be the first point returned. *end example*] The possible values for this attribute are defined by the ST_Ints simple type (§5.9.7.39).
step (Step)	Specifies the step to use in a data set. A step with a value of 2 will return every other item in the set. The possible values for this attribute are defined by the ST_Ints simple type (§5.9.7.39).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ForEach">
   <choice minOccurs="0" maxOccurs="unbounded">
      <element name="alg" type="CT_Algorithm" minOccurs="0" maxOccurs="1"/>
      <element name="shape" type="CT_Shape" minOccurs="0" maxOccurs="1"/>
      <element name="presOf" type="CT_PresentationOf" minOccurs="0" maxOccurs="1"/>
      <element name="constrLst" type="CT_Constraints" minOccurs="0" maxOccurs="1"/>
      <element name="ruleLst" type="CT_Rules" minOccurs="0" maxOccurs="1"/>
      <element name="forEach" type="CT_ForEach"/>
      <element name="layoutNode" type="CT_LayoutNode"/>
      <element name="choose" type="CT_Choose"/>
      <element name="extLst" type="a:CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </choice>
   <attribute name="name" type="xsd:string" use="optional" default=""/>
   <attribute name="ref" type="xsd:string" use="optional" default=""/>
   <attributeGroup ref="AG_IteratorAttributes"/>
</complexType>
```

5.9.2.15 if (If)

Like an if statement in a programming language, wraps elements which are to be used under the conditions defined by its attributes.

[*Example*: Consider the following example of an if element in a DrawingML diagram within the context of a choose statement:

```
<choose name="Name1">
    <if name="Name2" func="var" arg="dir" op="equ" val="norm">
          <alg type="snake">
                <param type="grDir" val="tL"/>
                <param type="flowDir" val="row"/>
                <param type="contDir" val="sameDir"/>
                <param type="off" val="ctr"/>
          </alg>
    </if>
    <else name="Name3">
          <alg type="snake">
                <param type="grDir" val="tR"/>
                <param type="flowDir" val="row"/>
                <param type="contDir" val="sameDir"/>
                <param type="off" val="ctr"/>
          </alg>
    </else>
</choose>
```

In this example, a if element is used to define a set of parameters associated with the snake algorithm when the diagram is in the normal direction. *end example*]

Parent Elements
choose (§5.9.2.6)

Child Elements	Subclause
alg (Algorithm)	§5.9.2.3
choose (Choose Element)	§5.9.2.6
constrLst (Constraint List)	§5.9.2.9
extLst (Extension List)	§5.9.2.13
forEach (For Each)	§5.9.2.14
layoutNode (Layout Node)	§5.9.2.19
presOf (Presentation Of)	§5.9.2.21
ruleLst (Rule List)	§5.9.2.25
shape (Shape)	§5.9.2.27

Attributes	Description
arg (Argument)	Specifies the variable to use as part of the test in an if element. Ignored unless the function attribute is set to "var". The possible values for this attribute are defined by the ST_FunctionArgument simple type (§5.9.7.29).
axis (Axis)	Specifies the axis on which to select data from the data model. [*Example*: For example, `axis="ch"` will select children of the current point node and `axis="des"` will select all descendants. *end example*] The possible values for this attribute are defined by the ST_AxisTypes simple type (§5.9.7.7).

cnt (Count)	Specifies the count of items to use in a data set.
	
	[*Example*: Consider the following example of a forEach in a DrawingML diagram:
	
	`<forEach name="Name5" ref="" axis="ch" ptType="node" cnt="2">`
	
	`...`
	
	`</forEach>`
	
	In this example, up to two children will be obtained through this forEach. *end example*]
	
	The possible values for this attribute are defined by the ST_UnsignedInts simple type

Attributes	Description
	(§5.9.7.63).

func (Function)	The function used to evaluate the if condition.
	
	[*Example*: Consider the following example of `func` being used in DrawingML:
	
	`<if name="Name2" func="var" arg="dir" op="equ" val="norm">`
	`...`
	`</if>`
	
	In this example, `func` is set to `var`. *end example*]
	
	The possible values for this attribute are defined by the ST_FunctionType simple type (§5.9.7.31).

hideLastTrans (Hide Last Transition)	In algorithms that support transitions, this attribute specifies that the last transition will not be rendered. This allows for diagrams that start and end with a node. The possible values for this attribute are defined by the ST_Booleans simple type (§5.9.7.9).

name (Name)	A unique identifier for the layout node.
	
	The function used to evaluate the if condition.
	
	[*Example*: Consider the following example of `name` being used in DrawingML:
	
	`<if name="Name2" func="var" arg="dir" op="equ" val="norm">`
	`...`
	`</if>`
	
	In this example, the `name` attribute is set to `Name2`. *end example*]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

op (Operator)	The operator used to evaluate the condition.
	
	[*Example*: Consider the following example of op being used in DrawingML:
	
	`<if name="Name2" func="var" arg="dir" op="equ" val="norm">`
	`...`
	`</if>`
	
	In this example, op is being used to test the equality of the argument and value. *end example*]
	
	The possible values for this attribute are defined by the ST_FunctionOperator simple type (§5.9.7.30).

ptType (Data Point	Specifies the type of data point to select.

Attributes	Description

Type)	[*Example*: Consider the following example of a forEach in a DrawingML diagram:
	
	`<forEach name="Name5" ref="" axis="ch" ptType="node" cnt="2">`
	
	`...`
	
	`</forEach>`
	
	In this example, the forEach will select all node type points in the set. *end example*]
	
	The possible values for this attribute are defined by the ST_ElementTypes simple type (§5.9.7.26).

st (Start)	Specifies where to start in a data set. [*Example*: Consider the following example of a forEach in a DrawingML diagram: `<presOf axis="desOrSelf" ptType="node" st="2"/>` In this example, the second element in the set will be the first point returned. *end example*] The possible values for this attribute are defined by the ST_Ints simple type (§5.9.7.39).
step (Step)	Specifies the step to use in a data set. A step with a value of 2 will return every other item in the set. The possible values for this attribute are defined by the ST_Ints simple type (§5.9.7.39).
val (Value)	An absolute value. The possible values for this attribute are defined by the ST_FunctionValue simple type (§5.9.7.32).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_When">
   <choice minOccurs="0" maxOccurs="unbounded">
      <element name="alg" type="CT_Algorithm" minOccurs="0" maxOccurs="1"/>
      <element name="shape" type="CT_Shape" minOccurs="0" maxOccurs="1"/>
      <element name="presOf" type="CT_PresentationOf" minOccurs="0" maxOccurs="1"/>
      <element name="constrLst" type="CT_Constraints" minOccurs="0" maxOccurs="1"/>
      <element name="ruleLst" type="CT_Rules" minOccurs="0" maxOccurs="1"/>
      <element name="forEach" type="CT_ForEach"/>
      <element name="layoutNode" type="CT_LayoutNode"/>
      <element name="choose" type="CT_Choose"/>
      <element name="extLst" type="a:CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </choice>
   <attribute name="name" type="xsd:string" use="optional" default=""/>
   <attributeGroup ref="AG_IteratorAttributes"/>
   <attribute name="func" type="ST_FunctionType" use="required"/>
   <attribute name="arg" type="ST_FunctionArgument" use="optional" default="none"/>
   <attribute name="op" type="ST_FunctionOperator" use="required"/>
   <attribute name="val" type="ST_FunctionValue" use="required"/>
</complexType>
```

5.9.2.16 layoutDef (Layout Definition)

This element is the root element for defining a layout definition. The layout definition is defined through a set of nested layout nodes. The layout definition is responsible for defining the look of a diagram.

Parent Elements
Root element of DrawingML Diagram Layout Definition part

Child Elements	Subclause
catLst (Category List)	§5.9.2.5
clrData (Color Transform Sample Data)	§5.9.2.7
desc (Description)	§5.9.2.11
extLst (Extension List)	§5.9.2.13
layoutNode (Layout Node)	§5.9.2.19
sampData (Sample Data)	§5.9.2.26
styleData (Style Data)	§5.9.2.29
title (Title)	§5.9.2.30

Attributes	Description
defStyle (Default Style)	This attribute defines a reference to a default style which is to be applied to the diagram. The possible values for this attribute are defined by the XML Schema string datatype.

Attributes	Description
minVer (Minimum Version)	Minimum product version that can support this layout definition. The possible values for this attribute are defined by the XML Schema string datatype.
uniqueId (Unique Identifier)	The unique identifier for this layout definition. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DiagramDefinition">
   <sequence>
      <element name="title" type="CT_Name" minOccurs="0" maxOccurs="unbounded"/>
      <element name="desc" type="CT_Description" minOccurs="0" maxOccurs="unbounded"/>
      <element name="catLst" type="CT_Categories" minOccurs="0"/>
      <element name="sampData" type="CT_SampleData" minOccurs="0"/>
      <element name="styleData" type="CT_SampleData" minOccurs="0"/>
      <element name="clrData" type="CT_SampleData" minOccurs="0"/>
      <element name="layoutNode" type="CT_LayoutNode"/>
      <element name="extLst" type="a:CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="uniqueId" type="xsd:string" use="optional" default=""/>
   <attribute name="minVer" type="xsd:string" use="optional"
     default="http://schemas.openxmlformats.org/drawingml/2006/diagram"/>
   <attribute name="defStyle" type="xsd:string" use="optional" default=""/>
</complexType>
```

5.9.2.17 layoutDefHdr (Layout Definition Header)

This element is the header information representing the minimum knowledge needed by an application to preload information about a layout definition. This preloading allows for the actual load of the layout definition to occur at a later time which will help with any performance concerns an application may have.

[*Example*: Consider the following example of a layoutDefHdr within a DrawingML diagram:

```
<layoutDefHdr uniqueId="urn:layout/default">
    <title val="Basic Block List" />
    <desc val="" />
    <catLst>
        <cat type="list" pri="1000" />
    </catLst>
</layoutDefHdr>
```

In this example we define a title along with a category and prioritization for the diagram referenced by the `uniqueId` of `urn:layout:default`. *end example*]

Parent Elements
layoutDefHdrLst (§5.9.2.18)

Child Elements	Subclause
catLst (Category List)	§5.9.2.5
desc (Description)	§5.9.2.11
extLst (Extension List)	§5.9.2.13
title (Title)	§5.9.2.30

Attributes	Description
defStyle (Default Style)	This attribute defines a reference to a default style which is to be applied to the diagram. The possible values for this attribute are defined by the XML Schema string datatype.
minVer (Minimum Version)	Minimum product version that can support this Diagram Layout. The possible values for this attribute are defined by the XML Schema string datatype.
resId (Resource Identifier)	Resource ID used internally. The possible values for this attribute are defined by the XML Schema int datatype.
uniqueId (Unique Identifier)	The unique identifier for this layout definition. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DiagramDefinitionHeader">
   <sequence>
      <element name="title" type="CT_Name" minOccurs="1" maxOccurs="unbounded"/>
      <element name="desc" type="CT_Description" minOccurs="1" maxOccurs="unbounded"/>
      <element name="catLst" type="CT_Categories" minOccurs="0"/>
      <element name="extLst" type="a:CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="uniqueId" type="xsd:string" use="required"/>
   <attribute name="minVer" type="xsd:string" use="optional"
     default="http://schemas.openxmlformats.org/drawingml/2006/diagram"/>
   <attribute name="defStyle" type="xsd:string" use="optional" default=""/>
   <attribute name="resId" type="xsd:int" use="optional" default="0"/>
</complexType>
```

5.9.2.18 layoutDefHdrLst (Diagram Layout Header List)

This element is simply a list of layout definition headers. This list of headers is used internally as a way to group all of the layout definition headers together into a single structure.

Child Elements	Subclause
layoutDefHdr (Layout Definition Header)	§5.9.2.17

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DiagramDefinitionHeaderLst">
   <sequence>
      <element name="layoutDefHdr" type="CT_DiagramDefinitionHeader" minOccurs="0"
        maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.9.2.19 layoutNode (Layout Node)

The layout node is the basic building block of diagrams. The layout node is responsible for defining how shapes are arranged in a diagram and how the data maps to a particular shape in a diagram.

[*Example*: Consider the following example of a basic layout node defined in a DrawingML diagram:

```
<layoutNode name="node">
    <varLst>
          <bulletEnabled val="1"/>
    </varLst>
    <presOf axis="desOrSelf" ptType="node"/>
    <alg type="tx"/>
    <shape type="rect"
xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
r:blip="">
          <adjLst/>
    </shape>
    <constrLst/>
    <ruleLst>
          <rule type="primFontSz" forName="" val="2" fact="NaN" max="NaN"/>
    </ruleLst>
</layoutNode>
```

In this example we define a layout node which holds text and is a rectangle. *end example*]

Parent Elements
else (§5.9.2.12); forEach (§5.9.2.14); if (§5.9.2.15); layoutDef (§5.9.2.16); layoutNode (§5.9.2.19)

Child Elements	Subclause
alg (Algorithm)	§5.9.2.3
choose (Choose Element)	§5.9.2.6
constrLst (Constraint List)	§5.9.2.9
extLst (Extension List)	§5.9.2.13
forEach (For Each)	§5.9.2.14

Child Elements	Subclause
layoutNode (Layout Node)	§5.9.2.19
presOf (Presentation Of)	§5.9.2.21
ruleLst (Rule List)	§5.9.2.25
shape (Shape)	§5.9.2.27
varLst (Variable List)	§5.9.2.31

Attributes	Description
chOrder (Child Order)	Specifes the ordering of the child layout nodes for a given layout node. The possible values for this attribute are defined by the ST_ChildOrderType simple type (§5.9.7.15).
moveWith (Move With)	Reference to another layout node that this layout node moves with. The possible values for this attribute are defined by the XML Schema string datatype.
name (Name)	A unique identifier for the layout node. The possible values for this attribute are defined by the XML Schema string datatype.
styleLbl (Style Label)	Specify which formatting option from a style or color variation should be applied to this layout node. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LayoutNode">
   <choice minOccurs="0" maxOccurs="unbounded">
      <element name="alg" type="CT_Algorithm" minOccurs="0" maxOccurs="1"/>
      <element name="shape" type="CT_Shape" minOccurs="0" maxOccurs="1"/>
      <element name="presOf" type="CT_PresentationOf" minOccurs="0" maxOccurs="1"/>
      <element name="constrLst" type="CT_Constraints" minOccurs="0" maxOccurs="1"/>
      <element name="ruleLst" type="CT_Rules" minOccurs="0" maxOccurs="1"/>
      <element name="varLst" type="CT_LayoutVariablePropertySet" minOccurs="0" maxOccurs="1"/>
      <element name="forEach" type="CT_ForEach"/>
      <element name="layoutNode" type="CT_LayoutNode"/>
      <element name="choose" type="CT_Choose"/>
      <element name="extLst" type="a:CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </choice>
   <attribute name="name" type="xsd:string" use="optional" default=""/>
   <attribute name="styleLbl" type="xsd:string" use="optional" default=""/>
   <attribute name="chOrder" type="ST_ChildOrderType" use="optional" default="b"/>
   <attribute name="moveWith" type="xsd:string" use="optional" default=""/>
</complexType>
```

5.9.2.20 param (Parameter)

The parameter element modifies the default behavior of an algorithm.

[*Example*: Consider the following example of a param being used in a DrawingML diagram:

```
<alg type="snake">
    <param type="grDir" val="tL"/>
    <param type="flowDir" val="row"/>
    <param type="contDir" val="sameDir"/>
    <param type="off" val="ctr"/>
</alg>
```

In this example we see the snake algorithm being utilized and four parameters being set which are associated with the snake algorithm and modify its behavior. *end example*]

Parent Elements
alg (§5.9.2.3)

Attributes	Description
type (Parameter Type)	Specifies the parameter which is being modified. The possible values for this attribute are defined by the ST_ParameterId simple type (§5.9.7.48).
val (Value)	Specifies the actual value to be given to the type defined. The possible values for this attribute are defined by the ST_ParameterVal simple type (§5.9.7.49).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Parameter">
   <attribute name="type" type="ST_ParameterId" use="required"/>
   <attribute name="val" type="ST_ParameterVal" use="required"/>
</complexType>
```

5.9.2.21 presOf (Presentation Of)

This element specifies a particular data model point which is to be mapped to the containing layout node. This attribute is responsible for defining the mapping of data to the layout nodes in a diagram.

[*Example*: Consider the following example of presOf in use within a DrawingML diagram:

```
<presOf axis="desOrSelf" ptType="node"/>
```

In this example the presOf element is mapping to a particular data model point. *end example*]

Parent Elements
else (§5.9.2.12); forEach (§5.9.2.14); if (§5.9.2.15); layoutNode (§5.9.2.19)

Child Elements	Subclause
extLst (Extension List)	§5.9.2.13

Attributes	Description
axis (Axis)	Specifies the axis on which to select data from the data model. [*Example*: For example, `axis="ch"` will select children of the current point node and `axis="des"` will select all descendants. *end example*] The possible values for this attribute are defined by the ST_AxisTypes simple type (§5.9.7.7).

cnt (Count)	Specifies the count of items to use in a data set.
	
	[*Example*: Consider the following example of a forEach in a DrawingML diagram:
	
	`<forEach name="Name5" ref="" axis="ch" ptType="node" cnt="2">`
	
	`...`
	
	`</forEach>`
	
	In this example, up to two children will be obtained through this forEach. *end example*]
	
	The possible values for this attribute are defined by the ST_UnsignedInts simple type (§5.9.7.63).

hideLastTrans (Hide Last Transition)	In algorithms that support transitions, this attribute specifies that the last transition will not be rendered. This allows for diagrams that start and end with a node. The possible values for this attribute are defined by the ST_Booleans simple type (§5.9.7.9).

ptType (Data Point Type)	Specifies the type of data point to select.
	
	[*Example*: Consider the following example of a forEach in a DrawingML diagram:
	
	`<forEach name="Name5" ref="" axis="ch" ptType="node" cnt="2">`
	
	`...`
	
	`</forEach>`

Attributes	Description
	In this example, the forEach will select all node type points in the set. *end example*] The possible values for this attribute are defined by the ST_ElementTypes simple type (§5.9.7.26).
st (Start)	Specifies where to start in a data set. [*Example*: Consider the following example of a forEach in a DrawingML diagram: `<presOf axis="desOrSelf" ptType="node" st="2"/>` In this example, the second element in the set will be the first point returned. *end example*] The possible values for this attribute are defined by the ST_Ints simple type (§5.9.7.39).
step (Step)	Specifies the step to use in a data set. A step with a value of 2 will return every other item in the set. The possible values for this attribute are defined by the ST_Ints simple type (§5.9.7.39).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PresentationOf">
   <sequence>
      <element name="extLst" type="a:CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attributeGroup ref="AG_IteratorAttributes"/>
</complexType>
```

5.9.2.22 relIds (Explicit Relationships to Diagram Parts)

This element specifies the relationship IDs used to explicitly reference each of the four constituent parts of a DrawingML diagram:

- Diagram Colors (cs attribute)
- Diagram Data (dm attribute)
- Diagram Layout Definition (lo attribute)
- Diagram Style (qs attribute)

Attributes	Description
cs (Explicit Relationship to Diagram Colors Part) Namespace: .../officeDocument	Specifies the relationship ID for the explicit relationship to the Diagram Colors part used by this diagram. This relationship shall be of type http://schemas.openxmlformats.org/officeDocument/2006/relationships/diagramColors or the document shall be considered non-conformant.

Attributes	Description
/2006/relationships	The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
dm (Explicit Relationship to Diagram Data Part) Namespace: .../officeDocument/2006/relationships	Specifies the relationship ID for the explicit relationship to the Diagram Data part used by this diagram. This relationship shall be of type http://schemas.openxmlformats.org/officeDocument/2006/relationships/diagramData or the document shall be considered non-conformant. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
lo (Explicit Relationship to Diagram Layout Definition Part) Namespace: .../officeDocument/2006/relationships	Specifies the relationship ID for the explicit relationship to the Diagram Layout Definition part used by this diagram. This relationship shall be of type http://schemas.openxmlformats.org/officeDocument/2006/relationships/diagramLayout or the document shall be considered non-conformant. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
qs (Explicit Relationship to Style Definition Part) Namespace: .../officeDocument/2006/relationships	Specifies the relationship ID for the explicit relationship to the Diagram Style part used by this diagram. This relationship shall be of type http://schemas.openxmlformats.org/officeDocument/2006/relationships/diagramQuickStyle or the document shall be considered non-conformant. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RelIds">
   <attribute ref="r:dm" use="required"/>
   <attribute ref="r:lo" use="required"/>
   <attribute ref="r:qs" use="required"/>
   <attribute ref="r:cs" use="required"/>
</complexType>
```

5.9.2.23 resizeHandles (Shape Resize Style)

This element defines the behavior when resizing shapes within a diagram. Because the size of the shape plays a large role in the overall layout of other nodes within the diagram, there are two ways resize can occur on a node.

Parent Elements

Parent Elements
presLayoutVars (§5.9.5.4); varLst (§5.9.2.31)

Attributes	Description
val (Shape Resize Style Type)	Specifies the behavior for a shape when resizing shapes within a diagram. If this attribute is not specified, the default value shall be `rel`. The possible values for this attribute are defined by the ST_ResizeHandlesStr simple type (§5.9.7.53).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ResizeHandles">
   <attribute name="val" type="ST_ResizeHandlesStr" default="rel" use="optional"/>
</complexType>
```

5.9.2.24 rule (Rule)

This element allows for a rule to be specified which changes the value of an existing constraint.

[*Example*: Consider the following example of a rule in a DrawingML diagram:

```
<ruleLst>
   <rule type="primFontSz" val="2" />
</ruleLst>
```

In this example a rule is being defined that will shrink the primary font size down to a lower limit of 2pt font when the text no longer fits correctly in the layout node. *end example*]

Parent Elements
ruleLst (§5.9.2.25)

Child Elements	Subclause
extLst (Extension List)	§5.9.2.13

Attributes	Description
fact (Factor)	Factor used in a reference constraint or a rule in order to modify a referenced value by the factor defined. [*Example*: Consider the following example of fact in use in a DrawingML diagram: `<constr type="w" for="ch" forName="transition1" refType="w"`

Attributes	Description
	refFor="ch" refForName="node1" op="equ" fact="0.1"/> In this example, the width for transition1 is being defined as one-tenth the width of node1. *end example*] The possible values for this attribute are defined by the XML Schema double datatype.
for (For)	Specifies the axis of layout nodes to apply a constraint or rule to. [*Example*: Consider the following example of for in use in a DrawingML diagram: <constr type="w" for="ch" forName="transition1" refType="w" refFor="ch" refForName="node1" op="equ" fact="0.1"/> In this example, the for attribute is specifying that node1 is a child node to the current layout node. *end example*] The possible values for this attribute are defined by the ST_ConstraintRelationship simple type (§5.9.7.20).
forName (For Name)	Specifies the name of the layout node to apply a constraint or rule to. [*Example*: Consider the following example of forName in use in a DrawingML diagram: <constr type="w" for="ch" forName="transition1" refType="w" refFor="ch" refForName="node1" op="equ" fact="0.1"/> In this example, forName is specifying the layout node named transition1 for its reference. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
max (Max Value)	Sets the maximum value for a constraint so rules can no longer increase the constraint beyond that value. The possible values for this attribute are defined by the XML Schema double datatype.
ptType (Data Point Type)	Specifies the type of data point to select. The possible values for this attribute are defined by the ST_ElementType simple type (§5.9.7.25).
type (Constraint Type)	Specifies the constraint to apply to this layout node. [*Example*: Consider the following example of type in use in a DrawingML diagram: <constr type="w" for="ch" forName="transition1" refType="w" refFor="ch" refForName="node1" op="equ" fact="0.1"/> In this example, type is specifying the width attribute of transition1. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_ConstraintType simple type (§5.9.7.21).
val (Value)	Specifies an absolute value instead of reference another constraint. [*Example*: Consider the following example of forName in use in a DrawingML diagram: `<constr type="w" for="ch" forName="transition1" val="10"/>` In this example, val is specifying the absolute value of the width of `transition1`. *end example*] The possible values for this attribute are defined by the XML Schema double datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NumericRule">
   <sequence>
      <element name="extLst" type="a:CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attributeGroup ref="AG_ConstraintAttributes"/>
   <attribute name="val" type="xsd:double" use="optional" default="NaN"/>
   <attribute name="fact" type="xsd:double" use="optional" default="NaN"/>
   <attribute name="max" type="xsd:double" use="optional" default="NaN"/>
</complexType>
```

5.9.2.25 ruleLst (Rule List)

This element is simply a list of rules.

This element allows for a rule to be specified which changes the value of an existing constraint.

[*Example*: Consider the following example of a ruleLst in a DrawingML diagram:

```
<ruleLst>
   <rule type="primFontSz" val="2" />
</ruleLst>
```

In this example a single rule is being defined in the ruleLst that will shrink the primary font size down to a lower limit of 2pt font when the text no longer fits correctly in the layout node. *end example*]

Parent Elements
else (§5.9.2.12); forEach (§5.9.2.14); if (§5.9.2.15); layoutNode (§5.9.2.19)

Child Elements	Subclause
rule (Rule)	§5.9.2.24

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Rules">
   <sequence>
      <element name="rule" type="CT_NumericRule" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.9.2.26 sampData (Sample Data)

This element defines the sample data model which is used to pre-populate a diagram with placeholder data in order for the diagram to display itself in the user interface which shows all of the available diagrams to a user.

[*Example*: Consider the following example of a sampData element within a DrawingML diagram:

```
<sampData>
    <dataModel>
          <ptLst>
                <pt modelId="0" type="doc"/>
                <pt modelId="1">
                      <prSet phldr="1"/>
                </pt>
                <pt modelId="2">
                      <prSet phldr="1"/>
                </pt>
                <pt modelId="3">
                      <prSet phldr="1"/>
                </pt>
                <pt modelId="4">
                      <prSet phldr="1"/>
                </pt>
                <pt modelId="5">
                      <prSet phldr="1"/>
                </pt>
          </ptLst>
          <cxnLst>
                <cxn modelId="6" srcId="0" destId="1" srcOrd="0" destOrd="0"/>
                <cxn modelId="7" srcId="0" destId="2" srcOrd="1" destOrd="0"/>
                <cxn modelId="8" srcId="0" destId="3" srcOrd="2" destOrd="0"/>
                <cxn modelId="9" srcId="0" destId="4" srcOrd="3" destOrd="0"/>
                <cxn modelId="10" srcId="0" destId="5" srcOrd="4" destOrd="0"/>
          </cxnLst>
          <bg/>
          <whole/>
    </dataModel>
</sampData>
```

In this example we define the sample data to consist of five nodes all attached to a document point type node. When displayed, this diagram will show five shapes in the diagram. *end example*]

Parent Elements
layoutDef (§5.9.2.16)

Child Elements	Subclause
dataModel (Data Model)	§5.9.2.10

Attributes	Description
useDef (Use Default)	If the value of this attribute is `true`, the data model defined in the clrData element is ignored and a default data model is used instead. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SampleData">
   <sequence>
      <element name="dataModel" type="CT_DataModel" minOccurs="0"/>
   </sequence>
   <attribute name="useDef" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.9.2.27 shape (Shape)

The shape displayed by the containing layout node. Not all layout nodes display shapes.

Parent Elements
else (§5.9.2.12); forEach (§5.9.2.14); if (§5.9.2.15); layoutNode (§5.9.2.19)

Child Elements	Subclause
adjLst (Shape Adjust List)	§5.9.2.2
extLst (Extension List)	§5.9.2.13

Attributes	Description
blip (Relationship to Image Part) Namespace: .../officeDocument/2006/relationshi	Specifies the relationship ID of the explicit relationship to an image which shall be used as the image for the contents of this shape. This relationship shall be of type http://schemas.openxmlformats.org/officeDocument/2006/relationships/image or

Attributes	Description
ps	the document shall be considered non-conformant. The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).
blipPhldr (Image Placeholder)	Specifies whether to use an image placeholder or not. The possible values for this attribute are defined by the XML Schema boolean datatype.
hideGeom (Hide Geometry)	When set to "true", hides the geometry of the shape. The text is still visible. The possible values for this attribute are defined by the XML Schema boolean datatype.
lkTxEntry (Prevent Text Editing)	Prevents text editing on this shape. The possible values for this attribute are defined by the XML Schema boolean datatype.
rot (Rotation)	Rotates the shape by the specified number of degrees. The possible values for this attribute are defined by the XML Schema double datatype.
type (Shape Type)	Specifies the type of shape. The possible values for this attribute are defined by the ST_LayoutShapeType simple type (§5.9.7.40).
zOrderOff (Z-Order Offset)	Offsets the shape from its default z-order stacking, which is based on the order the layout nodes appear in the XML. The possible values for this attribute are defined by the XML Schema int datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Shape">
   <sequence>
      <element name="adjLst" type="CT_AdjLst" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="a:CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="rot" type="xsd:double" use="optional" default="0"/>
   <attribute name="type" type="ST_LayoutShapeType" use="optional" default="none"/>
   <attribute ref="r:blip" use="optional"/>
   <attribute name="zOrderOff" type="xsd:int" use="optional" default="0"/>
   <attribute name="hideGeom" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="lkTxEntry" type="xsd:boolean" use="optional" default="false"/>
   <attribute name="blipPhldr" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.9.2.28 style (Shape Style)

This element specifies the style information for a shape, as defined by its DrawingML child elements.

Parent Elements

Parent Elements
prSet (§5.9.3.4); styleLbl (§5.9.5.10)

Child Elements	Subclause
effectRef (Effect Reference)	§5.1.4.2.8
fillRef (Fill Reference)	§5.1.4.2.10
fontRef (Font Reference)	§5.1.4.1.17
lnRef (Line Reference)	§5.1.4.2.19

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShapeStyle">
   <sequence>
      <element name="lnRef" type="CT_StyleMatrixReference" minOccurs="1" maxOccurs="1"/>
      <element name="fillRef" type="CT_StyleMatrixReference" minOccurs="1" maxOccurs="1"/>
      <element name="effectRef" type="CT_StyleMatrixReference" minOccurs="1" maxOccurs="1"/>
      <element name="fontRef" type="CT_FontReference" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.9.2.29 styleData (Style Data)

This element defines the style data model which is used to pre-populate a diagram with placeholder data in order for the diagram to display itself in the user interface which shows a quick style applied to the diagram.

[*Example*: Consider the following example of a styleData being used within a DrawingML diagram:

```
<styleData>
    <dataModel>
          <ptLst>
                <pt modelId="0" type="doc"/>
                <pt modelId="1"/>
                <pt modelId="2"/>
          </ptLst>
          <cxnLst>
                <cxn modelId="3" srcId="0" destId="1" srcOrd="0" destOrd="0"/>
                <cxn modelId="4" srcId="0" destId="2" srcOrd="1" destOrd="0"/>
          </cxnLst>
          <bg/>
          <whole/>
    </dataModel>
</styleData>
```

In this example we define a data model which has only two nodes which will be shown in the user interface when a layout definition is combined with this data model. *end example*]

Parent Elements
layoutDef (§5.9.2.16)

Child Elements	Subclause
dataModel (Data Model)	§5.9.2.10

Attributes	Description
useDef (Use Default)	If the value of this attribute is `true`, the data model defined in the clrData element is ignored and a default data model is used instead. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SampleData">
   <sequence>
      <element name="dataModel" type="CT_DataModel" minOccurs="0"/>
   </sequence>
   <attribute name="useDef" type="xsd:boolean" use="optional" default="false"/>
</complexType>
```

5.9.2.30 title (Title)

Title of the Diagram Layout.

Parent Elements
layoutDef (§5.9.2.16); layoutDefHdr (§5.9.2.17)

Attributes	Description
lang (Language)	Specifies the language of the title or description of this layout definition. The possible values for this attribute are defined by the XML Schema string datatype.
val (Value)	Specifies the title or description of this layout definition. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Name">
   <attribute name="lang" type="xsd:string" use="optional" default=""/>
   <attribute name="val" type="xsd:string" use="required"/>
</complexType>
```

5.9.2.31 varLst (Variable List)

This element consists of a list of variables which interact with user interface components.

[*Example*: Consider the following example of a varLst in a DrawingML diagram:

```
<varLst>
    <chMax val="2" />
    <dir val="norm" />
    <resizeHandles val="exact" />
</varLst>
```

In this example we see different variables being defined which modify the behavior of user interface components directly. *end example*]

Parent Elements
layoutNode (§5.9.2.19)

Child Elements	Subclause
animLvl (Level Animation)	§5.9.6.1
animOne (One by One Animation String)	§5.9.6.2
bulletEnabled (Show Insert Bullet)	§5.9.6.3
chMax (Maximum Children)	§5.9.6.4
chPref (Preferred Number of Children)	§5.9.6.5
dir (Diagram Direction)	§5.9.6.6
hierBranch (Organization Chart Branch Style)	§5.9.6.7
orgChart (Show Organization Chart User Interface)	§5.9.6.8
resizeHandles (Shape Resize Style)	§5.9.2.23

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LayoutVariablePropertySet">
   <sequence>
      <element name="orgChart" type="CT_OrgChart" minOccurs="0" maxOccurs="1"/>
      <element name="chMax" type="CT_ChildMax" minOccurs="0" maxOccurs="1"/>
      <element name="chPref" type="CT_ChildPref" minOccurs="0" maxOccurs="1"/>
      <element name="bulletEnabled" type="CT_BulletEnabled" minOccurs="0" maxOccurs="1"/>
      <element name="dir" type="CT_Direction" minOccurs="0" maxOccurs="1"/>
      <element name="hierBranch" type="CT_HierBranchStyle" minOccurs="0" maxOccurs="1"/>
      <element name="animOne" type="CT_AnimOne" minOccurs="0" maxOccurs="1"/>
      <element name="animLvl" type="CT_AnimLvl" minOccurs="0" maxOccurs="1"/>
      <element name="resizeHandles" type="CT_ResizeHandles" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.9.3 Data

This section specifies the data that is to be contained within a diagram.

5.9.3.1 bg (Background Formatting)

This element defines formatting that can be applied to the background shape of the entire diagram. The background shape can hold formatting options just as a normal shape can hold within DrawingML.

[*Example*: Consider the following example of a bg in DiagramML:

```
<bg>
     <solidFill>
          <schemeClr val="tx1"/>
     </solidFill>
     <effectLst>
          <glow rad="152400">
               <schemeClr val="accent1">
                    <alpha val="75000"/>
               </schemeClr>
          </glow>
     </effectLst>
</bg>
```

In this example we see a solid fill applied to the background of the diagram along with a glow. *end example*]

Parent Elements
dataModel (§5.9.2.10)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
effectDag (Effect Container)	§5.1.10.25
effectLst (Effect Container)	§5.1.10.26
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
solidFill (Solid Fill)	§5.1.10.54

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BackgroundFormatting">
   <sequence>
      <group ref="EG_FillProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_EffectProperties" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.9.3.2 cxn (Connection)

This element defines a connection between two points. A connection defines a relationship between two points in a diagram.

[*Example*: Consider the following example of a cxn in DiagramML:

```
<cxnLst>
    <cxn modelId="7" srcId="0" destId="1" srcOrd="0" destOrd="0"/>
    <cxn modelId="8" srcId="0" destId="2" srcOrd="1" destOrd="0"/>
    <cxn modelId="9" srcId="0" destId="3" srcOrd="2" destOrd="0"/>
    <cxn modelId="10" srcId="0" destId="4" srcOrd="3" destOrd="0"/>
    <cxn modelId="11" srcId="0" destId="5" srcOrd="4" destOrd="0"/>
    <cxn modelId="12" srcId="0" destId="6" srcOrd="5" destOrd="0"/>
</cxnLst>
```

In this example we see 6 cxn elements defined within a cxnLst element (§5.9.3.3). In this example, a relationship is being defined between point 0 and every other point in the diagram. *end example*]

Parent Elements
cxnLst (§5.9.3.3)

Child Elements	Subclause
extLst (Extension List)	§5.9.2.13

Attributes	Description
destId (Destination Identifier)	The model identifier of the destination point for a connection. [*Example*: Consider the following example cxn within DiagramML: `<cxn modelId="10" srcId="0" destId="4" srcOrd="3" destOrd="0"/>` In this example we see the destination identifier referencing a point who's model identifier is 4. *end example*] The possible values for this attribute are defined by the ST_ModelId simple type

Attributes	Description
	(§5.9.7.42).
destOrd (Destination Position)	The relative position of the destination point among it's siblings. [*Example*: Consider the following example cxn within DiagramML: `<cxn modelId="10" srcId="0" destId="4" srcOrd="3" destOrd="0"/>` In this example we see the destination position is 0. This means that it is ranked first among its siblings if there are sibling points present. *end example*] The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
modelId (Model Identifier)	The unique identifier associated with this cxn. [*Example*: Consider the following example cxn within DiagramML: `<cxn modelId="10" srcId="0" destId="4" srcOrd="3" destOrd="0"/>` In this example we see the model identifier is 10. *end example*] The possible values for this attribute are defined by the ST_ModelId simple type (§5.9.7.42).
parTransId (Parent Transition Identifier)	The model identifier of the point representing the parent transition. An example of a parent transition can be thought of as the shape connecting two points, such as an arrow in the diagram. The unique identifier associated with this cxn. [*Example*: Consider the following example cxn within DiagramML: `<cxn modelId="10" srcId="0" destId="4" srcOrd="3" destOrd="0" parTransId="9" sibTransId="5"/>` In this example we see the parent transition identifier is referencing a point who's model identifier is 9. *end example*] The possible values for this attribute are defined by the ST_ModelId simple type (§5.9.7.42).
presId (Presentation Identifier)	The unique identifier of the layout associated to the cxn (only the active presentation (layout) is saved so all the presId's in the file should be the same). [*Example*: Consider the following example cxn within DiagramML: `<cxn modelId="10" type="presParOf" srcId="0" destId="4" srcOrd="3" destOrd="0" presId="urn:sampleLayouts/layout1"/>`

Attributes	Description
	In this example we see the presentation identifier is `urn:sampleLayouts/layout1`. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
sibTransId (Sibling Transition Identifier)	The model identifier of the point representing the sibling transition. An example of a sibling transition can be thought of as the shape connecting two points, such as an arrow in the diagram. [*Example*: Consider the following example cxn within DiagramML: `<cxn modelId="10" srcId="0" destId="4" srcOrd="3" destOrd="0" parTransId="9" sibTransId="5"/>` In this example we see the sibling transition identifier is referencing a point who's model identifier is 5. *end example*] The possible values for this attribute are defined by the ST_ModelId simple type (§5.9.7.42).
srcId (Source Identifier)	The model identifier of the source point for a connection. [*Example*: Consider the following example cxn within DiagramML: `<cxn modelId="10" srcId="0" destId="4" srcOrd="3" destOrd="0"/>` In this example we see the souce identifier referencing a point who's model identifier is 0. *end example*] The possible values for this attribute are defined by the ST_ModelId simple type (§5.9.7.42).
srcOrd (Source Position)	The relative position of the source point among it's siblings. [*Example*: Consider the following example cxn within DiagramML: `<cxn modelId="10" srcId="0" destId="4" srcOrd="3" destOrd="0"/>` In this example we see the source position is 3. This means that it is ranked third among its siblings. *end example*] The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
type (Point Type)	The type of point, which will correspond to a connection in this case. [*Example*: Consider the following example cxn within DiagramML:

Attributes	Description
	<cxn modelId="10" type="presParOf" srcId="0" destId="4" srcOrd="3" destOrd="0" presId="urn:sampleLayouts/layout1"/> In this example we see the point type is defined as presParOf. *end example*] The possible values for this attribute are defined by the ST_CxnType simple type (§5.9.7.23).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Cxn">
   <sequence>
      <element name="extLst" type="a:CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="modelId" type="ST_ModelId" use="required"/>
   <attribute name="type" type="ST_CxnType" use="optional" default="parOf"/>
   <attribute name="srcId" type="ST_ModelId" use="required"/>
   <attribute name="destId" type="ST_ModelId" use="required"/>
   <attribute name="srcOrd" type="xsd:unsignedInt" use="required"/>
   <attribute name="destOrd" type="xsd:unsignedInt" use="required"/>
   <attribute name="parTransId" type="ST_ModelId" use="optional" default="0"/>
   <attribute name="sibTransId" type="ST_ModelId" use="optional" default="0"/>
   <attribute name="presId" type="xsd:string" use="optional" default=""/>
</complexType>
```

5.9.3.3 cxnLst (Connection List)

This element defines a group of connections. There can be a connection list defined for any data model which holds all of the connections between points defined in the diagram.

[*Example*: Consider the following example of a cxnLst in DiagramML:

```
<cxnLst>
   <cxn modelId="7" srcId="0" destId="1" srcOrd="0" destOrd="0"/>
   <cxn modelId="8" srcId="0" destId="2" srcOrd="1" destOrd="0"/>
   <cxn modelId="9" srcId="0" destId="3" srcOrd="2" destOrd="0"/>
   <cxn modelId="10" srcId="0" destId="4" srcOrd="3" destOrd="0"/>
   <cxn modelId="11" srcId="0" destId="5" srcOrd="4" destOrd="0"/>
   <cxn modelId="12" srcId="0" destId="6" srcOrd="5" destOrd="0"/>
</cxnLst>
```

In this example we see 6 cxn elements (§5.9.3.2) defined within a cxnLst element. In this example, a relationship is being defined between point 0 and every other point in the diagram. *end example*]

Parent Elements
dataModel (§5.9.2.10)

Child Elements	Subclause
cxn (Connection)	§5.9.3.2

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CxnList">
   <sequence>
      <element name="cxn" type="CT_Cxn" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.9.3.4 prSet (Property Set)

This element holds properties and customizations which are used throughout certain elements in DiagramML. The properties can be grouped into the following general categories:

- Presentation Properties - presLayoutVars, style, presAssocId, presName, presStyleLbl, presStyleIdx, presStyleCnt
- Document Properties - loTypeId, loCatId, qsTypeId, qaCatId, csTypeId, coherent3DOff
- Semantic Element Properties - phldrT, phldr
- Customization Properties - custAng, custFlipVert, custFlipHor, custSzX, custSzY, custScaleX, custScaleY, custT, custLinFactX, custLinFactY, custLinFactNeighborX, custLinFactNeighborY, custRadScaleRad, custRadScaleInc

[*Example*: Consider the basic example of prSet in use in DrawingML on a document point type:

```
<prSet loTypeId="urn:microsoft.com/office/officeart/2005/8/layout/default"
loCatId="list"
qsTypeId="urn:microsoft.com/office/officeart/2005/8/quickstyle/3d1" qsCatId="3D"
csTypeId="urn:microsoft.com/office/officeart/2005/8/colors/colorful2"
csCatId="colorful" phldr="1"/>
```

In this example we define the layout identifier, the category of the layout, the quick style identifier, the quick style category, along with the color style and color style category. *end example*]

Parent Elements
pt (§5.9.3.5)

Child Elements	Subclause
presLayoutVars (Presentation Layout Variables)	§5.9.5.4
style (Shape Style)	§5.9.2.28

Attributes	Description

Attributes	Description
coherent3DOff (Coherent 3D Behavior)	Enables or disables the Coherent 3D behavior for styles that specify this property. The possible values for this attribute are defined by the XML Schema boolean datatype.
csCatId (Color Transform Category)	This attribute specifies the identifier of the current color transform category. The possible values for this attribute are defined by the XML Schema string datatype.
csTypeId (Color Transform Type Identifier)	This attribute specifies the identifier of the currently applied color transform. The possible values for this attribute are defined by the XML Schema string datatype.
custAng (Custom Rotation)	Specifies the amount that rotation is customized by. The possible values for this attribute are defined by the XML Schema int datatype.
custFlipHor (Custom Horizontal Flip)	Specifies if there is a custom horizontal flip applied. The possible values for this attribute are defined by the XML Schema boolean datatype.
custFlipVert (Custom Vertical Flip)	Specifies if there is a custom vertical flip applied. The possible values for this attribute are defined by the XML Schema boolean datatype.
custLinFactNeighborX (Neighbor Offset Width)	Specifies the percentage of the neighbor's width used for offsetting shape. The possible values for this attribute are defined by the XML Schema int datatype.
custLinFactNeighborY (Neighbor Offset Height)	Specifies the percentage of the neighbor's height used for offsetting shape. The possible values for this attribute are defined by the XML Schema int datatype.
custLinFactX (Custom Factor Width)	Specifies the percentage of the current shape width used for offsetting the shape. The possible values for this attribute are defined by the XML Schema int datatype.
custLinFactY (Custom Factor Height)	Specifies the percentage of the current shape height used for offsetting the shape. The possible values for this attribute are defined by the XML Schema int datatype.
custRadScaleInc (Include Angle Scale)	Specifies the amount that the include angle has been scaled by. The possible values for this attribute are defined by the XML Schema int datatype.
custRadScaleRad (Radius Scale)	Specifies how much the radius has been scaled. The possible values for this attribute are defined by the XML Schema int datatype.
custScaleX (Width Scale)	Specifies the amount that the width has been scaled by. The possible values for this attribute are defined by the XML Schema int datatype.
custScaleY (Height Scale)	Specifies the amount that the height has been scaled by.

Attributes	Description
	The possible values for this attribute are defined by the XML Schema int datatype.
custSzX (Fixed Width Override)	Specifies a fixed width override for a shape. The possible values for this attribute are defined by the XML Schema int datatype.
custSzY (Fixed Height Override)	Specifies a fixed height override for a shape. The possible values for this attribute are defined by the XML Schema int datatype.
custT (Text Changed)	Specifies if the text has been customized which allows layout to ignore automatic formatting options available to the text. The possible values for this attribute are defined by the XML Schema boolean datatype.
loCatId (Current Diagram Category)	Specifies the current identifier of the layout category applied to the diagram. The possible values for this attribute are defined by the XML Schema string datatype.
loTypeId (Current Diagram Type)	Specifies the identifier for the layout currently applied to the diagram. The possible values for this attribute are defined by the XML Schema string datatype.
phldr (Placeholder)	Indicates that the point is a placeholder or sample item. The possible values for this attribute are defined by the XML Schema boolean datatype.
phldrT (Placeholder Text)	The text used for display in the element if the placeholder flag is set to `true`. If this property is not set than default placeholder text will be used. The possible values for this attribute are defined by the XML Schema string datatype.
presAssocID (Presentation Element Identifier)	The point associated with this presentation element. This identifier is used together with presName to create a unique key for presentation element indexing. The possible values for this attribute are defined by the ST_ModelId simple type (§5.9.7.42).
presName (Presentation Name)	The layout node name of this presentation element. This name is used together with presAssocID to create a unique key for presentation element indexing. The possible values for this attribute are defined by the XML Schema string datatype.
presStyleCnt (Presentation Style Count)	Specifies the layout node style count of this presentation element. The possible values for this attribute are defined by the XML Schema int datatype.
presStyleIdx (Presentation Style Index)	Specifies the layout node style index of this presentation element. The possible values for this attribute are defined by the XML Schema int datatype.
presStyleLbl (Presentation Style Label)	Specifies the layout node style label of this presentation element. The possible values for this attribute are defined by the XML Schema string datatype.

Attributes	Description
qsCatId (Current Style Category)	Specifies the identifier of the category of the currently applied quick style. The possible values for this attribute are defined by the XML Schema string datatype.
qsTypeId (Current Style Type)	Specifies the identifier of the currently applied quick style. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ElemPropSet">
   <sequence>
      <element name="presLayoutVars" type="CT_LayoutVariablePropertySet" minOccurs="0"
        maxOccurs="1"/>
      <element name="style" type="a:CT_ShapeStyle" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="presAssocID" type="ST_ModelId" use="optional"/>
   <attribute name="presName" type="xsd:string" use="optional"/>
   <attribute name="presStyleLbl" type="xsd:string" use="optional"/>
   <attribute name="presStyleIdx" type="xsd:int" use="optional"/>
   <attribute name="presStyleCnt" type="xsd:int" use="optional"/>
   <attribute name="loTypeId" type="xsd:string" use="optional"/>
   <attribute name="loCatId" type="xsd:string" use="optional"/>
   <attribute name="qsTypeId" type="xsd:string" use="optional"/>
   <attribute name="qsCatId" type="xsd:string" use="optional"/>
   <attribute name="csTypeId" type="xsd:string" use="optional"/>
   <attribute name="csCatId" type="xsd:string" use="optional"/>
   <attribute name="coherent3DOff" type="xsd:boolean" use="optional"/>
   <attribute name="phldrT" type="xsd:string" use="optional"/>
   <attribute name="phldr" type="xsd:boolean" use="optional"/>
   <attribute name="custAng" type="xsd:int" use="optional"/>
   <attribute name="custFlipVert" type="xsd:boolean" use="optional"/>
   <attribute name="custFlipHor" type="xsd:boolean" use="optional"/>
   <attribute name="custSzX" type="xsd:int" use="optional"/>
   <attribute name="custSzY" type="xsd:int" use="optional"/>
   <attribute name="custScaleX" type="xsd:int" use="optional"/>
   <attribute name="custScaleY" type="xsd:int" use="optional"/>
   <attribute name="custT" type="xsd:boolean" use="optional"/>
   <attribute name="custLinFactX" type="xsd:int" use="optional"/>
   <attribute name="custLinFactY" type="xsd:int" use="optional"/>
   <attribute name="custLinFactNeighborX" type="xsd:int" use="optional"/>
   <attribute name="custLinFactNeighborY" type="xsd:int" use="optional"/>
   <attribute name="custRadScaleRad" type="xsd:int" use="optional"/>
   <attribute name="custRadScaleInc" type="xsd:int" use="optional"/>
</complexType>
```

5.9.3.5 pt (Point)

This element defines a point in DiagramML. A point in DiagramML is defined to hold data associated with a particular point or node in a diagram. Transitions between nodes in a diagram along with the nodes themselves are defined as different types of points. A point is not only responsible for holding the data associated with a

node in a diagram, but also for holding customization properties made to the text and shape associated with the particular node.

[*Example*: Consider the following example of a pt in DiagramML:

```
<pt modelId="{C6A8900D-3F1B-4F1D-A514-4E8BDD964568}">
    <prSet phldrT="[Text]"/>
    <spPr/>
    <t>
          <bodyPr/>
          <lstStyle/>
          <p>
                <r>
                      <rPr lang="en-US" smtClean="0"/>
                      <t>Text 2</t>
                </r>
                <endParaRPr lang="en-US" dirty="0"/>
          </p>
    </t>
</pt>
```

In this example we define a point which holds the data associated with a node in a diagram. The actual text in the diagram is defined in the text body, t, tag and consists of the string "Text 2". There are no overrides made to the shape properties and placeholder text defined for this node when there is no text body present. *end example*]

Parent Elements
ptLst (§5.9.3.6)

Child Elements	Subclause
extLst (Extension List)	§5.9.2.13
prSet (Property Set)	§5.9.3.4
spPr (Shape Properties)	§5.9.3.7
t (Text Body)	§5.9.3.8

Attributes	Description
cxnId (Connection Identifier)	The model identifier of the connection that represents the transition node. [*Example*: Consider the following example of a cxnId: `<dgm:pt modelId="5" type="parTrans" cxnId="9">`

Attributes	Description
	<dgm:prSet/> <dgm:spPr/> </dgm:pt> In this example we define the connection related to this point to reference connection 9. *end example*] The possible values for this attribute are defined by the ST_ModelId simple type (§5.9.7.42).
modelId (Model Identifier)	The unique identifier of the element within the data model. This identifier should be unique only relative to the containing data model. [*Example*: Consider the following example of a cxnId: <dgm:pt modelId="5" type="parTrans" cxnId="9"> <dgm:prSet/> <dgm:spPr/> </dgm:pt> In this example we define the point type is to be 5. *end example*] The possible values for this attribute are defined by the ST_ModelId simple type (§5.9.7.42).
type (Point Type)	The type of point that this is. [*Example*: Consider the following example of a cxnId: <dgm:pt modelId="5" type="parTrans" cxnId="9"> <dgm:prSet/> <dgm:spPr/> </dgm:pt> In this example the point type is defined as a parTrans point type. *end example*] The possible values for this attribute are defined by the ST_PtType simple type (§5.9.7.50).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Pt">
   <sequence>
      <element name="prSet" type="CT_ElemPropSet" minOccurs="0" maxOccurs="1"/>
      <element name="spPr" type="a:CT_ShapeProperties" minOccurs="0" maxOccurs="1"/>
      <element name="t" type="a:CT_TextBody" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="a:CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="modelId" type="ST_ModelId" use="required"/>
   <attribute name="type" type="ST_PtType" use="optional" default="node"/>
   <attribute name="cxnId" type="ST_ModelId" use="optional" default="0"/>
</complexType>
```

5.9.3.6 ptLst (Point List)

This element simply holds a list of points within the data model.

[*Example*: Consider the following example of a very simple point list in DiagramML:

```
<dgm:ptLst>
    <dgm:pt modelId="0" type="doc"/>
    <dgm:pt modelId="1"/>
    <dgm:pt modelId="2"/>
    <dgm:pt modelId="3"/>
    <dgm:pt modelId="4"/>
    <dgm:pt modelId="5"/>
    <dgm:pt modelId="6"/>
</dgm:ptLst>
```

In this example we define a single document type point and five node type points. *end example*]

Parent Elements
dataModel (§5.9.2.10)

Child Elements	Subclause
pt (Point)	§5.9.3.5

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PtList">
   <sequence>
      <element name="pt" type="CT_Pt" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.9.3.7 spPr (Shape Properties)

This element specifies the properties for a single shape in a diagram's data, as defined using DrawingML child elements.

Parent Elements
pt (§5.9.3.5)

Child Elements	Subclause
blipFill (Picture Fill)	§5.1.10.14
custGeom (Custom Geometry)	§5.1.11.8
effectDag (Effect Container)	§5.1.10.25
effectLst (Effect Container)	§5.1.10.26
extLst (Extension List)	§5.1.2.1.15
gradFill (Gradient Fill)	§5.1.10.33
grpFill (Group Fill)	§5.1.10.35
ln (Outline)	§5.1.2.1.24
noFill (No Fill)	§5.1.10.44
pattFill (Pattern Fill)	§5.1.10.47
prstGeom (Preset geometry)	§5.1.11.18
scene3d (3D Scene Properties)	§5.1.4.1.26
solidFill (Solid Fill)	§5.1.10.54
sp3d (Apply 3D shape properties)	§5.1.7.12
xfrm (2D Transform for Individual Objects)	§5.1.9.6

Attributes	Description
bwMode (Black and White Mode) Namespace: .../drawingml/2006/main	Specifies that the picture should be rendered using only black and white coloring. That is the coloring information for the picture should be converted to either black or white when rendering the picture. No gray is to be used in rendering this image, only stark black and stark white. [*Note*: This does not mean that the picture itself that is stored within the file is necessarily a black and white picture. This attribute instead sets the rendering mode that the picture will have applied to when rendering. *end note*] The possible values for this attribute are defined by the ST_BlackWhiteMode simple type (§5.1.12.10).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShapeProperties">
   <sequence>
      <element name="xfrm" type="CT_Transform2D" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_Geometry" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_FillProperties" minOccurs="0" maxOccurs="1"/>
      <element name="ln" type="CT_LineProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_EffectProperties" minOccurs="0" maxOccurs="1"/>
      <element name="scene3d" type="CT_Scene3D" minOccurs="0" maxOccurs="1"/>
      <element name="sp3d" type="CT_Shape3D" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="bwMode" type="ST_BlackWhiteMode" use="optional"/>
</complexType>
```

5.9.3.8 t (Text Body)

Text body containing the default body, paragraph and character properties. There should be a signle paragraph and no text runs. Any runs in the first paragraph and paragraphs in addition to the first are ignored.

Parent Elements
pt (§5.9.3.5)

Child Elements	Subclause
bodyPr (Body Properties)	§5.1.5.1.1
lstStyle (Text List Styles)	§5.1.5.4.12
p (Text Paragraphs)	§5.1.5.2.6

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextBody">
   <sequence>
      <element name="bodyPr" type="CT_TextBodyProperties" minOccurs="1" maxOccurs="1"/>
      <element name="lstStyle" type="CT_TextListStyle" minOccurs="0" maxOccurs="1"/>
      <element name="p" type="CT_TextParagraph" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.9.3.9 whole (Whole E2O Formatting)

Formatting that applies to the entire diagram object, and not just the background, includes line and effect properties.

Parent Elements
dataModel (§5.9.2.10)

Child Elements	Subclause
effectDag (Effect Container)	§5.1.10.25
effectLst (Effect Container)	§5.1.10.26
ln (Outline)	§5.1.2.1.24

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_WholeE2oFormatting">
   <sequence>
      <element name="ln" type="CT_LineProperties" minOccurs="0" maxOccurs="1"/>
      <group ref="EG_EffectProperties" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.9.4 Color Information

This section defines the coloring information that is to be associated with a diagram.

5.9.4.1 cat (Color Transform Category)

This element specifies the category in the user interface that a color transform is to be displayed within.

[*Example*: Consider the following example of a cat in DiagramML:

```
<dgm:cat type="mainScheme" pri="10300"/>
```

In this example we see a cat defined with a type and priority. *end example*]

Parent Elements
catLst (§5.9.4.2)

Attributes	Description
pri (Priority)	The priority within the category for this color variation determines the order in which it will display in the user interface. The lower numbers are to be displayed at the beginning of the list. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
type (Category Type)	The category type used to organize the color transforms in the user interface. The possible values for this attribute are defined by the XML Schema anyURI datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CTCategory">
   <attribute name="type" type="xsd:anyURI" use="required"/>
   <attribute name="pri" type="xsd:unsignedInt" use="required"/>
</complexType>
```

5.9.4.2 catLst (Color Transform Category List)

This element defines a list of color transform categories. This list can be used to populate user interface components which could separate color transforms into categories.

[*Example*: Consider the following example of a catLst in DiagramML:

```
<dgm:catLst>
     <dgm:cat type="mainScheme" pri="10300"/>
</dgm:catLst>
```

In this example we see a catLst defined which holds a single *color transform category* (§5.9.4.1). *end example*]

Parent Elements
colorsDef (§5.9.4.3); colorsDefHdr (§5.9.4.4)

Child Elements	Subclause
cat (Color Transform Category)	§5.9.4.1

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CTCategories">
   <sequence minOccurs="0" maxOccurs="unbounded">
      <element name="cat" type="CT_CTCategory" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.9.4.3 colorsDef (Color Transform Definitions)

This element is the root element for color transforms. Held within this element are all of the available color transforms themselves along with other elements and attributes associated with defining the general color transform properties and attributes.

[*Example*: Consider the following example of a colorsDef in DiagramML:

```
<dgm:colorsDef
xmlns:dgm="http://schemas.openxmlformats.org/drawingml/2006/3/diagram"
xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/3/main"
uniqueId="urn:microsoft.com/office/officeart/2005/8/colors/accent0_3"
minVer="12.0">
    <dgm:title lang="" val=""/>
    <dgm:desc lang="" val=""/>
    <dgm:catLst>
          <dgm:cat type="mainScheme" pri="10300"/>
    </dgm:catLst>
    <dgm:styleLbl name="node0">
          ...
```

```
    </dgm:styleLbl>
    ...
</dgm:colorsDef>
```

In this example we see a sampling of a colorsDef being defined with a number of styleLbl elements held within the colorsDef. *end example*]

Parent Elements
Root element of DrawingML Diagram Colors part

Child Elements	Subclause
catLst (Color Transform Category List)	§5.9.4.2
desc (Description)	§5.9.4.6
extLst (Extension List)	§5.9.2.13
styleLbl (Style Label)	§5.9.4.10
title (Title)	§5.9.4.11

Attributes	Description

minVer (Minimum Version)	The minimum product version that can support this color transform.
	
	[*Example*: Consider the following example:
	
	`<colorsDef uniqueId="urn:colors/accent0_3" minVer="12.0">`
	`...`
	`</colorsDef>`
	
	In this example we see the minVer set to `12.0`. *end example*]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

uniqueId (Unique ID)	A unique id associated with the color transform definition.
	
	[*Example*: Consider the following example:
	
	`<colorsDef uniqueId="urn:colors/accent0_3" minVer="12.0">`
	`...`
	`</colorsDef>`
	
	In this example we see the uniqueId set to `urn:colors/accent0_3`. *end example*]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ColorTransform">
   <sequence>
      <element name="title" type="CT_CTName" minOccurs="0" maxOccurs="unbounded"/>
      <element name="desc" type="CT_CTDescription" minOccurs="0" maxOccurs="unbounded"/>
      <element name="catLst" type="CT_CTCategories" minOccurs="0"/>
      <element name="styleLbl" type="CT_CTStyleLabel" minOccurs="0" maxOccurs="unbounded"/>
      <element name="extLst" type="a:CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="uniqueId" type="xsd:string" use="optional" default=""/>
   <attribute name="minVer" type="xsd:string" use="optional"
     default="http://schemas.openxmlformats.org/drawingml/2006/diagram"/>
</complexType>
```

5.9.4.4 colorsDefHdr (Color Transform Definition Header)

This element specifies header information associated with a color transform definition. The header information is used by an application to preprocess required data in order to help with possible performance concerns associated with an initial full load of a color transform definition.

[*Example*: Consider the following example of a colorsDefHdr within DiagramML:

```
<colorsDefHdr uniqueId="urn:colors/accent0_1">
    <title val="Main 1" />
    <desc val="" />
    <catLst>
        <cat type="mainScheme" pri="10100" />
    </catLst>
</colorsDefHdr>
```

In this example we see a color transform definition header which defines a title and category for a set of color transforms. *end example*]

Parent Elements
colorsDefHdrLst (§5.9.4.5)

Child Elements	Subclause
catLst (Color Transform Category List)	§5.9.4.2
desc (Description)	§5.9.4.6
extLst (Extension List)	§5.9.2.13
title (Title)	§5.9.4.11

Attributes	Description
minVer (Minimum	The minimum product version that can support the associated color transform definition.

Attributes	Description
Version)	The possible values for this attribute are defined by the XML Schema string datatype.
resId (Resource ID)	This attribute is the id which associates this header to the actual color transform definition. The possible values for this attribute are defined by the XML Schema int datatype.
uniqueId (Unique ID)	This attribute defines a unique identifier for the associated color transform definition. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ColorTransformHeader">
   <sequence>
      <element name="title" type="CT_CTName" minOccurs="1" maxOccurs="unbounded"/>
      <element name="desc" type="CT_CTDescription" minOccurs="1" maxOccurs="unbounded"/>
      <element name="catLst" type="CT_CTCategories" minOccurs="0"/>
      <element name="extLst" type="a:CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="uniqueId" type="xsd:string" use="required"/>
   <attribute name="minVer" type="xsd:string" use="optional"
     default="http://schemas.openxmlformats.org/drawingml/2006/diagram"/>
   <attribute name="resId" type="xsd:int" use="optional" default="0"/>
</complexType>
```

5.9.4.5 colorsDefHdrLst (Color Transform Header List)

This element is simply a list of color transform definition headers and is used to consolidate multiple headers in a group.

Child Elements	Subclause
colorsDefHdr (Color Transform Definition Header)	§5.9.4.4

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ColorTransformHeaderLst">
   <sequence>
      <element name="colorsDefHdr" type="CT_ColorTransformHeader" minOccurs="0"
        maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.9.4.6 desc (Description)

This element holds a description for a color definition. The description can be used to describe the qualities associated with a particular color transform definition.

Parent Elements

Parent Elements
colorsDef (§5.9.4.3); colorsDefHdr (§5.9.4.4)

Attributes	Description
lang (Language)	The natural language of the color transform definition. The possible values for this attribute are defined by the XML Schema string datatype.
val (Description Value)	The string which is used as the description of the color transform definition. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CTDescription">
   <attribute name="lang" type="xsd:string" use="optional" default=""/>
   <attribute name="val" type="xsd:string" use="required"/>
</complexType>
```

5.9.4.7 effectClrLst (Effect Color List)

This element defines a list of colors applied to effects within a color transform.

[*Example*: Consider the following example of an effectClrLst in DiagramML:

```
<dgm:effectClrLst meth="repeat">
    <a:schemeClr val="dk2">
          <a:tint val="60000"/>
    </a:schemeClr>
</dgm:effectClrLst>
```

In this example we see a single color defined in the effectClrLst, more specifically we see a scheme color being utilized with a tint applied to the color. *end example*]

Parent Elements
styleLbl (§5.9.4.10)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

Attributes	Description

hueDir (Hue Direction)	The direction around the color wheel the hue shift (if defined) will occur.
	
	[*Example*: Consider the following example of a hueDir in use:
	
	`<dgm:lnClrLst hueDir="cw" meth="repeat">`
	`  <a:schemeClr val="dk2">`
	`  <a:tint val="60000"/>`
	`  </a:schemeClr>`
	`</dgm:lnClrLst>`
	
	In this example an lnClrLst is defined with a hue direction defined as clockwise. *end example*]
	
	The possible values for this attribute are defined by the ST_HueDir simple type (§5.9.7.37).

meth (Color Application Method Type)	The type of method used to apply the color transform.
	
	[*Example*: Consider the following example of a meth in use:
	
	`<dgm:lnClrLst hueDir="cw" meth="repeat">`
	`  <a:schemeClr val="dk2">`
	`  <a:tint val="60000"/>`
	`  </a:schemeClr>`
	`</dgm:lnClrLst>`
	
	In this example and lnClrLst is defined using the `repeat` color application method. *end example*]
	
	The possible values for this attribute are defined by the ST_ClrAppMethod simple type (§5.9.7.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Colors">
   <sequence>
      <group ref="a:EG_ColorChoice" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="meth" type="ST_ClrAppMethod" use="optional" default="span"/>
   <attribute name="hueDir" type="ST_HueDir" use="optional" default="cw"/>
</complexType>
```

5.9.4.8 fillClrLst (Fill Color List)

This element defines a list of colors which are used as fill colors in the color transform. The fill colors define the color of the nodes in a diagram.

[*Example*: Consider the following example of a fillClrLst in DiagramML:

```
<dgm:fillClrLst meth="repeat">
     <a:schemeClr val="dk2"/>
</dgm:fillClrLst>
```

In this example the fillClrList contains a single scheme color and is utilizing the repeat method for color application. *end example*]

Parent Elements
styleLbl (§5.9.4.10)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

Attributes	Description

hueDir (Hue Direction)	The direction around the color wheel the hue shift (if defined) will occur.
	
	[*Example*: Consider the following example of a hueDir in use:
	
	`<dgm:lnClrLst hueDir="cw" meth="repeat">`
	`  <a:schemeClr val="dk2">`
	`  <a:tint val="60000"/>`
	`  </a:schemeClr>`
	`</dgm:lnClrLst>`
	
	In this example an lnClrLst is defined with a hue direction defined as clockwise. *end example*]
	
	The possible values for this attribute are defined by the ST_HueDir simple type (§5.9.7.37).

meth (Color Application Method Type)	The type of method used to apply the color transform.
	
	[*Example*: Consider the following example of a meth in use:
	
	`<dgm:lnClrLst hueDir="cw" meth="repeat">`
	`  <a:schemeClr val="dk2">`

Attributes	Description
	<a:tint val="60000"/> </a:schemeClr> </dgm:lnClrLst> In this example and lnClrLst is defined using the repeat color application method. *end example*] The possible values for this attribute are defined by the ST_ClrAppMethod simple type (§5.9.7.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Colors">
   <sequence>
      <group ref="a:EG_ColorChoice" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="meth" type="ST_ClrAppMethod" use="optional" default="span"/>
   <attribute name="hueDir" type="ST_HueDir" use="optional" default="cw"/>
</complexType>
```

5.9.4.9 linClrLst (Line Color List)

This element defines a list of colors which are used as line colors in the color transform. The line colors define the color of the lines used on a given node in a diagram

[*Example*: Consider the following example of a lnClrLst in DiagramML:

```
<dgm:lnClrLst meth="repeat">
     <a:schemeClr val="dk2"/>
</dgm:lnClrLst>
```

In this example the lnClrList contains a single scheme color and is utilizing the repeat method for color application. *end example*]

Parent Elements
styleLbl (§5.9.4.10)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

Attributes	Description

hueDir (Hue Direction)	The direction around the color wheel the hue shift (if defined) will occur.
	
	[*Example*: Consider the following example of a hueDir in use:
	
	`<dgm:lnClrLst hueDir="cw" meth="repeat">`
	`  <a:schemeClr val="dk2">`
	`  <a:tint val="60000"/>`
	`  </a:schemeClr>`
	`</dgm:lnClrLst>`
	
	In this example an lnClrLst is defined with a hue direction defined as clockwise. *end example*]
	
	The possible values for this attribute are defined by the ST_HueDir simple type (§5.9.7.37).

meth (Color Application Method Type)	The type of method used to apply the color transform.
	
	[*Example*: Consider the following example of a meth in use:
	
	`<dgm:lnClrLst hueDir="cw" meth="repeat">`
	`  <a:schemeClr val="dk2">`
	`  <a:tint val="60000"/>`
	`  </a:schemeClr>`
	`</dgm:lnClrLst>`
	
	In this example and lnClrLst is defined using the `repeat` color application method. *end example*]
	
	The possible values for this attribute are defined by the ST_ClrAppMethod simple type (§5.9.7.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Colors">
   <sequence>
      <group ref="a:EG_ColorChoice" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="meth" type="ST_ClrAppMethod" use="optional" default="span"/>
   <attribute name="hueDir" type="ST_HueDir" use="optional" default="cw"/>
</complexType>
```

5.9.4.10 styleLbl (Style Label)

This element defines a style label. The style label is used to define a color transform that is applied to a given node in a diagram.

[*Example*: Consider the following example of a styleLbl in DiagramML:

```
<dgm:styleLbl name="exampleStyleLabel">
    <dgm:fillClrLst>
          <a:schemeClr val="accent2"/>
          <a:schemeClr val="accent3"/>
    </dgm:fillClrLst>
    <dgm:linClrLst meth="repeat">
          <a:schemeClr val="lt1"/>
    </dgm:linClrLst>
    <dgm:effectClrLst/>
    <dgm:txLinClrLst/>
    <dgm:txFillClrLst/>
    <dgm:txEffectClrLst/>
</dgm:styleLbl>
```

In this example we see a style label defined in its entirety. This style label can be used on a layout node in order to define the color transform that is to be applied to the node. *end example*]

Parent Elements
colorsDef (§5.9.4.3)

Child Elements	Subclause
effectClrLst (Effect Color List)	§5.9.4.7
extLst (Extension List)	§5.9.2.13
fillClrLst (Fill Color List)	§5.9.4.8
linClrLst (Line Color List)	§5.9.4.9
txEffectClrLst (Text Effect Color List)	§5.9.4.12
txFillClrLst (Text Fill Color List)	§5.9.4.13
txLinClrLst (Text Line Color List)	§5.9.4.14

Attributes	Description

name (Name)	A name given to the style label. This name can be referenced by layout nodes in order to apply the style label to the layout node.
	
	[*Example*: Consider the following example of a styleLbl in DiagramML:
	`<dgm:styleLbl name="exampleStyleLabel">`
	`...`
	`</dgm:styleLbl>`
	
	In this example we see a style label defined with the name `exampleStyleLabel`

Attributes	Description
	defined. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CTStyleLabel">
   <sequence>
      <element name="fillClrLst" type="CT_Colors" minOccurs="0" maxOccurs="1"/>
      <element name="linClrLst" type="CT_Colors" minOccurs="0" maxOccurs="1"/>
      <element name="effectClrLst" type="CT_Colors" minOccurs="0" maxOccurs="1"/>
      <element name="txLinClrLst" type="CT_Colors" minOccurs="0" maxOccurs="1"/>
      <element name="txFillClrLst" type="CT_Colors" minOccurs="0" maxOccurs="1"/>
      <element name="txEffectClrLst" type="CT_Colors" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="a:CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="name" type="xsd:string" use="required"/>
</complexType>
```

5.9.4.11 title (Title)

The name or title given to the color definition header.

Parent Elements
colorsDef (§5.9.4.3); colorsDefHdr (§5.9.4.4)

Attributes	Description
lang (Language)	The natural language of the title or description of a color transform definition. The possible values for this attribute are defined by the XML Schema string datatype.
val (Description Value)	A string used for a description of a color transform definition. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CTName">
   <attribute name="lang" type="xsd:string" use="optional" default=""/>
   <attribute name="val" type="xsd:string" use="required"/>
</complexType>
```

5.9.4.12 txEffectClrLst (Text Effect Color List)

This element defines a list of colors which are used as text effect colors in the color transform. The text effect colors define the color of the text effects used on a given node in a diagram

[*Example*: Consider the following example of a txEffectClrLst in DiagramML:

```
<dgm:txEffectClrLst meth="repeat">
     <a:schemeClr val="dk2"/>
</dgm:txEffectClrLst>
```

In this example the txEffectClrLst contains a single scheme color and is utilizing the repeat method for color application. *end example*]

Parent Elements
styleLbl (§5.9.4.10)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

Attributes	Description

hueDir (Hue Direction)	The direction around the color wheel the hue shift (if defined) will occur.
	
	[*Example*: Consider the following example of a hueDir in use:
	
	`<dgm:lnClrLst hueDir="cw" meth="repeat">`
	`  <a:schemeClr val="dk2">`
	`  <a:tint val="60000"/>`
	`  </a:schemeClr>`
	`</dgm:lnClrLst>`
	
	In this example an lnClrLst is defined with a hue direction defined as clockwise. *end example*]
	
	The possible values for this attribute are defined by the ST_HueDir simple type (§5.9.7.37).

meth (Color Application Method Type)	The type of method used to apply the color transform.
	
	[*Example*: Consider the following example of a meth in use:
	
	`<dgm:lnClrLst hueDir="cw" meth="repeat">`
	`  <a:schemeClr val="dk2">`
	`  <a:tint val="60000"/>`
	`  </a:schemeClr>`

Attributes	Description
	</dgm:lnClrLst> In this example and lnClrLst is defined using the repeat color application method. *end example*] The possible values for this attribute are defined by the ST_ClrAppMethod simple type (§5.9.7.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Colors">
   <sequence>
      <group ref="a:EG_ColorChoice" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="meth" type="ST_ClrAppMethod" use="optional" default="span"/>
   <attribute name="hueDir" type="ST_HueDir" use="optional" default="cw"/>
</complexType>
```

5.9.4.13 txFillClrLst (Text Fill Color List)

This element defines a list of colors which are used as text colors in the color transform. The text colors define the color of the text used in a given node in a diagram

[*Example*: Consider the following example of a txFillClrLst in DiagramML:

```
<dgm:txFillClrLst meth="repeat">
     <a:schemeClr val="dk2"/>
</dgm:txFillClrLst>
```

In this example the txFillClrLst contains a single scheme color and is utilizing the repeat method for color application. *end example*]

Parent Elements
styleLbl (§5.9.4.10)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

Attributes	Description

hueDir (Hue Direction)	The direction around the color wheel the hue shift (if defined) will occur.
	
	[*Example*: Consider the following example of a hueDir in use:
	
	`<dgm:lnClrLst hueDir="cw" meth="repeat">`
	`  <a:schemeClr val="dk2">`
	`  <a:tint val="60000"/>`
	`  </a:schemeClr>`
	`</dgm:lnClrLst>`
	
	In this example an lnClrLst is defined with a hue direction defined as clockwise. *end example*]
	
	The possible values for this attribute are defined by the ST_HueDir simple type (§5.9.7.37).

meth (Color Application Method Type)	The type of method used to apply the color transform.
	
	[*Example*: Consider the following example of a meth in use:
	
	`<dgm:lnClrLst hueDir="cw" meth="repeat">`
	`  <a:schemeClr val="dk2">`
	`  <a:tint val="60000"/>`
	`  </a:schemeClr>`
	`</dgm:lnClrLst>`
	
	In this example and lnClrLst is defined using the `repeat` color application method. *end example*]
	
	The possible values for this attribute are defined by the ST_ClrAppMethod simple type (§5.9.7.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Colors">
   <sequence>
      <group ref="a:EG_ColorChoice" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="meth" type="ST_ClrAppMethod" use="optional" default="span"/>
   <attribute name="hueDir" type="ST_HueDir" use="optional" default="cw"/>
</complexType>
```

5.9.4.14 txLinClrLst (Text Line Color List)

This element defines a list of colors which are used as text line colors in the color transform. The text line colors define the color of the line on text used in a given node in a diagram

[*Example*: Consider the following example of a txLinClrLst in DiagramML:

```
<dgm:txLinClrLst meth="repeat">
     <a:schemeClr val="dk2"/>
</dgm:txLinClrLst>
```

In this example the txLinClrLst contains a single scheme color and is utilizing the repeat method for color application. *end example*]

Parent Elements
styleLbl (§5.9.4.10)

Child Elements	Subclause
hslClr (Hue, Saturation, Luminance Color Model)	§5.1.2.2.13
prstClr (Preset Color)	§5.1.2.2.22
schemeClr (Scheme Color)	§5.1.2.2.29
scrgbClr (RGB Color Model - Percentage Variant)	§5.1.2.2.30
srgbClr (RGB Color Model - Hex Variant)	§5.1.2.2.32
sysClr (System Color)	§5.1.2.2.33

Attributes	Description

hueDir (Hue Direction)	The direction around the color wheel the hue shift (if defined) will occur.
	
	[*Example*: Consider the following example of a hueDir in use:
	
	`<dgm:lnClrLst hueDir="cw" meth="repeat">`
	`  <a:schemeClr val="dk2">`
	`  <a:tint val="60000"/>`
	`  </a:schemeClr>`
	`</dgm:lnClrLst>`
	
	In this example an lnClrLst is defined with a hue direction defined as clockwise. *end example*]
	
	The possible values for this attribute are defined by the ST_HueDir simple type (§5.9.7.37).

meth (Color Application Method Type)	The type of method used to apply the color transform.
	
	[*Example*: Consider the following example of a meth in use:
	
	`<dgm:lnClrLst hueDir="cw" meth="repeat">`
	`  <a:schemeClr val="dk2">`
	`  <a:tint val="60000"/>`
	`  </a:schemeClr>`

Attributes	Description
	</dgm:lnClrLst> In this example and lnClrLst is defined using the repeat color application method. *end example*] The possible values for this attribute are defined by the ST_ClrAppMethod simple type (§5.9.7.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Colors">
   <sequence>
      <group ref="a:EG_ColorChoice" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="meth" type="ST_ClrAppMethod" use="optional" default="span"/>
   <attribute name="hueDir" type="ST_HueDir" use="optional" default="cw"/>
</complexType>
```

5.9.5 Style Definitions

This section describes the styling information to be associated with a diagram.

5.9.5.1 cat (Category)

The category in the user interface where this quick style will display in the user interface.

[*Example*: Consider the following example of a cat in use in DiagramML:

```
<catLst>
    <cat type="3D" pri="11100"/>
</catLst>
```

In this example we see a 3D category type being defined with a priority of 11100. *end example*]

Parent Elements
catLst (§5.9.5.2)

Attributes	Description
pri (Priority)	The priority within the category for this style determines the order in which it will display in the user interface. Lower numbers are displayed at the beginning of the list. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.
type (Category Type)	Category type. This is used to organize the quick style in the user interface. The possible values for this attribute are defined by the XML Schema anyURI datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SDCategory">
   <attribute name="type" type="xsd:anyURI" use="required"/>
   <attribute name="pri" type="xsd:unsignedInt" use="required"/>
</complexType>
```

5.9.5.2 catLst (Category List)

This element is simply a list of categories.

[*Example*: Consider the following example of a catLst in use in DiagramML:

```
<catLst>
    <cat type="Simple" pri="10000"/>
    <cat type="3D" pri="11100"/>
</catLst>
```

In this example two categories defined in the category list. *end example*]

Parent Elements
styleDef (§5.9.5.7); styleDefHdr (§5.9.5.8)

Child Elements	Subclause
cat (Category)	§5.9.5.1

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SDCategories">
   <sequence minOccurs="0" maxOccurs="unbounded">
      <element name="cat" type="CT_SDCategory" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.9.5.3 desc (Style Label Description)

This element defines a description for a style label definition. The description is simply a string describing the characteristics of the style label definition.

[*Example*: Consider the following example of the desc element in DiagramML:

```
<desc lang="" val="3-D Style 1"/>
```

In this example we define the description to be `3-D Style 1`. *end example*]

Parent Elements
styleDef (§5.9.5.7); styleDefHdr (§5.9.5.8)

Attributes	Description
lang (Natural Language)	The natural language of the title or description of this quick style. The possible values for this attribute are defined by the XML Schema string datatype.
val (Description Value)	The string used for the description. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SDDescription">
   <attribute name="lang" type="xsd:string" use="optional" default=""/>
   <attribute name="val" type="xsd:string" use="required"/>
</complexType>
```

5.9.5.4 presLayoutVars (Presentation Layout Variables)

This element specified the layout property set. This set of properties determine different aspects concerning the layout of a diagram. All of the elements associated with enabling or disabling aspects of the user interface are also defined here.

Parent Elements
prSet (§5.9.3.4)

Child Elements	Subclause
animLvl (Level Animation)	§5.9.6.1
animOne (One by One Animation String)	§5.9.6.2
bulletEnabled (Show Insert Bullet)	§5.9.6.3
chMax (Maximum Children)	§5.9.6.4
chPref (Preferred Number of Children)	§5.9.6.5
dir (Diagram Direction)	§5.9.6.6
hierBranch (Organization Chart Branch Style)	§5.9.6.7
orgChart (Show Organization Chart User Interface)	§5.9.6.8
resizeHandles (Shape Resize Style)	§5.9.2.23

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LayoutVariablePropertySet">
   <sequence>
      <element name="orgChart" type="CT_OrgChart" minOccurs="0" maxOccurs="1"/>
      <element name="chMax" type="CT_ChildMax" minOccurs="0" maxOccurs="1"/>
      <element name="chPref" type="CT_ChildPref" minOccurs="0" maxOccurs="1"/>
      <element name="bulletEnabled" type="CT_BulletEnabled" minOccurs="0" maxOccurs="1"/>
      <element name="dir" type="CT_Direction" minOccurs="0" maxOccurs="1"/>
      <element name="hierBranch" type="CT_HierBranchStyle" minOccurs="0" maxOccurs="1"/>
      <element name="animOne" type="CT_AnimOne" minOccurs="0" maxOccurs="1"/>
      <element name="animLvl" type="CT_AnimLvl" minOccurs="0" maxOccurs="1"/>
      <element name="resizeHandles" type="CT_ResizeHandles" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.9.5.5 scene3d (3-D Scene)

The 3-D scene which consists of a camera, a light rig, and an optional backdrop to catch shadows.

Parent Elements
styleDef (§5.9.5.7); styleLbl (§5.9.5.10)

Child Elements	Subclause
backdrop (Backdrop Plane)	§5.1.7.2
camera (Camera)	§5.1.7.5
extLst (Extension List)	§5.1.2.1.15
lightRig (Light Rig)	§5.1.7.9

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Scene3D">
   <sequence>
      <element name="camera" type="CT_Camera" minOccurs="1" maxOccurs="1"/>
      <element name="lightRig" type="CT_LightRig" minOccurs="1" maxOccurs="1"/>
      <element name="backdrop" type="CT_Backdrop" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.9.5.6 sp3d (3-D Shape Properties)

A set of 3-D properties which a shape can contain.

Parent Elements
styleLbl (§5.9.5.10)

Child Elements	Subclause
bevelB (Bottom Bevel)	§5.1.7.3
bevelT (Top Bevel)	§5.1.7.4
contourClr (Contour Color)	§5.1.7.6
extLst (Extension List)	§5.1.2.1.15
extrusionClr (Extrusion Color)	§5.1.7.7

Attributes	Description
contourW (Contour Width) Namespace: .../drawingml/2006/main	Defines the width of the contour on the shape. [*Example*: Consider the following example of a contourW in use within the sp3d element: `<a:sp3d extrusionH="165100" contourW="50800" prstMaterial="plastic"> <a:bevelT w="254000" h="254000"/> <a:bevelB w="254000" h="254000"/> <a:extrusionClr> <a:srgbClr val="FF0000"/> </a:extrusionClr> <a:contourClr> <a:schemeClr val="accent3"/> </a:contourClr> >/a:sp3d>` In this example, we see a countourW defined as `50800`. *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).
extrusionH (Extrusion Height) Namespace: .../drawingml/2006/main	Defines the height of the extrusion applied to the shape. [*Example*: Consider the following example of an extrusionH in use within the sp3d element: `<a:sp3d extrusionH="165100" contourW="50800" prstMaterial="plastic"> < <a:bevelT w="254000" h="254000"/> <a:bevelB w="254000" h="254000"/> <a:extrusionClr> <a:srgbClr val="FF0000"/> </a:extrusionClr> <a:contourClr> <a:schemeClr val="accent3"/> </a:contourClr>`

Attributes	Description
	</a:sp3d> In this example, we see a extrusionH defined as 165100. *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§5.1.12.42).
prstMaterial (Preset Material Type) Namespace: .../drawingml/2006/main	Defines the preset material which is combined with the lighting properties to give the final look and feel of a shape. [*Example*: Consider the following example of a prstMaterial in use within the sp3d element: <a:sp3d extrusionH="165100" contourW="50800" prstMaterial="plastic"> <a:bevelT w="254000" h="254000"/> <a:bevelB w="254000" h="254000"/> <a:extrusionClr> <a:srgbClr val="FF0000"/> </a:extrusionClr> <a:contourClr> <a:schemeClr val="accent3"/> </a:contourClr> </a:sp3d> In this example, we see a prstMaterial defined as plastic. *end example*] The possible values for this attribute are defined by the ST_PresetMaterialType simple type (§5.1.12.50).
z (Shape Depth) Namespace: .../drawingml/2006/main	Defines the z coordinate for the 3D shape. The possible values for this attribute are defined by the ST_Coordinate simple type (§5.1.12.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Shape3D">
   <sequence>
      <element name="bevelT" type="CT_Bevel" minOccurs="0" maxOccurs="1"/>
      <element name="bevelB" type="CT_Bevel" minOccurs="0" maxOccurs="1"/>
      <element name="extrusionClr" type="CT_Color" minOccurs="0" maxOccurs="1"/>
      <element name="contourClr" type="CT_Color" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="z" type="ST_Coordinate" use="optional" default="0"/>
   <attribute name="extrusionH" type="ST_PositiveCoordinate" use="optional" default="0"/>
   <attribute name="contourW" type="ST_PositiveCoordinate" use="optional" default="0"/>
   <attribute name="prstMaterial" type="ST_PresetMaterialType" use="optional" default="warmMatte"/>
</complexType>
```

5.9.5.7 styleDef (Style Definition)

This element is the root tag for a style definition.

[*Example*: Consider the following example of a styleDef in DiagramML:

```
<dgm:styleDef
xmlns:dgm="http://schemas.openxmlformats.org/drawingml/2006/3/diagram"
xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/3/main"
uniqueId="urn:microsoft.com/office/officeart/2005/8/quickstyle/3d1"
minVer="12.0">
    <dgm:title lang="" val="3-D Style 1"/>
    <dgm:desc lang="" val="3-D Style 1"/>
    <dgm:catLst>
          <dgm:cat type="3D" pri="11100"/>
    </dgm:catLst>
    <dgm:scene3d>
          <a:camera prst="orthographicFront"/>
          <a:lightRig rig="threePt" dir="t"/>
    </dgm:scene3d>
    <dgm:style>
                    ...
    </dgm:style>
    <dgm:styleLbl name="node0">
          <dgm:scene3d>
                <a:camera prst="orthographicFront"/>
                <a:lightRig rig="flat" dir="t"/>
          </dgm:scene3d>
          <dgm:sp3d prstMaterial="flat">
                <a:bevelT w="120900" h="88900"/>
                <a:bevelB w="88900" h="31750" prst="angle"/>
          </dgm:sp3d>
```

```
            <dgm:txPr/>
            <dgm:style>
                  <a:lnRef idx="0">
                        <a:scrgbClr r="0" g="0" b="0"/>
                  </a:lnRef>
                  <a:fillRef idx="3">
                        <a:scrgbClr r="0" g="0" b="0"/>
                  </a:fillRef>
                  <a:effectRef idx="2">
                        <a:scrgbClr r="0" g="0" b="0"/>
                  </a:effectRef>
                  <a:fontRef idx="minor">
                        <a:schemeClr val="lt1"/>
                  </a:fontRef>
            </dgm:style>
      </dgm:styleLbl>
...
</styleDef>
```

In this example we see a styleDef being defined along with many properties. *end example*]

Parent Elements
Root element of DrawingML Diagram Style part

Child Elements	Subclause
catLst (Category List)	§5.9.5.2
desc (Style Label Description)	§5.9.5.3
extLst (Extension List)	§5.9.2.13
scene3d (3-D Scene)	§5.9.5.5
styleLbl (Style Label)	§5.9.5.10
title (Title)	§5.9.5.11

Attributes	Description

minVer (Minimum Version)	The minimum product version that can support this quick style.
	
	[*Example*: Consider the following example of a styleDef in DiagramML:
	
	`<styleDef uniqueId="urn:quickstyle/3d1" minVer="12.0">`
	
	`...`

Attributes	Description
	</styleDef> In this example we see the minVer defined to be 12.0. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
uniqueId (Unique Style ID)	Unique ID that identifies a style. [*Example*: Consider the following example of a styleDef in DiagramML: <styleDef uniqueId="urn:quickstyle/3d1" minVer="12.0"> ... </styleDef> In this example we see the uniqueId defined to be urn:quickstyle/3d1. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_StyleDefinition">
   <sequence>
      <element name="title" type="CT_SDName" minOccurs="0" maxOccurs="unbounded"/>
      <element name="desc" type="CT_SDDescription" minOccurs="0" maxOccurs="unbounded"/>
      <element name="catLst" type="CT_SDCategories" minOccurs="0"/>
      <element name="scene3d" type="a:CT_Scene3D" minOccurs="0" maxOccurs="1"/>
      <element name="styleLbl" type="CT_StyleLabel" minOccurs="1" maxOccurs="unbounded"/>
      <element name="extLst" type="a:CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="uniqueId" type="xsd:string" use="optional" default=""/>
   <attribute name="minVer" type="xsd:string" use="optional"
     default="http://schemas.openxmlformats.org/drawingml/2006/diagram"/>
</complexType>
```

5.9.5.8 styleDefHdr (Style Definition Header)

This element specifies header information associated with a style definition. The header information is used by an application to preprocess required data in order to help with possible performance concerns associated with an initial full load of a color transform definition.

[*Example*: Consider the following example of a styleDefHdr element within DiagramML:

```
<styleDefHdr uniqueId="urn:quicktyle/3d1">
    <title val="3D" />
```

```
      <desc val="" />
      <catLst>
          <cat type="3D" pri="10100" />
      </catLst>
  </ styleDefHdr >
```

In this example we see a style definition header which defines a title and category for a set of style definitions. *end example*]

Parent Elements
styleDefHdrLst (§5.9.5.9)

Child Elements	Subclause
catLst (Category List)	§5.9.5.2
desc (Style Label Description)	§5.9.5.3
extLst (Extension List)	§5.9.2.13
title (Title)	§5.9.5.11

Attributes	Description
minVer (Minimum Version)	The minimum product version that can support this quick style. The possible values for this attribute are defined by the XML Schema string datatype.
resId (Resource ID)	This attribute is the id which associates this header to the actual style definition part. The possible values for this attribute are defined by the XML Schema int datatype.
uniqueId (Unique Style ID)	This attribute defines a unique identifier for the associated style definition. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_StyleDefinitionHeader">
   <sequence>
      <element name="title" type="CT_SDName" minOccurs="1" maxOccurs="unbounded"/>
      <element name="desc" type="CT_SDDescription" minOccurs="1" maxOccurs="unbounded"/>
      <element name="catLst" type="CT_SDCategories" minOccurs="0"/>
      <element name="extLst" type="a:CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="uniqueId" type="xsd:string" use="required"/>
   <attribute name="minVer" type="xsd:string" use="optional"
     default="http://schemas.openxmlformats.org/drawingml/2006/diagram"/>
   <attribute name="resId" type="xsd:int" use="optional" default="0"/>
</complexType>
```

5.9.5.9 styleDefHdrLst (List of Style Definition Headers)

This element is simply a list of style definition headers and is used to consolidate multiple headers into one group.

Child Elements	Subclause
styleDefHdr (Style Definition Header)	§5.9.5.8

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_StyleDefinitionHeaderLst">
   <sequence>
      <element name="styleDefHdr" type="CT_StyleDefinitionHeader" minOccurs="0"
        maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

5.9.5.10 styleLbl (Style Label)

This element defines the actual style which is applied to a node in a diagram The style is referenced from within layout node. The style label contains formatting (without defining color) such as the 3D properties and text properties associated with a shape.

[*Example*: Consider the following example of a styleLbl in DiagramML:

```
<styleLbl name="node0">
   <scene3d>
         <camera prst="orthographicFront"/>
         <lightRig rig="flat" dir="t"/>
   </scene3d>
   <sp3d prstMaterial="flat">
         <bevelT w="120900" h="88900"/>
         <bevelB w="88900" h="31750" prst="angle"/>
   </sp3d>
   <txPr/>
   <style>
         <lnRef idx="0">
               <scrgbClr r="0" g="0" b="0"/>
         </lnRef>
         <fillRef idx="3">
               <scrgbClr r="0" g="0" b="0"/>
         </fillRef>
         <effectRef idx="2">
               <scrgbClr r="0" g="0" b="0"/>
         </effectRef>
         <fontRef idx="minor">
               <schemeClr val="lt1"/>
```

```
            </fontRef>
        </style>
    </styleLbl>
```

In this example we see a styleLbl defined which sets 3D properties for the scene, shape 3D properties, line, fill, effect and font properties. *end example*]

Parent Elements
styleDef (§5.9.5.7)

Child Elements	Subclause
extLst (Extension List)	§5.9.2.13
scene3d (3-D Scene)	§5.9.5.5
sp3d (3-D Shape Properties)	§5.9.5.6
style (Shape Style)	§5.9.2.28
txPr (Text Properties)	§5.9.5.12

Attributes	Description
name (Style Name)	The name of the style. This apears as the tooltip in the user interface. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_StyleLabel">
   <sequence>
      <element name="scene3d" type="a:CT_Scene3D" minOccurs="0" maxOccurs="1"/>
      <element name="sp3d" type="a:CT_Shape3D" minOccurs="0" maxOccurs="1"/>
      <element name="txPr" type="CT_TextProps" minOccurs="0" maxOccurs="1"/>
      <element name="style" type="a:CT_ShapeStyle" minOccurs="0" maxOccurs="1"/>
      <element name="extLst" type="a:CT_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>
   </sequence>
   <attribute name="name" type="xsd:string" use="required"/>
</complexType>
```

5.9.5.11 title (Title)

This element defines the title given to a style definition header. The title is simply a name for the style definition.

[*Example*: Consider the following example of title being used in DiagramML:

```
<styleDefHdr uniqueId="urn:quickstyle/3d1" minVer="12.0">
     <title val="My Title"/>
```

```
        <desc val="My Description"/>

        ...

    </styleDefHdr>
```

In this example we see the title being set to `My Title`. *end example*]

Parent Elements
styleDef (§5.9.5.7); styleDefHdr (§5.9.5.8)

Attributes	Description
lang (Natural Language)	The natural language of the title or description of this quick style. The possible values for this attribute are defined by the XML Schema string datatype.
val (Description Value)	The string used for the description. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SDName">
   <attribute name="lang" type="xsd:string" use="optional" default=""/>
   <attribute name="val" type="xsd:string" use="required"/>
</complexType>
```

5.9.5.12 txPr (Text Properties)

This element defines special text formatting that can be applied to text through a style label.

Parent Elements
styleLbl (§5.9.5.10)

Child Elements	Subclause
flatTx (No text in 3D scene)	§5.1.7.8
sp3d (Apply 3D shape properties)	§5.1.7.12

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextProps">
   <sequence>
      <group ref="a:EG_Text3D" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

5.9.6 Layout Definition

This section specifies the node layout information to be associated with a diagram.

5.9.6.1 animLvl (Level Animation)

This variable is used to indicate the animate by level string which is displayed to a user in the user interface.

[*Example*: Consider the following example of animLvl in use in DiagramML:

```
<varLst>
     <chMax val="1" />
     <dir val="norm" />
     <animLvl val="ctr" />
     <resizeHandles val="exact" />
</varLst>
```

In this example we see that the animLvl is set to `ctr`. This is being defined in a radial type diagram which will allow the user to specify that animation is to start at the center of the diagram. *end example*]

Parent Elements
presLayoutVars (§5.9.5.4); varLst (§5.9.2.31)

Attributes	Description
val (Level Animation Value)	This attribute indicates the string to use for level animation in the user interface. The possible values for this attribute are defined by the ST_AnimLvlStr simple type (§5.9.7.2).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AnimLvl">
   <attribute name="val" type="ST_AnimLvlStr" default="none" use="optional"/>
</complexType>
```

5.9.6.2 animOne (One by One Animation String)

This variable is used to indicate the string to use for one-by-one animation in the user interface. This is used primarily when defining hierarchical diagrams to specify different ways animations applies to different levels of the diagram.

[*Example*: Consider the following example of animOne used in a hierarchical type diagram:

```
<varLst>
     <chPref val="1" />
     <dir val="norm" />
     <animOne val="branch" />
```

```
        <animLvl val="lvl" />
        <resizeHandles val="exact" />
    </varLst>
```

In this example we see that the animOne element is defined to animate the diagram per branch. *end example*]

Parent Elements
presLayoutVars (§5.9.5.4); varLst (§5.9.2.31)

Attributes	Description
val (One By One Animation Value)	Specifies the type of one-by-one animation to use for a diagram. The possible values for this attribute are defined by the ST_AnimOneStr simple type (§5.9.7.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AnimOne">
   <attribute name="val" type="ST_AnimOneStr" default="one" use="optional"/>
</complexType>
```

5.9.6.3 bulletEnabled (Show Insert Bullet)

This element is used to indicate when to enable the 'Insert Bullet' button in the user interface.

[*Example*: Consider the following example of bulletEnabled in DiagramML:

```
<varLst>
    <bulletEnabled val="true" />
</varLst>
```

In this example we see that the insert button in the user interface is to be enabled when the focus is within the containing layout node. *end example*]

Parent Elements
presLayoutVars (§5.9.5.4); varLst (§5.9.2.31)

Attributes	Description
val (Show Insert Bullet Value)	This attribute is used to indicate when to enable the 'Insert Bullet' button. A value of `true` will enable the insert bullet button. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BulletEnabled">
   <attribute name="val" type="xsd:boolean" default="false" use="optional"/>
</complexType>
```

5.9.6.4 chMax (Maximum Children)

This element is used to indicate when to enable and disable the user interface components associated with adding a new shape to a diagram. This element defines a max number of nodes a diagram can support through the user interface directly.

[*Example*: Consider the following example of chMax usage in DiagramML:

```
<varLst>
    <chMax val="5"/>
    <dir val="norm"/>
    <resizeHandles val="exact" />
</varLst>
```

In this example we define the user interface to only be enabled to insert five nodes. *end example*]

Parent Elements
presLayoutVars (§5.9.5.4); varLst (§5.9.2.31)

Attributes	Description
val (Maximum Children Value)	This attribute indicates the maximum number of children the node can have before the user interface should be disabled. A value of -1 indicates an infinite number of children. Default value is -1. The possible values for this attribute are defined by the ST_NodeCount simple type (§5.9.7.43).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ChildMax">
   <attribute name="val" type="ST_NodeCount" default="-1" use="optional"/>
</complexType>
```

5.9.6.5 chPref (Preferred Number of Children)

This variable indicates the number of children that the current node prefers to have. This determines what the next action of the 'Add Shape' button should be in the user interface.

[*Example*: Consider the following example of chPref being used in DiagramML:

```
<varLst>
    <chMax val="3" />
```

```
      <chPref val="1" />
      <dir val="norm" />
      <animLvl val="lvl" />
      <resizeHandles val="rel" />
   </varLst>
```

In this example, chPref is set to a single node and the user interface will disable after a single node has been inserted. *end example*]

Parent Elements
presLayoutVars (§5.9.5.4); varLst (§5.9.2.31)

Attributes	Description
val (Preferred Number of CHildren Value)	This attribute indicates the number of children that the current node prefers to have. This determines what the next action of the 'Add Shape' button should be. A value of -1 indicates an infinite number of children. Default value is -1. The possible values for this attribute are defined by the ST_NodeCount simple type (§5.9.7.43).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ChildPref">
   <attribute name="val" type="ST_NodeCount" default="-1" use="optional"/>
</complexType>
```

5.9.6.6 dir (Diagram Direction)

This element indicates whether the diagram should switch direction. This element provides the ability to define different behavior for diagrams considering LTR or RTL directions.

Parent Elements
presLayoutVars (§5.9.5.4); varLst (§5.9.2.31)

Attributes	Description
val (Diagram Direction Value)	This variable indicates whether the diagram should switch direction. The possible values for this attribute are defined by the ST_Direction simple type (§5.9.7.24).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Direction">
   <attribute name="val" type="ST_Direction" default="norm" use="optional"/>
</complexType>
```

5.9.6.7 hierBranch (Organization Chart Branch Style)

This element defines the layout style of a branch in an organizational chart.

[*Example*: Consider the following example of hierBranch being used in DiagramML:

```
<varLst>
    <hierBranch val="init" />
</varLst>
```

In this example the value of hierBranch is defined as `init` which will be a kind of not set state, or initial state. *end example*]

Parent Elements
presLayoutVars (§5.9.5.4); varLst (§5.9.2.31)

Attributes	Description
val (Organization Chart Branch Style Value)	The value of this attribute indicates the layout style of a branch in an organization chart. The default value is `std`. The possible values for this attribute are defined by the ST_HierBranchStyle simple type (§5.9.7.35).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_HierBranchStyle">
   <attribute name="val" type="ST_HierBranchStyle" default="std" use="optional"/>
</complexType>
```

5.9.6.8 orgChart (Show Organization Chart User Interface)

This element is used to indicate when to show user interface controls specifically associated with organizational charts such as being able to add an assistant to a selected node.

[*Example*: Consider the following example of orgChart used in DiagramML:

```
<varLst>
    <orgChart val="true" />
    <chPref val="1" />
    <dir val="norm" />
    <animOne val="branch" />
    <animLvl val="lvl" />
    <resizeHandles val="rel" />
</varLst>
```

In this example we set the orgChart value to `true` indicating that the organizational chart specific user interface controls are to be enabled when the containing diagram is used. *end example*]

Parent Elements
presLayoutVars (§5.9.5.4); varLst (§5.9.2.31)

Attributes	Description
val (Show Organization Chart User Interface Value)	This attribute value specifies when to show the 'Insert Assistant' user interface control and the 'Change Layout' user interface for this diagram. The possible values for this attribute are defined by the XML Schema boolean datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OrgChart">
   <attribute name="val" type="xsd:boolean" default="false" use="optional"/>
</complexType>
```

5.9.7 Simple Types

This is the complete list of simple types in the http://schemas.openxmlformats.org/drawingml/2006/diagram namespace.

5.9.7.1 ST_AlgorithmType (Algorithm Types)

Types of available algorithms.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
composite (Composite)	The composite algorithm specifies the size and position for all child layout nodes. You can use it to create graphics with a predetermined layout or in combination with other algorithms to create more complex shapes.
conn (Connector Algorithm)	The connector algorithm lays out and routes connecting lines, arrows, and shapes between layout nodes.
cycle (Cycle Algorithm)	The cycle algorithm lays out child layout nodes around a circle or portion of a circle using equal angle spacing.
hierChild (Hierarchy Child Algorithm)	The hierarchy child algorithm works with the hierRoot algorithm to create hierarchical tree layouts. This algorithm aligns and positions its child layout nodes in a linear path under the hierRoot layout node.
hierRoot (Hierarchy Root Algorithm)	The hierarchy root algorithm works with the hierChild algorithm to create hierarchical tree layouts. The hierRoot algorithm aligns and positions the hierRoot

Enumeration Value	Description
	layout node in relation to the hierChild layout nodes.
lin (Linear Algorithm)	The linear algorithm lays out child layout nodes along a linear path.
pyra (Pyramid Algorithm)	The pyramid algorithm lays out child layout nodes along a vertical path and works with the trapezoid shape to create a pyramid.
snake (Snake Algorithm)	The snake algorithm lays out child layout nodes along a linear path in two dimensions, allowing the linear flow to continue across multiple rows or columns.
sp (Space Algorithm)	The space algorithm is used to specify a minimum space between other layout nodes or as an indication to do nothing with the layout node's size and position.
tx (Text Algorithm)	The text algorithm sizes text to fit inside a shape and controls its margins and alignment.

Referenced By
alg@type (§5.9.2.3)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_AlgorithmType" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="composite"/>
      <enumeration value="conn"/>
      <enumeration value="cycle"/>
      <enumeration value="hierChild"/>
      <enumeration value="hierRoot"/>
      <enumeration value="pyra"/>
      <enumeration value="lin"/>
      <enumeration value="sp"/>
      <enumeration value="tx"/>
      <enumeration value="snake"/>
   </restriction>
</simpleType>
```

5.9.7.2 ST_AnimLvlStr (Animation Level String Definition)

This simple type specifies the possible values for the string that should be displayed by a consumer for level animation of this diagram.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description

Enumeration Value	Description
ctr (From Center Animation)	This value specifies that the consumer shall allow "From Center At Once" or "From Center One by One" animation styles for this diagram.
lvl (By Level Animation)	This value specifies that the consumer shall display "By Level" animation types for this diagram.
none (Disable Level At Once)	This value specifies that the consumer shall disable level at once animation.

Referenced By
animLvl@val (§5.9.6.1); ST_FunctionValue (§5.9.7.32)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_AnimLvlStr" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="none"/>
      <enumeration value="lvl"/>
      <enumeration value="ctr"/>
   </restriction>
</simpleType>
```

5.9.7.3 ST_AnimOneStr (One by One Animation Value Definition)

This simple type defines the possible values for the string to use for one by one animation in the UI. Default value is one.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
branch (By Branch One By One)	This value specifies that the one by one animation string in the user interface should read "By Branch One By One".
none (Disable One-by-One)	This value specifies that the consumer should disable one by one animation.
one (One By One)	This value specifies that the one by one animation string in the user interface should read "One By One".

Referenced By
animOne@val (§5.9.6.2); ST_FunctionValue (§5.9.7.32)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_AnimOneStr" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="none"/>
      <enumeration value="one"/>
      <enumeration value="branch"/>
   </restriction>
</simpleType>
```

5.9.7.4 ST_ArrowheadStyle (Arrowhead Styles)

This simple type defines different arrowhead style types for connectors.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
arr (Arrowhead Present)	Specifies that an arrowhead is to be used on the connector.
auto (Auto)	Specifies that the algorithm defines if an arrowhead is to be used on a connector.
noArr (No Arrowhead)	Specifies no arrowhead is to be used on the connector.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ArrowheadStyle" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="auto"/>
      <enumeration value="arr"/>
      <enumeration value="noArr"/>
   </restriction>
</simpleType>
```

5.9.7.5 ST_AutoTextRotation (Auto Text Rotation)

This simple type defines how text rotates within a shape when the shape is rotated by an algorithm during layout.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
grav (Gravity)	Specifies that when the angle of the text hits the

Enumeration Value	Description
	threshold of 90 degrees and 180 degrees, the text rotates by 180 degrees.
none (None)	Specifies that text always rotates with the shape.
upr (Upright)	Specifies that when the text angle hits 45, 135, 225, or 315 degree thresholds, then it rotates by negative 90 degrees.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

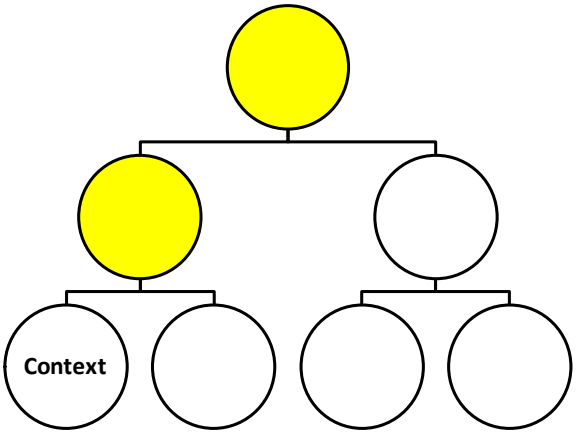
```
<simpleType name="ST_AutoTextRotation" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="none"/>
      <enumeration value="upr"/>
      <enumeration value="grav"/>
   </restriction>
</simpleType>
```

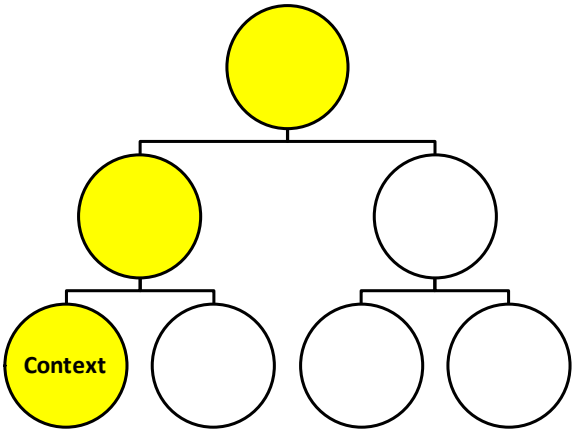
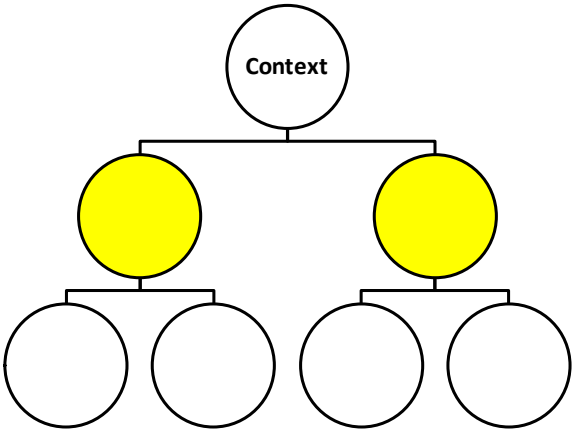
5.9.7.6 ST_AxisType (Axis Type)

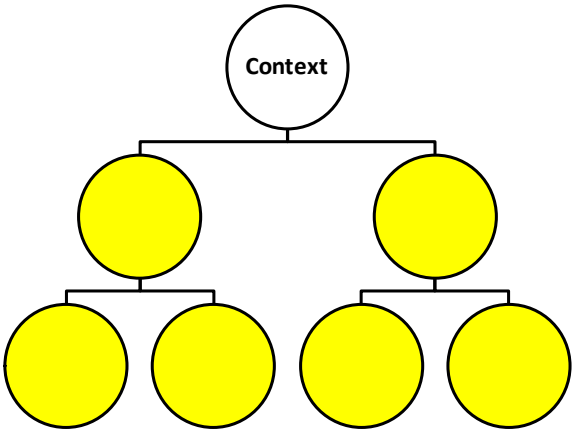
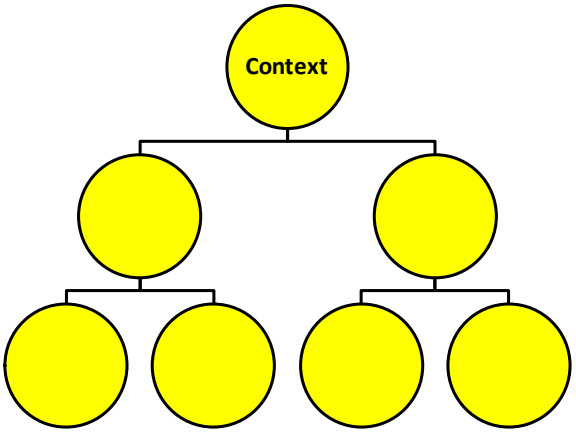
This simple type defines different node sets in relation to the current context node.

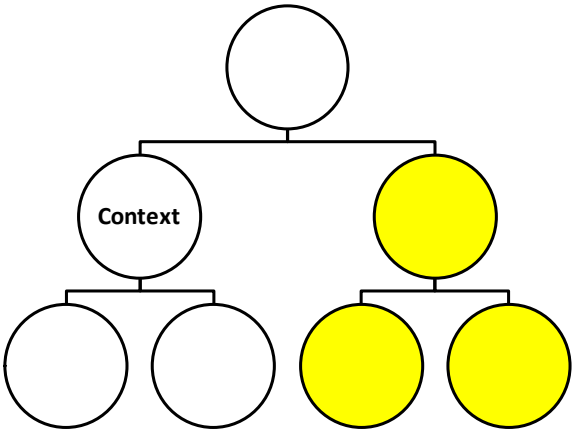
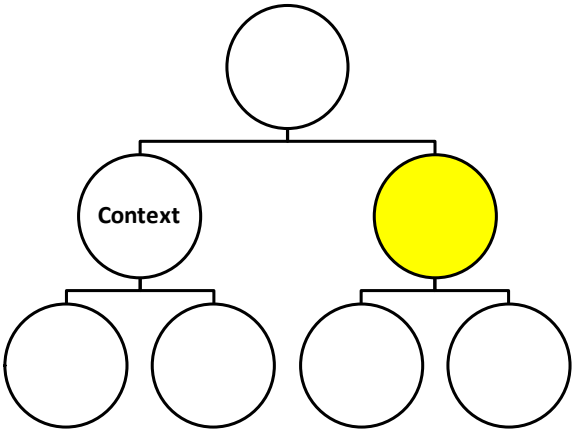
This simple type's contents are a restriction of the XML Schema token datatype.

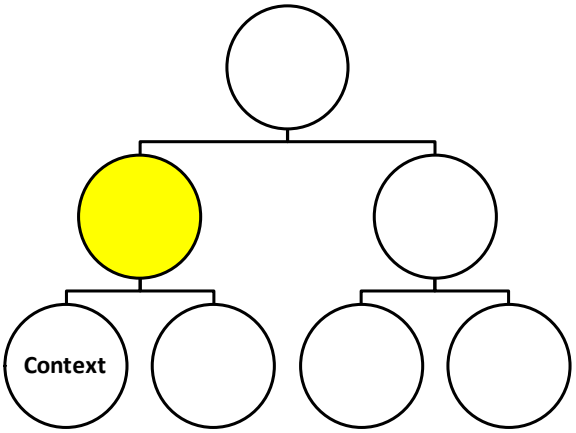
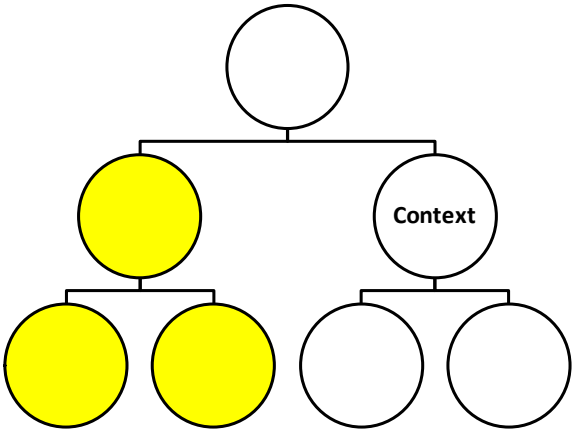
The following are possible enumeration values for this type:

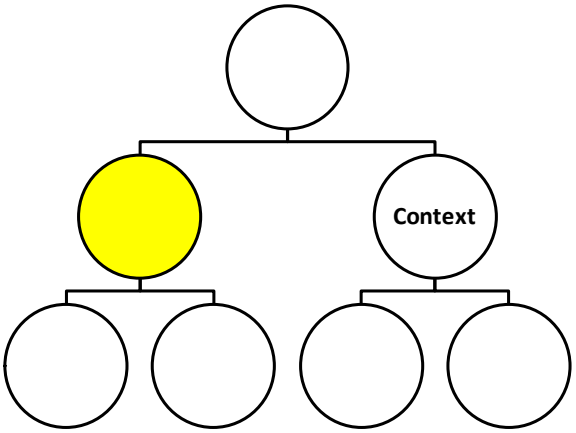
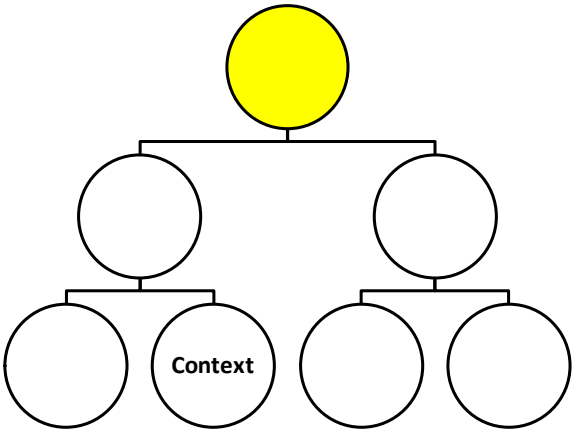
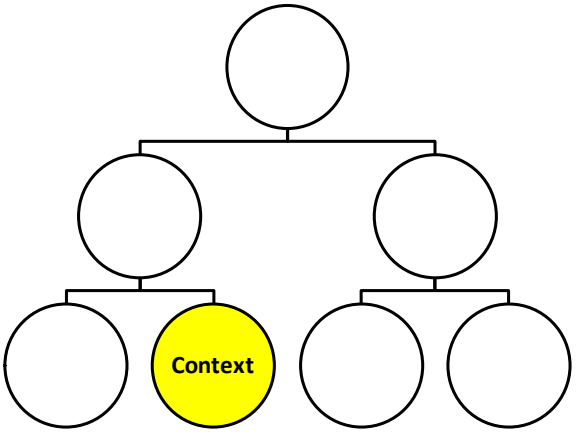
Enumeration Value	Description
ancst (Ancestor)	Specifies a set of nodes between the current context node and the root node, including the root node. [*Example*: Consider the following example 

Enumeration Value	Description
	end example]
ancstOrSelf (Ancestor or Self)	Specifies a set of nodes between the current context node and the root node, including the root node and the context node. [*Example*: Consider the following example  *end example*]
ch (Child)	Specifies a set of children of the current context node. [*Example*: Consider the following example  *end example*]
des (Descendant)	Specifies a set of all nodes beneath the current context node. [*Example*: Consider the following example

Enumeration Value	Description
	 *end example*]
desOrSelf (Descendant or Self)	Specifies a set of all nodes beneath the current context node, including the context node. [*Example*: Consider the following example  *end example*]
follow (Follow)	Specifies the set of nodes which are peers after the context node and all descendants of the peers. [*Example*: Consider the following example

Enumeration Value	Description
	 *end example*]
followSib (Follow Sibling)	Specifies the set of nodes which are peers after the context node. [*Example*: Consider the following example  *end example*]
none (None)	Specifies no node.
par (Parent)	Specifies the parent node. [*Example*: Consider the following example

Enumeration Value	Description
	 *end example*]
preced (Preceding)	Specifies the set of nodes which are peers before the context node and all the descendants of the peers. [*Example*: Consider the following example  *end example*]
precedSib (Preceding Sibling)	Specifies the set of nodes which are peers before the context node. [*Example*: Consider the following example

Enumeration Value	Description
	 *end example*]
root (Root)	Specifies the top-most node of the diagram. [*Example*: Consider the following example  *end example*]
self (Self)	Specifies the calling context node. [*Example*: Consider the following example  *end example*]

Referenced By
ST_AxisType (§5.9.7.6)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_AxisType" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="self"/>
      <enumeration value="ch"/>
      <enumeration value="des"/>
      <enumeration value="desOrSelf"/>
      <enumeration value="par"/>
      <enumeration value="ancst"/>
      <enumeration value="ancstOrSelf"/>
      <enumeration value="followSib"/>
      <enumeration value="precedSib"/>
      <enumeration value="follow"/>
      <enumeration value="preced"/>
      <enumeration value="root"/>
      <enumeration value="none"/>
   </restriction>
</simpleType>
```

5.9.7.7 ST_AxisTypes (Axis Type List)

This simple type represents a list of axis types.

This simple type allows a list of items of the ST_AxisType simple type (§5.9.7.6).

Referenced By
forEach@axis (§5.9.2.14); if@axis (§5.9.2.15); presOf@axis (§5.9.2.21)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_AxisTypes">
   <list itemType="ST_AxisType"/>
</simpleType>
```

5.9.7.8 ST_BendPoint (Bend Point)

This simple type defines where a bend is to occur within a connection between two nodes.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
beg (Beginning)	The bend is to occur at the beginning of the connection.

Enumeration Value	Description
def (Default)	The default bend is used. By default connections will bend in the center.
end (End)	The bend is to occur at the end of the connection.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_BendPoint" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="beg"/>
      <enumeration value="def"/>
      <enumeration value="end"/>
   </restriction>
</simpleType>
```

5.9.7.9 ST_Booleans (Boolean List.)

A list of booleans.

This simple type allows a list of items of the XML Schema boolean datatype.

Referenced By
forEach@hideLastTrans (§5.9.2.14); if@hideLastTrans (§5.9.2.15); presOf@hideLastTrans (§5.9.2.21)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Booleans">
   <list itemType="xsd:boolean"/>
</simpleType>
```

5.9.7.10 ST_BoolOperator (Boolean Constraint)

This simple type specified Boolean operations which can be applied to compare constraints.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
equ (Equal)	Equal operator.
gte (Greater Than or Equal to)	Specifies the greater than or equal to Boolean operator.
lte (Less Than or Equal to)	Specifies the less than or equal to Boolean operator.
none (None)	Specifies a none Boolean operator

Referenced By
constr@op (§5.9.2.8)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_BoolOperator" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="none"/>
      <enumeration value="equ"/>
      <enumeration value="gte"/>
      <enumeration value="lte"/>
   </restriction>
</simpleType>
```

5.9.7.11 ST_Breakpoint (Breakpoint)

This simple type defines at what point the wrapping of nodes occurs for the snake algorithm.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
bal (Balanced)	Specifies that the number of nodes in every row and every column should be equal.
endCnv (End of Canvas)	Specifies that nodes are added to the next column or row after filling the current column or row's space.
fixed (Fixed)	Specifies to use a user defined number of nodes in a column or row.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Breakpoint" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="endCnv"/>
      <enumeration value="bal"/>
      <enumeration value="fixed"/>
   </restriction>
</simpleType>
```

5.9.7.12 ST_CenterShapeMapping (Center Shape Mapping)

This simple type defines the behavior of the cycle algorithm.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
fNode (First Node)	Specifies a node which is always in the center of a cycle diagram.
none (None)	Specifies the normal layout of a cycle diagram.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_CenterShapeMapping" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="none"/>
      <enumeration value="fNode"/>
   </restriction>
</simpleType>
```

5.9.7.13 ST_ChildAlignment (Child Alignment)

This simple type defines how to align a node in its allocated space.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
b (Bottom)	Specifies to align the node to the bottom.
l (Left)	Specifies to align the node to the left.
r (Right)	Specifies to align the node to the right.
t (Top)	Specifies to align the node to the top.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ChildAlignment" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="t"/>
      <enumeration value="b"/>
      <enumeration value="l"/>
      <enumeration value="r"/>
   </restriction>
</simpleType>
```

5.9.7.14 ST_ChildDirection (Child Direction)

This simple type defines the layout direction of child nodes related to a specific parent node.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
horz (Horizontal)	Specifies that the child nodes are to be laid out horizontally.
vert (Vertical)	Specifies that the child nodes are to be laid out vertically.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ChildDirection" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="horz"/>
      <enumeration value="vert"/>
   </restriction>
</simpleType>
```

5.9.7.15 ST_ChildOrderType (Child Order)

This simple type specifies the child order for a given layout node.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
b (Bottom)	Child order along the bottom.
t (Top)	Top child order.

Referenced By
layoutNode@chOrder (§5.9.2.19)

The following XML Schema fragment defines the contents of this simple type:

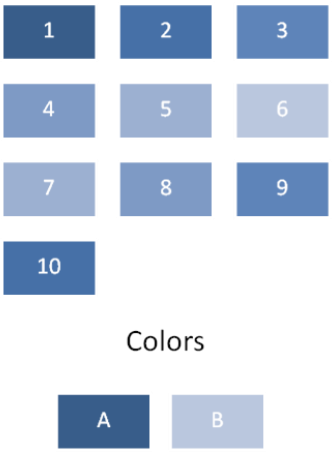
```
<simpleType name="ST_ChildOrderType" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="b"/>
      <enumeration value="t"/>
   </restriction>
</simpleType>
```

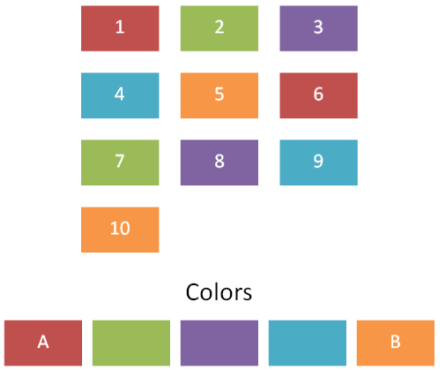
5.9.7.16 ST_ClrAppMethod (Color Application Method Type)

This type defines the way a given set of colors is applied to a set of nodes or items across a diagram.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
cycle (Cycle)	The colors will apply from A to B to A if A and B were the colors present. [*Example*: Consider the following image as an example of cycle applied to a diagram:  In this example, the color A is applied to node 1 and node 10. Color B is considered the node color between A and A across the diagram. Colors interpolate across the diagram from A to B back to A. *end example*]
repeat (Repeat)	The colors will apply from A through B to A through B if A through B were the colors present.

Enumeration Value	Description
	[*Example*: Consider the following image as an example of repeat applied to a diagram:  In this example, the color A is applied to node 1, the next color to node 2, and so on through color B, then this coloring is repeated until there are no more nodes to color. *end example*]
span (Span)	The colors will interpolate from A to B across the entire diagram if A and B were the colors present. [*Example*: Consider the following image as an example of span applied to a diagram:  In this example, the color A is applied to node 1, the color B is applied to node 10 and the colors applied to nodes 2 through 9 are interpolated between colors A and B. *end example*]

Referenced By
effectClrLst@meth (§5.9.4.7); fillClrLst@meth (§5.9.4.8); linClrLst@meth (§5.9.4.9); txEffectClrLst@meth

Referenced By
(§5.9.4.12); txFillClrLst@meth (§5.9.4.13); txLinClrLst@meth (§5.9.4.14)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ClrAppMethod">
   <restriction base="xsd:token">
      <enumeration value="span"/>
      <enumeration value="cycle"/>
      <enumeration value="repeat"/>
   </restriction>
</simpleType>
```

5.9.7.17 ST_ConnectorDimension (Connector Dimension)

This simple type defines the dimensionality of the connection between two nodes.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
1D (1 Dimension)	Specifies a one dimensional connection, or rather a line.
2D (2 Dimensions)	Specifies a two dimensional connection which has both width and height.
cust (Custom)	Specifies a custom connection type.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ConnectorDimension" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="1D"/>
      <enumeration value="2D"/>
      <enumeration value="cust"/>
   </restriction>
</simpleType>
```

5.9.7.18 ST_ConnectorPoint (Connector Point)

This simple type defines different connection sites available on a node.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
auto (Auto)	Specifies that the algorithm will determine the best connection site to use.
bCtr (Bottom Center)	Specifies that the bottom, center connection site is to be used.
bL (Bottom Left)	Specifies that the bottom, left connection site is to be used.
bR (Bottom Right)	Specifies that the bottom right connection site is to be used.
ctr (Center)	Specifies that the center connection site is to be used.
midL (Middle Left)	Specifies that the middle left connection site is to be used.
midR (Middle Right)	Specifies that the middle right connection site is to be used.
radial (Radial)	Specifies connections along a radial path to support the use of connections in cycle diagrams.
tCtr (Top Center)	Specifies that the top center connection site is to be used.
tL (Top Left)	Specifies that the top left connection site is to be used.
tR (Top Right)	Specifies that the top right connection site is to be used.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ConnectorPoint" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="auto"/>
      <enumeration value="bCtr"/>
      <enumeration value="ctr"/>
      <enumeration value="midL"/>
      <enumeration value="midR"/>
      <enumeration value="tCtr"/>
      <enumeration value="bL"/>
      <enumeration value="bR"/>
      <enumeration value="tL"/>
      <enumeration value="tR"/>
      <enumeration value="radial"/>
   </restriction>
</simpleType>
```

5.9.7.19 ST_ConnectorRouting (Connector Routing)

This simple type defines how the routing of a connection between two nodes is supposed to progress from node 1 to node 2.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
bend (Bending)	Specifies a bending connection which bends at a right angle.
curve (Curve)	Specifies a connection which is curved.
longCurve (Long Curve)	Specifies a connection that is curved that has a greater radius than a simple curved connection.
stra (Straight)	Specifies a straight connection.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ConnectorRouting" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="stra"/>
      <enumeration value="bend"/>
      <enumeration value="curve"/>
      <enumeration value="longCurve"/>
   </restriction>
</simpleType>
```

5.9.7.20 ST_ConstraintRelationship (Constraint Relationship)

This simple type specifies the types of constraint relationships which are present and can be used.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
ch (Child)	The constraint should reference a child node.
des (Descendant)	The layout node can map to the descendants of the data point.
self (Self)	The layout node maps to the current data point.

Referenced By
constr@for (§5.9.2.8); constr@refFor (§5.9.2.8); rule@for (§5.9.2.24)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ConstraintRelationship" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="self"/>
      <enumeration value="ch"/>
      <enumeration value="des"/>
   </restriction>
</simpleType>
```

5.9.7.21 ST_ConstraintType (Constraint Type)

This simple type defines the list of possible constraints available for use.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
alignOff (Alignment Offset)	This value defines the alignment offset for a node.
b (Bottom)	The bottom of the node.
begMarg (Beginning Margin)	Specifies the beginning margin.
begPad (Beginning Padding)	Specifies the beginning padding.
bendDist (Bending Distance)	Specifies the distance from the start of a connector to a bend in the connector.
bMarg (Bottom Margin)	Specifies the bottom margin.
bOff (Bottom Offset)	Specifies the bottom offset.
connDist (Connection Distance)	Specifies the connection distance.
ctrX (Center Height)	Specifies the center of the height.
ctrXOff (Center X Offset)	Specifies the center x coordinate offset.
ctrY (Center Width)	Specifies the center of the width.
ctrYOff (Center Y Offset)	Specifies the center y coordinate offset.
diam (Diameter)	Specifies the diameter.
endMarg (End Margin)	Specifies the ending margin.
endPad (End Padding)	Specifies the end padding.
h (Height)	Specifies the height.
hArH (Arrowhead Height)	Specifies the height of the arrowhead portion of the connector.
hOff (Height Offset)	Specifies the amount to offset the height.

Enumeration Value	Description
l (Left)	Specifies the left constraint.
lMarg (Left Margin)	Specifies the left margin.
lOff (Left Offset)	Specifies the left offset.
none (Unknown)	Unknown constraint.
primFontSz (Primary Font Size)	The primary font size.
pyraAcctRatio (Pyramid Accent Ratio)	Specifies the fraction of the width of the diagram that is reserved for the fly outs at their shortest distance.
r (Right)	Specifies the right constraint.
rMarg (Right Margin)	Specifies the right margin constraint.
rOff (Right Offset)	Specifies the right offset constraint.
secFontSz (Secondary Font Size)	The secondary font size.
secSibSp (Secondary Sibling Spacing)	The secondary sibling spacing.
sibSp (Sibling Spacing)	Specifies the minimum distance between sibling shapes.
sp (Spacing)	Specifies the spacing defined.
stemThick (Stem Thickness)	Specifies the thickness of the arrow's shaft.
t (Top)	Specifies the top constraint.
tMarg (Top Margin)	Top margin constraint.
tOff (Top Offset)	Top offset constraint.
userA (User Defined A)	User defined information.
userB (User Defined B)	User defined information.
userC (User Defined C)	User defined information.
userD (User Defined D)	User defined information.
userE (User Defined E)	User defined information.
userF (User Defined F)	User defined information.
userG (User Defined G)	User defined information.
userH (User Defined H)	User defined information.
userI (User Defined I)	User defined information.
userJ (User Defined J)	User defined information.
userK (User Defined K)	User defined information.
userL (User Defined L)	User defined information.
userM (User Defined M)	User defined information.
userN (User Defined N)	User defined information.
userO (User Defined O)	User defined information.
userP (User Defined P)	User defined information.

Enumeration Value	Description
userQ (User Defined Q)	User defined information.
userR (User Defined R)	User defined information.
userS (User Defined S)	User defined information.
userT (User Defined T)	User defined information.
userU (User Defined U)	User defined information.
userV (User Defined V)	User defined information.
userW (User Defined W)	User defined information.
userX (User Defined X)	User defined information.
userY (User Defined Y)	User defined information.
userZ (User Defined Z)	User defined information.
w (Width)	The width parameter.
wArH (Arrowhead Width)	Specifies the width of the arrowhead portion of the connector.
wOff (Width Offset)	Offsets the width by the specified amount.

Referenced By
constr@refType (§5.9.2.8); constr@type (§5.9.2.8); rule@type (§5.9.2.24)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ConstraintType" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="none"/>
      <enumeration value="alignOff"/>
      <enumeration value="begMarg"/>
      <enumeration value="bendDist"/>
      <enumeration value="begPad"/>
      <enumeration value="b"/>
      <enumeration value="bMarg"/>
      <enumeration value="bOff"/>
      <enumeration value="ctrX"/>
      <enumeration value="ctrXOff"/>
      <enumeration value="ctrY"/>
      <enumeration value="ctrYOff"/>
      <enumeration value="connDist"/>
      <enumeration value="diam"/>
      <enumeration value="endMarg"/>
      <enumeration value="endPad"/>
      <enumeration value="h"/>
      <enumeration value="hArH"/>
      <enumeration value="hOff"/>
      <enumeration value="l"/>
      <enumeration value="lMarg"/>
      <enumeration value="lOff"/>
      <enumeration value="r"/>
      <enumeration value="rMarg"/>
      <enumeration value="rOff"/>
      <enumeration value="primFontSz"/>
      <enumeration value="pyraAcctRatio"/>
      <enumeration value="secFontSz"/>
      <enumeration value="sibSp"/>
      <enumeration value="secSibSp"/>
      <enumeration value="sp"/>
      <enumeration value="stemThick"/>
      <enumeration value="t"/>
      <enumeration value="tMarg"/>
      <enumeration value="tOff"/>
      <enumeration value="userA"/>
      <enumeration value="userB"/>
      <enumeration value="userC"/>
      <enumeration value="userD"/>
      <enumeration value="userE"/>
      <enumeration value="userF"/>
      <enumeration value="userG"/>
      <enumeration value="userH"/>
      <enumeration value="userI"/>
      <enumeration value="userJ"/>
      <enumeration value="userK"/>
      <enumeration value="userL"/>
      <enumeration value="userM"/>
      <enumeration value="userN"/>
      <enumeration value="userO"/>
```

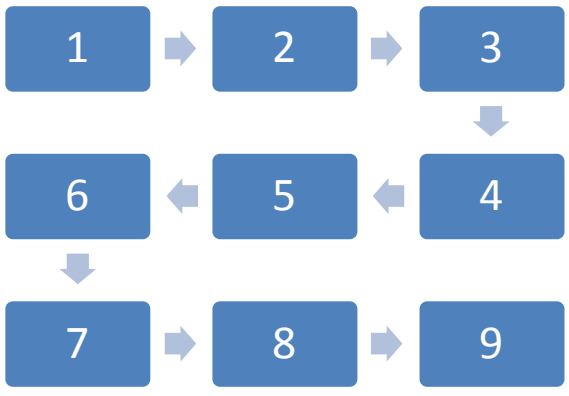
```
        <enumeration value="userP"/>
        <enumeration value="userQ"/>
        <enumeration value="userR"/>
        <enumeration value="userS"/>
        <enumeration value="userT"/>
        <enumeration value="userU"/>
        <enumeration value="userV"/>
        <enumeration value="userW"/>
        <enumeration value="userX"/>
        <enumeration value="userY"/>
        <enumeration value="userZ"/>
        <enumeration value="w"/>
        <enumeration value="wArH"/>
        <enumeration value="wOff"/>
    </restriction>
</simpleType>
```

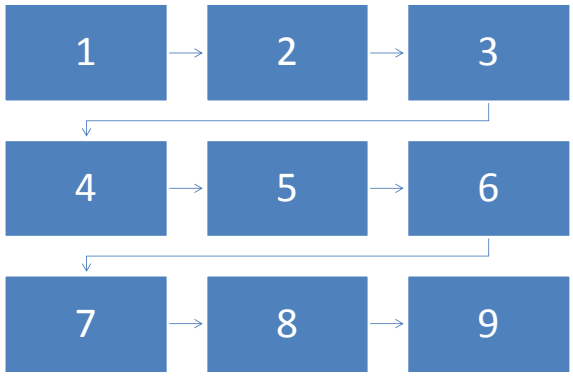
5.9.7.22 ST_ContinueDirection (Continue Direction)

This simple type specifies the behavior of the direction that additional nodes are added to new rows or columns in the snake algorithm.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
revDir (Reverse Direction)	Specifies that the direction is to be revered on a subsequent row or column. [*Example*: Consider the following diagram as an example of reverse direction  *end example*]
sameDir (Same Direction)	Specifies that the direction is to be maintained on a subsequent row or column.

Enumeration Value	Description
	[*Example*: Consider the following diagram as an example of same direction  *end example*]

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ContinueDirection" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="revDir"/>
      <enumeration value="sameDir"/>
   </restriction>
</simpleType>
```

5.9.7.23 ST_CxnType (Connection Type)

This simple type defines the different types of relationships that can be defined between two nodes.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
parOf (Parent Of)	This defines a parent-child relationship in the sense that node X is a parent of node Y.
presOf (Presentation Of)	A presentation type relationship. This type of relationship exists to actually present data.
presParOf (Presentation Parent Of)	A relationship defining a parent of a presentation node.
unknownRelationship (Unknown Relationship)	The type of relationship is unknown.

Referenced By
cxn@type (§5.9.3.2)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_CxnType">
    <restriction base="xsd:token">
        <enumeration value="parOf"/>
        <enumeration value="presOf"/>
        <enumeration value="presParOf"/>
        <enumeration value="unknownRelationship"/>
    </restriction>
</simpleType>
```

5.9.7.24 ST_Direction (Diagram Direction Definition)

This simple type defines the possible values for a diagram's direction when displayed in an application.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
norm (Normal Direction)	This value specifies that the direction of the diagram should not be switched.
rev (Reversed Direction)	This value specifies that the direction of the diagram should be switched.

Referenced By
dir@val (§5.9.6.6); ST_FunctionValue (§5.9.7.32)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Direction" final="restriction">
    <restriction base="xsd:token">
        <enumeration value="norm"/>
        <enumeration value="rev"/>
    </restriction>
</simpleType>
```

5.9.7.25 ST_ElementType (Element Type)

This simple type defines the different types of data points which are supported.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
all (All)	Defined as utilizing all of the nodes.
asst (Assistant)	The assistant nodes.
doc (Document)	Specifies the a node on the document level.
node (Node)	Data nodes that are children of other data nodes.
nonAsst (Non Assistant)	Selects all of the non-assistant nodes.
nonNorm (Non Normal)	Selects the non-normal elements.
norm (Normal)	Selects a normal elements.
parTrans (Parent Transition)	The transition associated with the parent node.
pres (Presentation)	This refers to a presentation node.
sibTrans (Sibling Transition)	Use only sibling transitions between data nodes. These transitions represent sibling relationships between nodes, and are frequently mapped to arrows between shapes in the drawing. A sibTrans value is sometimes used to create white space between nodes.

Referenced By
constr@ptType (§5.9.2.8); constr@refPtType (§5.9.2.8); rule@ptType (§5.9.2.24); ST_ElementType (§5.9.7.25)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ElementType" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="all"/>
      <enumeration value="doc"/>
      <enumeration value="node"/>
      <enumeration value="norm"/>
      <enumeration value="nonNorm"/>
      <enumeration value="asst"/>
      <enumeration value="nonAsst"/>
      <enumeration value="parTrans"/>
      <enumeration value="pres"/>
      <enumeration value="sibTrans"/>
   </restriction>
</simpleType>
```

5.9.7.26 ST_ElementTypes (Element Type List)

A list of element types.

This simple type allows a list of items of the ST_ElementType simple type (§5.9.7.25).

Referenced By

Referenced By
forEach@ptType (§5.9.2.14); if@ptType (§5.9.2.15); presOf@ptType (§5.9.2.21)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ElementTypes">
   <list itemType="ST_ElementType"/>
</simpleType>
```

5.9.7.27 ST_FallbackDimension (Fallback Dimension)

Specifies the dimensionality by which nodes can grow or shrink automatically.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
1D (1 Dimension)	Specifies that the node can grow or shrink by its height or its width, but not both.
2D (2 Dimensions)	Specifies that the node can grow or shrink by both height and width.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FallbackDimension" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="1D"/>
      <enumeration value="2D"/>
   </restriction>
</simpleType>
```

5.9.7.28 ST_FlowDirection (Flow Direction)

This simple type defines how the progression of new nodes are to be entered into the diagram.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
col (Column)	Specifies that the layout occurs in a column-based fashion. This would mean laying out the nodes from top to bottom, before moving left to right.
row (Row)	Specifies that the layout occurs in a row-based fashion.

Enumeration Value	Description
	This would mean laying out the nodes from left to right before moving from top to bottom.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FlowDirection" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="row"/>
      <enumeration value="col"/>
   </restriction>
</simpleType>
```

5.9.7.29 ST_FunctionArgument (Function Argument)

Conditional expression function argument.

This simple type is defined as a union of the following types:

- TheST_VariableType simple type (§5.9.7.64).

Referenced By
if@arg (§5.9.2.15)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FunctionArgument" final="restriction">
   <union memberTypes="ST_VariableType"/>
</simpleType>
```

5.9.7.30 ST_FunctionOperator (Function Operator)

This simple type defines the condition expression functions which can be used to perform operations.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
equ (Equal)	Equal function operator.
gt (Greater Than)	Specifies the grater than function operator.
gte (Greater Than or Equal to)	Specifies the greater than or equal to function operator.
lt (Less Than)	Specifies the less than function operator.

Enumeration Value	Description
lte (Less Than or Equal to)	Specifies the less than or equal to function operator.
neq (Not Equal To)	Specifies the not equal to function operator.

Referenced By
if@op (§5.9.2.15)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FunctionOperator" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="equ"/>
      <enumeration value="neq"/>
      <enumeration value="gt"/>
      <enumeration value="lt"/>
      <enumeration value="gte"/>
      <enumeration value="lte"/>
   </restriction>
</simpleType>
```

5.9.7.31 ST_FunctionType (Function Type)

This simple type defines the set of available conditional expression function types present for use.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
cnt (Count)	Specifies a count.
depth (Depth)	Specifies the depth.
maxDepth (Max Depth)	Defines the maximum depth.
pos (Position)	Retrieves the position of the node in the specified set of nodes.
posEven (Position Even)	Returns 1 if the specified node is at an even numbered position in the data model.
posOdd (Position Odd)	Returns 1 if the specified node is in an odd position in the data model.
revPos (Reverse Position)	Reverse position function.
var (Variable)	Used to reference a variable.

Referenced By
if@func (§5.9.2.15)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FunctionType" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="cnt"/>
      <enumeration value="pos"/>
      <enumeration value="revPos"/>
      <enumeration value="posEven"/>
      <enumeration value="posOdd"/>
      <enumeration value="var"/>
      <enumeration value="depth"/>
      <enumeration value="maxDepth"/>
   </restriction>
</simpleType>
```

5.9.7.32 ST_FunctionValue (Function Value)

Conditional expression function value.

This simple type is defined as a union of the following types:

- TheXML Schema int datatype.
- TheXML Schema boolean datatype.
- TheST_Direction simple type (§5.9.7.24).
- TheST_HierBranchStyle simple type (§5.9.7.35).
- TheST_AnimOneStr simple type (§5.9.7.3).
- TheST_AnimLvlStr simple type (§5.9.7.2).
- TheST_ResizeHandlesStr simple type (§5.9.7.53).

Referenced By
if@val (§5.9.2.15)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FunctionValue" final="restriction">
   <union memberTypes="xsd:int xsd:boolean ST_Direction ST_HierBranchStyle ST_AnimOneStr
     ST_AnimLvlStr ST_ResizeHandlesStr"/>
</simpleType>
```

5.9.7.33 ST_GrowDirection (Grow Direction)

This simple type defines different starting locations for nodes within the snake algorithm.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
bL (Bottom Left)	Specifies the placement of nodes is to start in the bottom left corner.

Enumeration Value	Description
bR (Bottom Right)	Specifies the placement of nodes is to start in the bottom right corner.
tL (Top Left)	Specifies the placement of nodes is to start in the top left corner.
tR (Top Right)	Specifies the placement of nodes is to stat in the top right corner.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_GrowDirection" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="tL"/>
      <enumeration value="tR"/>
      <enumeration value="bL"/>
      <enumeration value="bR"/>
   </restriction>
</simpleType>
```

5.9.7.34 ST_HierarchyAlignment (Hierarchy Alignment)

This simple type defines different relative locations of child nodes and their descendants to a parent node within a hierarchy diagram.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
bCtrCh (Bottom Center Child)	Specifies the child nodes are placed below the parent node and that they are center aligned to the parent node.
bCtrDes (Bottom Center Descendant)	Specifies the descendant nodes are placed below the parent node and that they are center aligned to the parent node.
bL (Bottom Left)	Specifies the child and descendant nodes are placed below the parent node and that the set is left aligned.
bR (Bottom Right)	Specifies the child and descendant nodes are placed below the parent node and the set is right aligned.
lB (Left Bottom)	Specifies the child and descendant nodes are placed to the left of the parent node and that the set is bottom aligned.

Enumeration Value	Description
lCtrCh (Left Center Child)	Specifies the child nodes are placed to the left of the parent node and that the set is center aligned.
lCtrDes (Left Center Descendant)	Specifies the descendant nodes are placed to the left of the parent node and that the set is center aligned.
lT (Left Top)	Specifies the child and descendant nodes are placed to the left of the parent node and that the set is top aligned.
rB (Right Bottom)	Specifies the child and descendant nodes are placed to the right of the parent node and that the set is bottom aligned.
rCtrCh (Right Center Children)	Specifies the child nodes are placed to the right of the parent node and that the set is center aligned.
rCtrDes (Right Center Descendants)	Specifies the descendant nodes are placed to the right of the parent node and that the set is center aligned.
rT (Right Top)	Specifies the child and descendant nodes are placed to the right of the parent node and that the set is top aligned.
tCtrCh (Top Center Children)	Specifies the child nodes are placed above the parent node and that the set is center aligned.
tCtrDes (Top Center Descendants)	Specifies the descendant nodes are placed above the parent node and that the set is center aligned.
tL (Top Left)	Specifies the child and descendant nodes are placed above the parent node and that the set is left aligned.
tR (Top Right)	Specifies the child and descendant nodes are placed above the parent node and that the set is right aligned.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

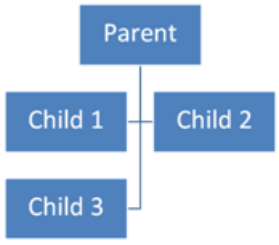
```
<simpleType name="ST_HierarchyAlignment" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="tL"/>
      <enumeration value="tR"/>
      <enumeration value="tCtrCh"/>
      <enumeration value="tCtrDes"/>
      <enumeration value="bL"/>
      <enumeration value="bR"/>
      <enumeration value="bCtrCh"/>
      <enumeration value="bCtrDes"/>
      <enumeration value="lT"/>
      <enumeration value="lB"/>
      <enumeration value="lCtrCh"/>
      <enumeration value="lCtrDes"/>
      <enumeration value="rT"/>
      <enumeration value="rB"/>
      <enumeration value="rCtrCh"/>
      <enumeration value="rCtrDes"/>
   </restriction>
</simpleType>
```

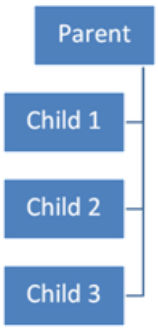
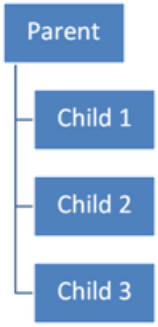
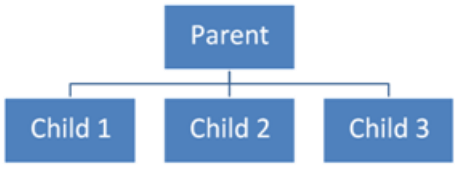
5.9.7.35 ST_HierBranchStyle (Hierarchy Branch Style Definition)

This simple type specifies the possible values for the branch style of a hierarchy diagram.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
hang (Hanging)	The branch style is hanging from the parent. [*Example*: Consider the following image as an example of a hanging branch style:  *end example*]
init (Initial)	This means that the value has not been set.
l (Left)	The branch style falls off the left. [*Example*: Consider the following image as an example

Enumeration Value	Description
	of a left hanging branch style:  *end example*]
r (Right)	The branch style falls off the right. [*Example*: Consider the following image as an example of a right hanging branch style:  *end example*]
std (Standard)	The standard branch style is to be used. [*Example*: Consider the following image as an example of a standard hanging branch style:  *end example*]

Referenced By
hierBranch@val (§5.9.6.7); ST_FunctionValue (§5.9.7.32)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_HierBranchStyle" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="l"/>
      <enumeration value="r"/>
      <enumeration value="hang"/>
      <enumeration value="std"/>
      <enumeration value="init"/>
   </restriction>
</simpleType>
```

5.9.7.36 ST_HorizontalAlignment (Horizontal Alignment)

This simple type defines the horizontal alignment.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
ctr (Center)	Specifies center alignment.
l (Left)	Specifies left alignment.
none (None)	Specifies no alignment defined.
r (Right)	Specifies right alignment.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_HorizontalAlignment" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="l"/>
      <enumeration value="ctr"/>
      <enumeration value="r"/>
      <enumeration value="none"/>
   </restriction>
</simpleType>
```

5.9.7.37 ST_HueDir (Hue Direction)

When given two colors to interpolate between, one can go in one of two directions around the color wheel to perform the interpolation. This type defines that direction.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
ccw (Counterclockwise Hue Direction)	A hue interpolation in the counterclockwise direction.
cw (Clockwise Hue Direction)	A hue interpolation in the clockwise direction.

Referenced By
effectClrLst@hueDir (§5.9.4.7); fillClrLst@hueDir (§5.9.4.8); linClrLst@hueDir (§5.9.4.9); txEffectClrLst@hueDir (§5.9.4.12); txFillClrLst@hueDir (§5.9.4.13); txLinClrLst@hueDir (§5.9.4.14)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_HueDir">
   <restriction base="xsd:token">
      <enumeration value="cw"/>
      <enumeration value="ccw"/>
   </restriction>
</simpleType>
```

5.9.7.38 ST_Index1 (1-Based Index)

A 1-based index.

This simple type's contents are a restriction of the XML Schema unsignedInt datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 1.

Referenced By
adj@idx (§5.9.2.1)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Index1">
   <restriction base="xsd:unsignedInt">
      <minInclusive value="1"/>
   </restriction>
</simpleType>
```

5.9.7.39 ST_Ints (Integer List)

A list of integers.

This simple type allows a list of items of the XML Schema int datatype.

Referenced By

Referenced By
forEach@st (§5.9.2.14); forEach@step (§5.9.2.14); if@st (§5.9.2.15); if@step (§5.9.2.15); presOf@st (§5.9.2.21); presOf@step (§5.9.2.21)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Ints">
   <list itemType="xsd:int"/>
</simpleType>
```

5.9.7.40 ST_LayoutShapeType (Layout Shape Type)

All of the available shape types.

This simple type is defined as a union of the following types:

- TheST_ShapeType simple type (§5.1.12.56).
- TheST_OutputShapeType simple type (§5.9.7.47).

Referenced By
shape@type (§5.9.2.27)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_LayoutShapeType" final="restriction">
   <union memberTypes="a:ST_ShapeType ST_OutputShapeType"/>
</simpleType>
```

5.9.7.41 ST_LinearDirection (Linear Direction)

This simple type defines the direction of growth of new nodes.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
fromB (From Bottom)	Specifies growth to start from the bottom.
fromL (From Left)	Specifies growth to start from the left.
fromR (From Right)	Specifies growth to start from the right.
fromT (From Top)	Specifies growth to start from the Top

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_LinearDirection" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="fromL"/>
      <enumeration value="fromR"/>
      <enumeration value="fromT"/>
      <enumeration value="fromB"/>
   </restriction>
</simpleType>
```

5.9.7.42 ST_ModelId (Model Identifier)

The unique ID of the element within the data model. Model Identifiers can be either longs or guids.

This simple type is defined as a union of the following types:

- TheXML Schema int datatype.
- TheST_Guid simple type (§5.1.12.27).

Referenced By
cxn@destId (§5.9.3.2); cxn@modelId (§5.9.3.2); cxn@parTransId (§5.9.3.2); cxn@sibTransId (§5.9.3.2); cxn@srcId (§5.9.3.2); prSet@presAssocID (§5.9.3.4); pt@cxnId (§5.9.3.5); pt@modelId (§5.9.3.5)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ModelId">
   <union memberTypes="xsd:int a:ST_Guid"/>
</simpleType>
```

5.9.7.43 ST_NodeCount (Number of Nodes Definition)

This simple type defines a count of the number of nodes for a property in a diagram. A value of -1 shall mean that the value is unbounded.

This simple type's contents are a restriction of the XML Schema int datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to -1.

Referenced By
chMax@val (§5.9.6.4); chPref@val (§5.9.6.5)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_NodeCount">
   <restriction base="xsd:int">
      <minInclusive value="-1"/>
   </restriction>
</simpleType>
```

5.9.7.44 ST_NodeHorizontalAlignment (Node Horizontal Alignment)

This simple type defines the horizontal alignment of a node.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
ctr (Center)	Specifies center alignment.
l (Left)	Specifies left alignment.
r (Right)	Specifies right alignment.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_NodeHorizontalAlignment" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="l"/>
      <enumeration value="ctr"/>
      <enumeration value="r"/>
   </restriction>
</simpleType>
```

5.9.7.45 ST_NodeVerticalAlignment (Node Vertical Alignment)

This simple type defines the vertical alignment of a node.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
b (Bottom)	Specifies bottom alignment.
mid (Middle)	Specifies middle alignment.
t (Top)	Specifies top alignment.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_NodeVerticalAlignment" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="t"/>
      <enumeration value="mid"/>
      <enumeration value="b"/>
   </restriction>
</simpleType>
```

5.9.7.46 ST_Offset (Offset)

This simple type defines whether or not subsequent rows or columns in the snake algorithm are offset from the preceding row or column.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
ctr (Center)	Specifies no offset.
off (Offset)	Specifies that the nodes will be shifted by some amount relative to the preceding row or column.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Offset" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="ctr"/>
      <enumeration value="off"/>
   </restriction>
</simpleType>
```

5.9.7.47 ST_OutputShapeType (Output Shape Type)

Shapes which are special specifically for a DrawingML diagram.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
conn (Connection)	Connection shape type.
none (None)	None.

Referenced By
ST_LayoutShapeType (§5.9.7.40)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_OutputShapeType" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="none"/>
      <enumeration value="conn"/>
   </restriction>
</simpleType>
```

5.9.7.48 ST_ParameterId (Parameter Identifier)

This simple type defines algorithm parameters which can be modified in order to adjust the behavior of algorithms for use in layout nodes.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
alignTx (Text Alignment)	This value defines how the text is aligned in a node.
ar (Aspect Ratio)	Specifies the aspect ratio (width to height) of the composite node to use when determining child constraints. A value of 0 means leave the width and height constraints as is. The algorithm may temporarily shrink one dimension to achieve that ratio. For example, if a composite node has a width constraint of 20 and height constraint of 10, and if ar=1.5, composite uses a width value of 15 to calculate the composite node's child constraints. However, the algorithm does not propagate this value to other nodes.
autoTxRot (Auto Text Rotation)	Auto text rotation.
begPts (Beginning Points)	Beginning Points
begSty (Beginning Arrowhead Style)	Beginning Arrowhead Style
bendPt (Bend Point)	The bend point.
bkpt (Breakpoint)	Specifies the point at which the diagram starts to snake. The value bal specifies that snaking begin at an even number of rows and columns. The value fixed specifies that snaking begin at a fixed point, for example, in a row that contains three nodes. The value endCnv specifies that snaking begin when there is no more room for a shape in the row.
bkPtFixedVal (Breakpoint Fixed Value)	Specifies where the snake should break, if bkpt=fixed.

Enumeration Value	Description
chAlign (Child Alignment)	Specifies the alignment of the children.
chDir (Child Direction)	The child direction.
connRout (Connection Route)	The route of the connection.
contDir (Continue Direction)	Specifies the direction of the subsequent row or column. For example, if the algorithm initially places the nodes from left to right, revDir places the nodes in the next row from right to left. However if the algorithm uses contDir, the nodes on the next row are arranged from left to right.
ctrShpMap (Center Shape Mapping)	Specifies where to place nodes in relation to the center circle.
dim (Connector Dimension)	Specifies the connector dimension.
dstNode (Destination Node)	Specifies the name of the layout node from which to end the connection from.
endPts (End Points)	Specifies the end points.
endSty (End Style)	Specifies the end style.
fallback (Fallback Scale)	1D specifies fallback. It only scales in one dimension. 2D specifies fallback. It scales in both dimensions equally.
flowDir (Flow Direction)	Specifies whether nodes are arranged in rows or columns.
grDir (Grow Direction)	Specifies from which corner the snake grows. For example, if the algorithm uses a top left value, the snake grows from the top left.
hierAlign (Hierarchy Alignment)	The alignment of the hierarchy.
horzAlign (Horizontal Alignment)	Aligns all the child nodes within the space reserved for the parent and adjusts child positions in the x direction.
linDir (Linear Direction)	Specifies the linear direction.
lnSpAfChP (Line Spacing After Children Paragraph)	Line spacing after children.
lnSpAfParP (Line Spacing After Parent Paragraph)	Line spacing after the parent.
lnSpCh (Line Spacing Children)	Line spacing of the children
lnSpPar (Line Spacing Parent)	Line spacing of the parent.
nodeHorzAlign (Node Horizontal Alignment)	Specifies how child nodes are aligned within the extents of the canvas. For example, you can align the tops of all the child nodes, but center all of them within the canvas.
nodeVertAlign (Node Vertical Alignment)	Specifies how child nodes are aligned within the extents of the canvas. Same as nodeHorzAlign, but in

Enumeration Value	Description
	the y direction.
off (Offset)	Specifies the offset.
parTxLTRAlign (Parent Text Left-to-Right Alignment)	Specifies the paragraph alignment of parent text when the shape has only parent text. This parameter applies when the text direction is left to right.
parTxRTLAlign (Parent Text Right-to-Left Alignment)	Specifies the paragraph alignment of parent text when the shape has only parent text. This parameter applies when the text direction is right to left.
pyraAcctBkgdNode (Pyramid Accent Background Node)	If pyramid has a composite child node, specifies the name of the node that is a child of the composite that makes up the child flyout shape. If the node specifies a shape of the nonIsoscelesTrapezoid autoshape, it modifies the adjust handles in order to fit the flyout flush against the side of the pyramid.
pyraAcctPos (Pyramid Accent Position)	Specifies the placement of the flyout grandchildren.
pyraAcctTxMar (Pyramid Accent Text Margin)	Specifies the placement of one edge of the child text (grandchild node). If the value is step, the text is against the edge of the pyramid. If the value is stack, the text aligns.
pyraAcctTxNode (Pyramid Accent Text Node)	If pyramid has a composite child node, specifies the child node that should hold the child text.
pyraLvlNode (Pyramid Level Node)	If pyramid has a composite child node, specifies the name of the node that is a child of the composite that makes up the pyramid itself. If the node specifies a trapezoid shape, it modifies the adjustment handles to construct a pyramid.
rotPath (Rotation Path)	The rotation path specified.
rtShortDist (Route Shortest Distance)	If true, the connector is routed through the shortest distance between the points.
secChAlign (Secondary Child Alignment)	The secondary child alignment.
secLinDir (Secondary Linear Direction)	The secondary linear direction.
shpTxLTRAlignCh (Shape Text Left-to-Right Alignment)	Specifies the paragraph alignment of all text within the shape when the shape contains both parent and child text. This parameter applies when the text direction is left to right.
shpTxRTLAlignCh (Shape Text Right-to-Left Alignment)	Specifies the paragraph alignment of all text within the shape when the shape contains both parent and child text. This parameter applies when the text direction is right to left.
spanAng (Span Angle)	Specifies the angle the cycle spans. Final shapealign text is placed at stAng+spanAng, unless spanAng=360.

Enumeration Value	Description
	In that case, the algorithm places the text so that shapes do not overlap.
srcNode (Source Node)	Specifies the name of the layout node from which to start the connection.
stAng (Start Angle)	Specifies the angle at which the first shape is placed. Angles are in degrees, measured clockwise from a line pointing straight upward from the center of the cycle.
stBulletLvl (Start Bullets At Level)	Specifies whether bullets start at the top level (1) or with children (2).
stElem (Start Element)	Specifies the point type of the layout node to use as the first shape in the cycle.
txAnchorHorz (Text Anchor Horizontal)	Specifies the y-axis position of the text area within a shape.
txAnchorHorzCh (Text Anchor Horizontal With Children)	Specifies that the definition can allow a different text anchoring on the x-axis, if child nodes exist in the shape.
txAnchorVert (Text Anchor Vertical)	Specifies the x-axis position of the text area within a shape.
txAnchorVertCh (Text Anchor Vertical With Children)	Specifies that the definition can allow a different text anchoring on the y-axis, if child nodes exist in the shape.
txBlDir (Text Block Direction)	Specifies whether the text block is vertical or horizontal.
txDir (Text Direction)	Specifies where the text of the first node starts.
vertAlign (Vertical Alignment)	Aligns all the child nodes within the space reserved for the parent and adjusts child positions in the y direction.

Referenced By
param@type (§5.9.2.20)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ParameterId" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="horzAlign"/>
      <enumeration value="vertAlign"/>
      <enumeration value="chDir"/>
      <enumeration value="chAlign"/>
      <enumeration value="secChAlign"/>
      <enumeration value="linDir"/>
      <enumeration value="secLinDir"/>
      <enumeration value="stElem"/>
      <enumeration value="bendPt"/>
      <enumeration value="connRout"/>
      <enumeration value="begSty"/>
      <enumeration value="endSty"/>
      <enumeration value="dim"/>
      <enumeration value="rotPath"/>
      <enumeration value="ctrShpMap"/>
      <enumeration value="nodeHorzAlign"/>
      <enumeration value="nodeVertAlign"/>
      <enumeration value="fallback"/>
      <enumeration value="txDir"/>
      <enumeration value="pyraAcctPos"/>
      <enumeration value="pyraAcctTxMar"/>
      <enumeration value="txBlDir"/>
      <enumeration value="txAnchorHorz"/>
      <enumeration value="txAnchorVert"/>
      <enumeration value="txAnchorHorzCh"/>
      <enumeration value="txAnchorVertCh"/>
      <enumeration value="parTxLTRAlign"/>
      <enumeration value="parTxRTLAlign"/>
      <enumeration value="shpTxLTRAlignCh"/>
      <enumeration value="shpTxRTLAlignCh"/>
      <enumeration value="autoTxRot"/>
      <enumeration value="grDir"/>
      <enumeration value="flowDir"/>
      <enumeration value="contDir"/>
      <enumeration value="bkpt"/>
      <enumeration value="off"/>
      <enumeration value="hierAlign"/>
      <enumeration value="bkPtFixedVal"/>
      <enumeration value="stBulletLvl"/>
      <enumeration value="stAng"/>
      <enumeration value="spanAng"/>
      <enumeration value="ar"/>
      <enumeration value="lnSpPar"/>
      <enumeration value="lnSpAfParP"/>
      <enumeration value="lnSpCh"/>
      <enumeration value="lnSpAfChP"/>
      <enumeration value="rtShortDist"/>
      <enumeration value="alignTx"/>
      <enumeration value="pyraLvlNode"/>
      <enumeration value="pyraAcctBkgdNode"/>
```

```
        <enumeration value="pyraAcctTxNode"/>
        <enumeration value="srcNode"/>
        <enumeration value="dstNode"/>
        <enumeration value="begPts"/>
        <enumeration value="endPts"/>
    </restriction>
</simpleType>
```

5.9.7.49 ST_ParameterVal (Parameter Values)

Specifies the list of parameter types that can be used by a diagram.

This simple type is defined as a union of the following types:

- TheST_HorizontalAlignment simple type (§5.9.7.36).
- TheST_VerticalAlignment simple type (§5.9.7.65).
- TheST_ChildDirection simple type (§5.9.7.14).
- TheST_ChildAlignment simple type (§5.9.7.13).
- TheST_SecondaryChildAlignment simple type (§5.9.7.55).
- TheST_LinearDirection simple type (§5.9.7.41).
- TheST_SecondaryLinearDirection simple type (§5.9.7.56).
- TheST_StartingElement simple type (§5.9.7.57).
- TheST_BendPoint simple type (§5.9.7.8).
- TheST_ConnectorRouting simple type (§5.9.7.19).
- TheST_ArrowheadStyle simple type (§5.9.7.4).
- TheST_ConnectorDimension simple type (§5.9.7.17).
- TheST_RotationPath simple type (§5.9.7.54).
- TheST_CenterShapeMapping simple type (§5.9.7.12).
- TheST_NodeHorizontalAlignment simple type (§5.9.7.44).
- TheST_NodeVerticalAlignment simple type (§5.9.7.45).
- TheST_FallbackDimension simple type (§5.9.7.27).
- TheST_TextDirection simple type (§5.9.7.62).
- TheST_PyramidAccentPosition simple type (§5.9.7.51).
- TheST_PyramidAccentTextMargin simple type (§5.9.7.52).
- TheST_TextBlockDirection simple type (§5.9.7.61).
- TheST_TextAnchorHorizontal simple type (§5.9.7.59).
- TheST_TextAnchorVertical simple type (§5.9.7.60).
- TheST_TextAlignment simple type (§5.9.7.58).
- TheST_AutoTextRotation simple type (§5.9.7.5).
- TheST_GrowDirection simple type (§5.9.7.33).
- TheST_FlowDirection simple type (§5.9.7.28).
- TheST_ContinueDirection simple type (§5.9.7.22).
- TheST_Breakpoint simple type (§5.9.7.11).
- TheST_Offset simple type (§5.9.7.46).

- TheST_HierarchyAlignment simple type (§5.9.7.34).
- TheXML Schema int datatype.
- TheXML Schema double datatype.
- TheXML Schema boolean datatype.
- TheXML Schema string datatype.
- TheST_ConnectorPoint simple type (§5.9.7.18).

Referenced By
param@val (§5.9.2.20)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ParameterVal">
   <union memberTypes="ST_HorizontalAlignment ST_VerticalAlignment ST_ChildDirection
     ST_ChildAlignment ST_SecondaryChildAlignment ST_LinearDirection ST_SecondaryLinearDirection
     ST_StartingElement ST_BendPoint ST_ConnectorRouting ST_ArrowheadStyle ST_ConnectorDimension
     ST_RotationPath ST_CenterShapeMapping ST_NodeHorizontalAlignment ST_NodeVerticalAlignment
     ST_FallbackDimension ST_TextDirection ST_PyramidAccentPosition ST_PyramidAccentTextMargin
     ST_TextBlockDirection ST_TextAnchorHorizontal ST_TextAnchorVertical ST_TextAlignment
     ST_AutoTextRotation ST_GrowDirection ST_FlowDirection ST_ContinueDirection ST_Breakpoint
     ST_Offset ST_HierarchyAlignment xsd:int xsd:double xsd:boolean xsd:string ST_ConnectorPoint"/>
</simpleType>
```

5.9.7.50 ST_PtType (Point Type)

This simple type defines the different point types which can be utilized to create diagrams in DiagramML.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
asst (Assistant Element)	This point type is used in a hierarchy diagram to represent an assistant element.
doc (Document)	This point type specifies a document type point. This point type can be thought of as the root node associated with the document itself.
node (Node)	The node point type specifies a basic point type.
parTrans (Parent Transition)	This point type specifies a parent transition element.
pres (Presentation)	Specifies a presentation point type.
sibTrans (Sibling Transition)	This point type specifies a sibling transition element.

Referenced By
pt@type (§5.9.3.5)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PtType">
   <restriction base="xsd:token">
      <enumeration value="node"/>
      <enumeration value="asst"/>
      <enumeration value="doc"/>
      <enumeration value="pres"/>
      <enumeration value="parTrans"/>
      <enumeration value="sibTrans"/>
   </restriction>
</simpleType>
```

5.9.7.51 ST_PyramidAccentPosition (Pyramid Accent Position)

This simple type defines different positioning for the accent shapes which can be associated with a pyramid algorithm.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
aft (Pyramid Accent After)	Specifies that the accent shapes are to be placed to the right of the pyramid.
bef (Before)	Specifies that the accent shapes are to be placed to the left of the pyramid.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PyramidAccentPosition" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="bef"/>
      <enumeration value="aft"/>
   </restriction>
</simpleType>
```

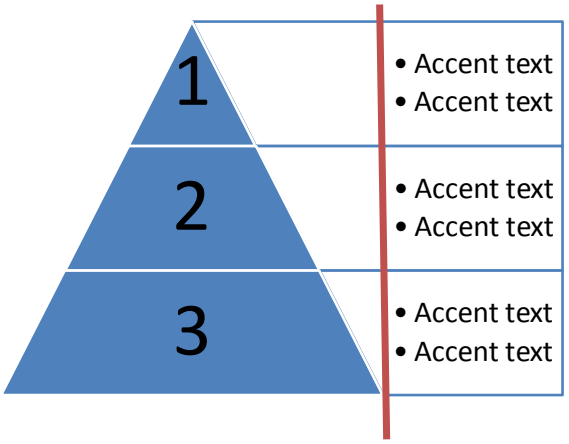
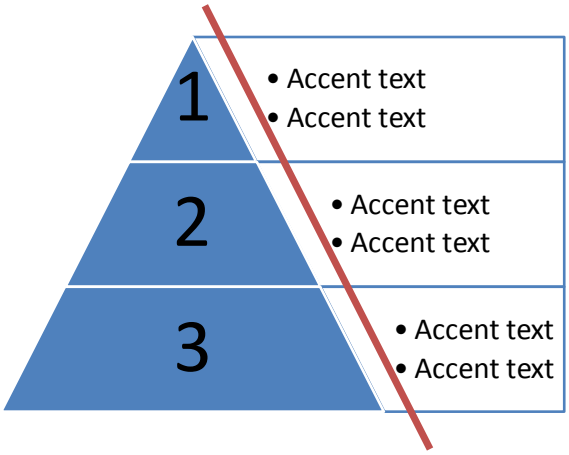
5.9.7.52 ST_PyramidAccentTextMargin (Pyramid Accent Text Margin)

This simple type defines different ways to lay out text in the accent shape for a pyramid algorithm.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description

Enumeration Value	Description
stack (Stack)	Specifies that all accent shape text is to be left aligned. [*Example*: Consider the following example of a pyramid diagram  *end example*]
step (Step)	Specifies that all accent shape text is to be relative to the pyramid. [*Example*: Consider the following example of a pyramid diagram  *end example*]

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_PyramidAccentTextMargin" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="step"/>
      <enumeration value="stack"/>
   </restriction>
</simpleType>
```

5.9.7.53 ST_ResizeHandlesStr (Resize Handle)

This simple type defines the possible behaviors when resizing shapes within a diagram. Because the size of the shape plays a large role in the overall layout of other nodes within the diagram, there are two ways resize can occur on a node.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
exact (Exact)	This value specifies that the resize of the shape occurs and sizes exactly to the size the user defines, which causes all other shapes in the diagram to shrink or grow accordingly.
rel (Relative)	This value specifies that resize operations happen relatively. This means that the relative size difference between nodes is maintained before and after the resize operation.

Referenced By
resizeHandles@val (§5.9.2.23); ST_FunctionValue (§5.9.7.32)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ResizeHandlesStr" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="exact"/>
      <enumeration value="rel"/>
   </restriction>
</simpleType>
```

5.9.7.54 ST_RotationPath (Rotation Path)

This simple type defines rotation properties for nodes within the cycle algorithm.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
alongPath (Along Path)	Specifies that the nodes should rotate in relation to their placement along the cycle.
none (None)	Specifies that the nodes should not rotate.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_RotationPath" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="none"/>
      <enumeration value="alongPath"/>
   </restriction>
</simpleType>
```

5.9.7.55 ST_SecondaryChildAlignment (Secondary Child Alignment)

This simple type defines different alignment properties of the both hanging layout type of the hierarchy algorithm.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
b (Bottom)	Specifies that the children nodes should be bottom aligned.
l (Left)	Specifies that the children nodes should be left aligned.
none (None)	Specifies no alignment.
r (Right)	Specifies that the children nodes should be right aligned.
t (Top)	Specifies that the children nodes should be top aligned.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SecondaryChildAlignment" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="none"/>
      <enumeration value="t"/>
      <enumeration value="b"/>
      <enumeration value="l"/>
      <enumeration value="r"/>
   </restriction>
</simpleType>
```

5.9.7.56 ST_SecondaryLinearDirection (Secondary Linear Direction)

This simple type defines different directions for the nodes in a both hanging layout in the hierarchy algorithm.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
fromB (From Bottom)	Specifies that the nodes begin from the bottom and move upward.
fromL (From Left)	Specifies that the nodes begin from the left and move right.
fromR (From Right)	Specifies that the nodes begin from the right and move left.
fromT (From Top)	Specifies that the nodes begin from the top and move downward.
none (None)	Specifies no direction.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SecondaryLinearDirection" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="none"/>
      <enumeration value="fromL"/>
      <enumeration value="fromR"/>
      <enumeration value="fromT"/>
      <enumeration value="fromB"/>
   </restriction>
</simpleType>
```

5.9.7.57 ST_StartingElement (Starting Element)

This simple type defines behavior for the first node in a cycle algorithm.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
node (Node)	Specifies that a node should be placed first.
trans (Transition)	Specifies that a transition should be placed first.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_StartingElement" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="node"/>
      <enumeration value="trans"/>
   </restriction>
</simpleType>
```

5.9.7.58 ST_TextAlignment (Text Alignment)

This simple type defines alignment types for text within a node.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
ctr (Center)	Specifies center aligned text.
l (Left)	Specifies left aligned text.
r (Right)	Specifies right aligned text.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextAlignment" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="l"/>
      <enumeration value="ctr"/>
      <enumeration value="r"/>
   </restriction>
</simpleType>
```

5.9.7.59 ST_TextAnchorHorizontal (Text Anchor Horizontal)

This simple type defines horizontal anchor points for text.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
ctr (Center)	Specifies text to be anchored to the center.
none (None)	Specifies no horizontal text anchor.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextAnchorHorizontal" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="none"/>
      <enumeration value="ctr"/>
   </restriction>
</simpleType>
```

5.9.7.60 ST_TextAnchorVertical (Text Anchor Vertical)

This simple type defines vertical anchor points for text.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
b (Bottom)	Specifies text to be anchored to the bottom.
mid (Middle)	Specifies text to be anchored to the middle.
t (Top)	Specifies text to be anchored to the top.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextAnchorVertical" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="t"/>
      <enumeration value="mid"/>
      <enumeration value="b"/>
   </restriction>
</simpleType>
```

5.9.7.61 ST_TextBlockDirection (Text Block Direction)

This simple type defines different layout options for text within a node.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
horz (Horizontal)	Specifies that the text is to be horizontal.
vert (Vertical Direction)	Specifies that the text is to be vertical.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextBlockDirection" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="horz"/>
      <enumeration value="vert"/>
   </restriction>
</simpleType>
```

5.9.7.62 ST_TextDirection (Text Direction)

This simple type defines different way the growth of additional text can occur within a node.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
fromB (From Bottom)	Specifies additional text grows from the bottom.
fromT (From Top)	Specifies additional text grows from the top.

Referenced By

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TextDirection" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="fromT"/>
      <enumeration value="fromB"/>
   </restriction>
</simpleType>
```

5.9.7.63 ST_UnsignedInts (Unsigned Integer List)

A list of unsigned integers.

This simple type allows a list of items of the XML Schema unsignedInt datatype.

Referenced By
forEach@cnt (§5.9.2.14); if@cnt (§5.9.2.15); presOf@cnt (§5.9.2.21)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_UnsignedInts">
   <list itemType="xsd:unsignedInt"/>
</simpleType>
```

5.9.7.64 ST_VariableType (Variable Type)

Conditional expression variable type.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
animLvl (Animation Level)	Specifies the animation level
animOne (Animate One)	Specifies animate as one.
bulEnabled (Bullets Enabled)	Specifies bullets enabled.
chMax (Child Max)	The maximum number of children.
chPref (Child Preference)	The preferred number of children.
dir (Direction)	Specifies the direction of the diagram.
hierBranch (Hierarchy Branch)	The hierarchy branch.
none (Unknown)	Unknown variable type.
orgChart (Organizational Chart Algorithm)	Algorithm that lays out an org chart.
resizeHandles (Resize Handles)	Specifies the resize handles.

Referenced By
ST_FunctionArgument (§5.9.7.29)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_VariableType" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="none"/>
      <enumeration value="orgChart"/>
      <enumeration value="chMax"/>
      <enumeration value="chPref"/>
      <enumeration value="bulEnabled"/>
      <enumeration value="dir"/>
      <enumeration value="hierBranch"/>
      <enumeration value="animOne"/>
      <enumeration value="animLvl"/>
      <enumeration value="resizeHandles"/>
   </restriction>
</simpleType>
```

5.9.7.65 ST_VerticalAlignment (Vertical Alignment)

This simple type defines different vertical alignment parameters.

This simple type's contents are a restriction of the XML Schema token datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
b (Bottom)	Specifies bottom aligned.
mid (Middle)	Specifies middle aligned.
none (None)	Specifies no vertical alignment.
t (Top)	Specifies top aligned.

Referenced By
ST_ParameterVal (§5.9.7.49)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_VerticalAlignment" final="restriction">
   <restriction base="xsd:token">
      <enumeration value="t"/>
      <enumeration value="mid"/>
      <enumeration value="b"/>
      <enumeration value="none"/>
   </restriction>
</simpleType>
```

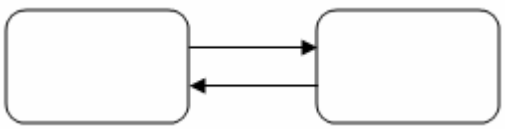
6. VML Reference Material

6.1 VML

VML is a language for defining graphical objects in cases where DrawingML does not apply, such as text boxes and shapes in WordprocessingML documents and comments and controls in SpreadsheetML documents. This namespace provides the base elements and attributes for defining shape primitives. Other VML namespaces define elements that layer on information beyond the baseline graphical definition. To maintain backward compatibility, all VML namespaces defined in this specification maintain the legacy namespace structure already used by millions of documents.

[*Note*: The VML format is a legacy format originally introduced with Office 2000 and is included and fully defined in this Standard for backwards compatibility reasons. The DrawingML format is a newer and richer format created with the goal of eventually replacing any uses of VML in the Office Open XML formats. VML should be considered a deprecated format included in Office Open XML for legacy reasons only and new applications that need a file format for drawings are strongly encouraged to use preferentially DrawingML .*end note*]

[*Example*: Assume the following shapes exist in a WordprocessingML document:



The drawing consists of four shapes. The arrows are specified by extending the shape type base definition in the shapetype element. Each shape representing an arrow references the shapetype it is extending via its type attribute.

```
<v:shapetype id="_x0000_t32" coordsize="21600,21600" o:spt="32" o:oned="t"
  path="m,l21600,21600e" filled="f">
  <v:path arrowok="t" fillok="f" o:connecttype="none"/>
  <o:lock v:ext="edit" shapetype="t"/>
</v:shapetype>

<v:shape id="_x0000_s1030" type="#_x0000_t32" style="position:absolute;left:0;
  text-align:left;margin-left:105pt;margin-top:36pt;width:48pt;height:0;flip:x;
  z-index:251661312" o:connectortype="straight">
  <v:stroke endarrow="block"/>
</v:shape>

<v:shape id="_x0000_s1029" type="#_x0000_t32" style="position:absolute;left:0;
  text-align:left;margin-left:105pt;margin-top:21.75pt;width:48pt;height:0;
```

```
    z-index:251660288" o:connectortype="straight">
    <v:stroke endarrow="block"/>
  </v:shape>
```

The rounded rectangles use the VML roundrect element.

```
  <v:roundrect id="_x0000_s1028" style="position:absolute;left:0;
    text-align:left;margin-left:153pt;margin-top:8.25pt;width:68.25pt;height:42pt;
    z-index:251659264" arcsize="10923f"/>

  <v:roundrect id="_x0000_s1027" style="position:absolute;left:0;
    text-align:left;margin-left:36.75pt;margin-top:8.25pt;width:68.25pt;
    height:42pt;z-index:251658240" arcsize="10923f"/>
```

end example]

Note that, throughout VML, numeric values that are allowed to take units may be specified in: cm (centimeters), mm (millimeters), in (inches), pt (points), pc (picas), px (pixels).

6.1.1 Table of Contents

This subclause is informative.

End of informative text.

6.1.2 Elements

The following elements comprise the contents of the urn:schemas-microsoft-com:vml namespace:

[*Note*: As the VML format is a format provided for backward compatibility, many VML elements are defined in the same urn:schemas-microsoft-com:vml namespace that is already used by millions of documents already using VML. *end note*]

6.1.2.1 arc (Arc Segment)

This element specifies an arc defined as a segment of an oval. The CSS2 style content width and height define the width and height of that oval. The arc is defined by the intersection of the oval with the start and end radius vectors given by the angles. The angles are calculated on the basis of a circle (width equal to height) which is then scaled anisotropically to the desired width and height.

[*Example:* The following specifies a simple half-circle arc open at the top:

```
<v:arc
  style="position:relative;top:120;left:20;width:200;height:200"
  startangle="90" endangle="270">
</v:arc>
```

The shape looks like this:

end example]

Parent Elements
background (§2.2.1); group (§6.1.2.7); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23)

Child Elements	Subclause
anchorlock (Anchor Location Is Locked)	§6.3.2.1
borderbottom (Bottom Border)	§6.3.2.2
borderleft (Left Border)	§6.3.2.3
borderright (Right Border)	§6.3.2.4
bordertop (Top Border)	§6.3.2.5
callout (Callout)	§6.2.2.2
ClientData (Attached Object Data)	§6.4.2.12
clippath (Shape Clipping Path)	§6.2.2.3
extrusion (3D Extrusion)	§6.2.2.10
fill (Shape Fill Properties)	§6.1.2.5
formulas (Set of Formulas)	§6.1.2.6
handles (Set of Handles)	§6.1.2.9
imagedata (Image Data)	§6.1.2.11
lock (Shape Protections)	§6.2.2.17
path (Shape Path)	§6.1.2.14
shadow (Shadow Effect)	§6.1.2.18
signatureline (Digital Signature Line)	§6.2.2.29
skew (Skew Transform)	§6.2.2.30
stroke (Line Stroke Settings)	§6.1.2.21
textbox (Text Box)	§6.1.2.22
textdata (VML Diagram Text)	§6.5.2.2
textpath (Text Layout Path)	§6.1.2.23
wrap (Text Wrapping)	§6.3.2.6

Attributes	Description
allowincell (Allow in Table Cell) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape can be placed in a table. Default is `false`. [*Example:* `<v:shape ... o:allowincell="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
allowoverlap (Allow Shape Overlap) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape can overlap another shape. Default is `true`. If `false`, the shape will shift left or right so as not to overlap another shape, similar to the behavior of the HTML float attribute. [*Example:* `<v:shape ... o:allowoverlap="false" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

alt (Alternate Text)	Specifies alternative text describing the graphical object. This text should provide a brief description of the shape for use by accessibility tools. Default is no value.
	
	[*Example:* The alt text describes the basic shape:
	
	`<v:shape ... fillcolor="red" alt="Red rectangle"></v:shape>`
	
	The alt text describes the contents of a shape displaying an image:
	
	`<v:shape ... alt="Picture of a sunset"></v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

borderbottomcolor (Bottom Border Color)	Specifies the bottom border color of an inline shape. Default is no value. [*Example:*

Attributes	Description
Namespace: urn:schemas-microsoft-com:office:office	<v:shape ... o:borderbottomcolor="red" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
borderleftcolor (Border Left Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the left border color of an inline shape. Default is no value. [*Example:* <v:shape ... o:borderleftcolor="red" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
borderrightcolor (Border Right Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the right border color of an inline shape. Default is no value. [*Example:* <v:shape ... o:borderrightcolor="red" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
bordertopcolor (Border Top Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the top border color of an inline shape. Default is no value. [*Example:* <v:shape ... o:bordertopcolor="red" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
bullet (Graphical Bullet) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the shape is a graphical bullet. Default is `false`. [*Example:* <v:shape ... o:bullet="true" ... > </v:shape> *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
button (Button Behavior Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape will exhibit button press behavior on click. Default is `false`. [*Example:* `<v:shape ... o:button="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
bwmode (Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies how a shape will render for black-and-white output devices. When a shape is printed on a black-and-white printer or displayed in a black-and-white view in an application, several options are possible. Default is `auto`, which will use o:bwnormal for normal black-and-white rendering and o:bwpure for pure black-and-white rendering. bwnormal and bwpure are subordinate to bwmode. If bwmode is "auto" then the value for bwnormal or bwpure is used depending on what the output format is. An application may define for itself what, if any, difference there is between normal B&W and pure B&W. For example, normal B&W might allow greyscale and pure B&W might not. [*Example:* This shape renders in grayscale in a black-and-white environment: `<v:shape ... o:bwmode="grayscale" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
bwnormal (Normal Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies the black-and-white mode for normal black-and-white output devices. Default is `auto`. [*Example:* This shape renders in a pale grayscale in a normal black-and-white environment: `<v:shape ... o:bwmode="auto" o:bwnormal="lightgrayscale" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
bwpure (Pure	Specifies the black-and-white mode for pure black-and-white output devices. Default is

Attributes	Description
Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	auto. [*Example:* This shape renders in high contrast when in a pure black-and-white environment: `<v:shape ... o:bwmode="auto" o:bwpure="highcontrast" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
chromakey (Image Transparency Color)	Specifies a color value that will be transparent and show anything behind the shape. Default is no value. [*Example:* `<v:image ... chromakey="white" ...> </v:image>` *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).
class (CSS Reference)	Specifies a reference to the definition of a CSS style. Default is no value. [*Example:* The snippets below are equivalent: `... .narrowstyle {width:50;height:100} ... <v:shape ... class="narrowstyle" style="top:1;left:1"> </v:shape>` `<v:shape ... style="top:1;left:1; width:50;height:100"> </v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
clip (Clipping Toggle)	Specifies that the clipping region is active. This is used in conjunction with the clippath (§6.2.2.3) element to create a clipping region.

Attributes	Description

Namespace: urn:schemas-microsoft-com:office:office	[*Example:*
	
	`<v:shape ... o:clip="true">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

cliptowrap (Clip to Wrapping Polygon)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies that the clipping region for the shape aligns with the wrapping polygon that tightly surrounds the entire shape (essentially, that the shape shall not be drawn beyond its wrapping polygon's extents – if it does, the shape shall be clipped). Default is `false`.
	
	[*Example:*
	
	`<v:shape ... o:cliptowrap="true">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

connectortype (Shape Connector Type)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies the type of connector used for joining shapes. Default is `straight`.
	
	[*Example:*
	
	`<v:shape ... o:connectortype="elbow" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_ConnectorType simple type (§6.2.3.6).

coordorigin (Coordinate Space Origin)	Specifies the coordinate of the top left corner of the shape's coordinate space. This determines the position of the (0,0) coordinate space origin within the shape's bounding box. Default is "0,0", which places the (0,0) origin at the top left corner of the bounding box. This affects shape properties that specify coordinate positions, such as the path attribute. Thus a path can be defined against a generic (0,0) origin and the coordorigin value translates the entire path within the shape's bounding space. [*Example:* The horizontal and vertical coordinate space ranges from -100 to +100 because the coordinate space (coordsize) is 200 by 200 and the top left coordinate is (-100,-100).

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>The (0,0) origin lies at the center of the shape's bounding box, as evidenced by the position of the shape's path within the coordinate space:

<pre><v:shape ... coordsize="200,200"
  coordorigin="-100,-100"
  path="m 0,0 l 0,50, 50,50, 50,0 x e">
</v:shape></pre>
 

end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>coordsize
(Coordinate Space Size)</td><td>Specifies the size of the shape's coordinate space in coordinate units. Default is "1000,1000".

The physical size of a coordinate unit length is determined by both the size of the coordinate space (coordsize) and the size of the shape (style width and height). The coordsize attribute defines the number of horizontal and vertical subdivisions into which the shape's bounding box is divided. The combination of coordsize and style width/height effective scales the shape anisotropically.

[Example: The path is 50 units wide and tall, which is 25% of the size of the coordinate space:

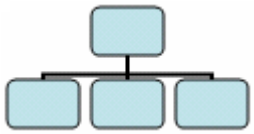
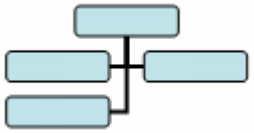
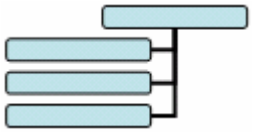
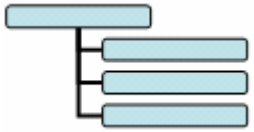
<pre><v:shape ... coordsize="200,200"
  coordorigin="-100,-100"
  path="m 0,0 l 0,50, 50,50, 50,0 x e">
</v:shape></pre>
 

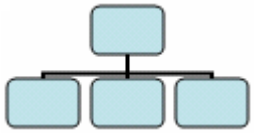
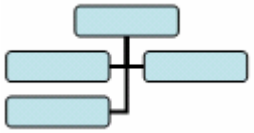
end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>dgmlayout
(Diagram Node Layout Identifier)

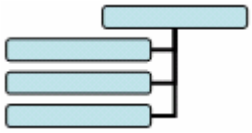
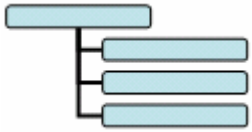
Namespace: urn:schemas-microsoft-</td><td>Specifies the type of automatic layout to apply to the diagram node. This is only meaningful for organization charts. Valid values for organization charts are:

<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Top-down, centered layout.</td></tr></table></td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>com:office:office</td><td>
<table>
<tr><td></td><td>  </td></tr>
<tr><td>1</td><td>Hanging, both sides layout.
  </td></tr>
<tr><td>2</td><td>Hanging, right side layout.
  </td></tr>
<tr><td>3</td><td>Hanging, left side layout.
  </td></tr>
</table>
[Example:
<pre><v:shape ... dgmlayout="1">
</v:shape></pre>
end example]
The possible values for this attribute are defined by the XML Schema integer datatype.
</td></tr>
<tr><td>dgmlayoutmru (Diagram Node Recent Layout Identifier)

Namespace: urn:schemas-microsoft-com:office:office</td><td>
Specifies the type of automatic layout most recently used on the child elements of the diagram node. This is only meaningful for organization charts. Valid values for organization charts are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Top-down, centered layout.
  </td></tr>
<tr><td>1</td><td>Hanging, both sides layout.
  </td></tr>
<tr><td>2</td><td>Hanging, right side layout.</td></tr>
</table>
</td></tr>
</table>

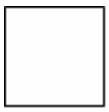
Attributes	Description

	<table><tr><td></td><td>  </td></tr><tr><td>3</td><td>Hanging, left side layout.  </td></tr></table>
	[*Example:*
	
	`<v:shape ... dgmlayout="1">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema integer datatype.

dgmnodekind (Diagram Node Identifier)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies an optional, application-specific parameter that is intended to be used by the application to tag different types of nodes in a diagram.
	
	[*Example:*
	
	`<v:shape ... dgmnodekind="1">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema integer datatype.

doubleclicknotify (Double-click Notification Toggle)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies that an event message is sent when a shape is double-clicked. Default is `false`.
	
	[*Example:*
	
	`<v:shape ... o:doubleclicknotify="true" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

endAngle (Ending Angle)	Specifies the angle that defines the endpoint of the arc. The angle is measured in degrees clockwise from the vertical. Default is `90`. [*Example:* This arc ends at the bottom center of the shape's region:

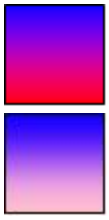
Attributes	Description
	<v:arc ... endangle="180" ... > </v:arc> *end example*] The possible values for this attribute are defined by the XML Schema decimal datatype.
fillcolor (Fill Color)	Specifies the color to use for the fill. Default is white. If the fill element (§6.1.2.5) is present, its color attribute takes precedence. Colors are typically specified as either a named color, such as red, or six hexadecimal digits representing the red, green and blue values of the color, such as #00FF30. Full details are specified in the simple type description. [*Example:* This shape is red if its fill is visible: <v:shape ... fillcolor="red" ... > </v:shape> This is equivalent to: <v:shape ... fillcolor="#ff0000" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).
filled (Shape Fill Toggle)	Specifies whether the closed path will be filled. Default is true. This attribute is overridden by the fill on attribute. [*Example:* <v:shape ... filled="f" fillcolor="red" ...> </v:shape>  *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
forcedash (Force Dashed Outline)	Specifies whether a dashed outline is used to draw a shape when a shape has no line or fill. Default is false.

Attributes	Description
Namespace: urn:schemas-microsoft-com:office:office	Used by PresentationML placeholders to draw a dashed outline when there is no line and no fill for a shape. [*Example:* `<v:shape ... o:forcedash="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
hr (Horizontal Rule Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies that a shape is a horizontal rule. Default is `false`. [*Example:* `<v:shape ... o:hr="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
hralign (Horizontal Rule Alignment) Namespace: urn:schemas-microsoft-com:office:office	Specifies the alignment of a horizontal rule. Default is `left`. [*Example:* `<v:shape ... o:hralign="center" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_HrAlign simple type (§6.2.3.14).
href (Hyperlink Target)	Specifies a hyperlink URL target for the shape. Default is no value. [*Example:* `<v:shape ... href="http://www.openxmlformats.org" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

Attributes	Description
hrnoshade (Horizontal Rule 3D Shading Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies that the horizontal rule does not have 3-D shading. Default is `false`. [*Example:* `<v:shape ... o:hrnoshade="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
hrpct (Horizontal Rule Length Percentage) Namespace: urn:schemas-microsoft-com:office:office	Specifies the length of a horizontal rule as a percentage of page width. Default is 0. [*Example:* `<v:shape ... o:hrpct="85" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema float datatype.
hrstd (Horizontal Rule Standard Display Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape is displayed as a standard horizontal rule. Only applies if hr is `true`. Default is `false`. [*Example:* `<v:shape ... o:hrstd="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
id (Unique Identifier)	Specifies a unique identifier that can be used to reference a VML object. Default is no value. [*Example:* `<v:shape ... id="myShape" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

Attributes	Description
insetmode (Text Inset Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the application calculates the internal text margin instead of using the inset attribute. Default is `custom`. This attribute is only meaningful for text boxes. [*Example:* `<v:shape ... o:insetmode="auto" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_InsetMode simple type (§6.2.3.15).
insetpen (Inset Border From Path)	Specifies that the border shall be displayed inside of the path defining the shape, rather than along the path (the default border placement), or outside of the path as might be done with an image. [*Example:* `<v:shape ... insetpen="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
ole (Embedded Object Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the shape is an embedded object. Default is `false`. [*Example:* `<v:shape ... o:ole="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalseBlank simple type (§6.2.3.24).
oleicon (Embedded Object Icon Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether an embedded object will be displayed as an icon. Default is `false`. [*Example:* `<v:shape ... o:oleicon="true" ... ></v:shape>` *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
oned (Shape Handle Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the extra handles of a shape are hidden. If `true`, hides all shape handles except the top left and bottom right; that is, the same handles that are used for a straight line segment. Default is `false`. [*Example:* `<v:shape ... o:oned="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

opacity (Fill Color Opacity)	Specifies the opacity of the primary fill color. Default is 1.0. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8.
	
	[*Example:* The red color is 25% opaque:
	
	`<v:fill type="gradient" color="red" color2="blue" opacity=".25"></v:fill>`
	
	  `opacity="1"`
	 `opacity=".25"`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

preferrelative (Relative Resize Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the original size of an object is saved after reformatting. Default is `false`. If `true`, the original size of the object is stored and all resizing is based on a percentage of that original size. Otherwise, each resizing will reset the scale to 100%. [*Example:* `<v:shape ... o:preferrelative="true" ... ></v:shape>` *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
print (Print Toggle)	Specifies whether the shape will be printed. Default is `true`. [*Example:* `<v:shape ... print="false" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
regroupid (Regroup ID) Namespace: urn:schemas-microsoft-com:office:office	Specifies a previous group for a shape. An ID number is used to identify groups of shapes that are no longer grouped. This allows shapes to be regrouped programmatically. [*Example:* The shape was part of a group identified by the ID 040754: `<v:shape ... o:regroupid="040754" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema integer datatype.
spid (Optional String) Namespace: urn:schemas-microsoft-com:office:office	Specifies an optional string that an application may use to Identify the particular shape. Default is no value. The possible values for this attribute are defined by the XML Schema string datatype.
spt (Optional Number) Namespace: urn:schemas-microsoft-com:office:office	Specifies an optional number that an application may use to associate the particular shape with a defined shape type. Default is 0. The possible values for this attribute are defined by the XML Schema float datatype.
startAngle (Starting Angle)	Specifies an angle that defines the starting point of the arc. The angle is measured in degrees clockwise from the vertical. Default is `0`. [*Example:* This arc begins in the upper-right quadrant:

Attributes	Description
	<v:arc ... startangle="45" ... > </v:arc> *end example*] The possible values for this attribute are defined by the XML Schema decimal datatype.
strokecolor (Shape Stroke Color)	Specifies the primary color of the brush to use to stroke the path of the shape. Default is black. The color attribute of the stroke element (§6.1.2.21) overrides this. Colors are typically specified as either a named color, such as red, or six hexadecimal digits representing the red, green and blue values of the color, such as #00FF30. Full details are specified in the simple type description. [*Example:* <v:shape ... strokecolor="red" ...> </v:shape>  *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).
stroked (Shape Stroke Toggle)	Specifies whether the path defining the shape is stroked with a solid line. The stroke element (§6.1.2.21) defines other strokes. The on attribute of the stroke element overrides this attribute. Default is true. [*Example:* <v:shape ... fillcolor="red" stroked="false" strokecolor="blue"...> </v:shape>  *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
strokeweight (Shape Stroke	Specifies the width of the brush to use to stroke the path. Default is 1 point. If a number is given without units, the emu is used. The weight attribute of the stroke element

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>Weight)</td><td>(§6.1.2.21) overrides this attribute.

[Example:

<pre><v:shape ... strokeweight="3pt" ... >
</v:shape></pre>
 

end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>style (Shape Styling Properties)</td><td>Specifies the CSS2 styling properties of the shape. This uses the syntax described in the "Visual formatting model" of the Cascading Style Sheets, Level 2 specification, a Recommendation of the World Wide Web Consortium available here: http://www.w3.org/TR/REC-CSS2. Full descriptions of each property are not repeated here, but the VML treatment of each property is defined. Allowed properties include:

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>flip</td><td>Specifies that the orientation of a shape is flipped. Default is no value. Allowed values are:

• x - Flip along the y-axis, reversing the x-coordinates.
• y - Flip along the x-axis, reversing the y-coordinates.
• xy - Flip along both the y- and x-axis.
• yx - Flip along both the x- and y-axis.</td></tr>
<tr><td>height</td><td>Specifies the height of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are:

• auto - Default position of an element in the flow of the page.
• <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed.
• <percentage>- Value expressed as a percentage of the parent object's height.</td></tr>
<tr><td>left</td><td>Specifies the position of the left of the containing block of the shape relative to the element left of it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are:</td></tr>
</table>
</td></tr>
</table>

Attributes	Description	
		• auto - Default position of an element in the flow of the page. • <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	margin-bottom	Specifies the position of the bottom of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	margin-left	Specifies the position of the left of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	margin-right	Specifies the position of the right of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.

Attributes	Description	
	margin-top	Specifies the position of the top of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	mso-position-horizontal	Specifies the horizontal positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute • left • center • right • inside • outside
	mso-position-horizontal-relative	Specifies relative horizontal position data for objects in WordprocessingML documents. This modifies the mso-position-horizontal property. Default is text. Allowed values are: • margin • page • text • char
	mso-position-vertical	Specifies the vertical positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute • top • center • bottom • inside • outside

Attributes	Description	
	mso-position-vertical-relative	Specifies relative vertical position data for objects in WordprocessingML documents. This modifies the mso-position-vertical property. Default is text. Allowed values are: • margin • page • text • line
	mso-wrap-distance-bottom	Specifies the distance from the bottom of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
	mso-wrap-distance-left	Specifies the distance from the left side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
	mso-wrap-distance-right	Specifies the distance from the right side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
	mso-wrap-distance-top	Specifies the distance from the top of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
	mso-wrap-edited	Specifies whether the wrap coordinates were customized by the user. If the wrap coordinates are generated by an editor, this property is true; otherwise they were customized by a user. Default is false.
	mso-wrap-style	Specifies the wrapping mode for text in shapes in WordprocessingML documents. Default is square. Allowed values are: • square - Wraps text inside the shape in a square. • none - Text does not wrap.
	position	Specifies the type of positioning used to place an element. Default is static. When the element is contained inside a group, this property must be absolute. Allowed values are:

Attributes	Description	
		• `static` - The element is positioned according to the normal flow of the page. The `top` and `left` properties are ignored. If the object is anchored inline, this value is used. • `absolute` - The element is positioned relative to the parent, using the `top` and `left` properties. • `relative` - The element is positioned according to the normal flow of the page, but the `top` and `left` properties are used. The overlap of overlapping elements is governed by the `z-index` property.
	`rotation`	Specifies the angle that a shape is rotated, in degrees. Default is 0. Positive angles are clockwise.
	`top`	Specifies the position of the top of the containing block of the shape relative to the element above it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	`visibility`	Specifies whether a shape is displayed. Only `inherit` and `hidden` are used; any other values are mapped to `inherit`. Default is `inherit`. Allowed values are: • `hidden` - The shape is not visible, but is still part of the flow of the objects in the browser. Mouse events are not processed. • `inherit` - The visibility state is inherited from the parent of the shape.
	`width`	Specifies the width of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed.

Attributes	Description
	<table><tr><td></td><td>• <percentage>- Value expressed as a percentage of the parent object's width.</td></tr><tr><td>z-index</td><td>Specifies the display order of overlapping shapes. Default is 0. This property shall not be used for shapes anchored inline. Allowed values are: • auto - Uses the order that the shapes appear in the page, bottom to top. • <order>- A number that represents the stacking precedence. Shapes with higher numbers are placed on top of those with lower numbers. Negative numbers are allowed.</td></tr></table> The following properties are only used by the textbox element (§6.1.2.22): <table><tr><th>Property</th><th>Description</th></tr><tr><td>direction</td><td>Specifies the direction of the text in the textbox. Default is ltr. This property is superceded by the mso-direction-alt property if that is specified. Allowed values are: • ltr - Test is displayed left-to-right. • rtl - Test is displayed right-to-left.</td></tr><tr><td>layout-flow</td><td>Determines the flow of the text layout in a textbox. Default is horizontal. Allowed values are: • horizontal - Text is displayed horizontally. • vertical - Text is displayed vertically. • vertical-ideographic - Ideographic text is displayed vertically. • horizontal-ideographic - Ideographic text is displayed horizontally.</td></tr><tr><td>mso-direction-alt</td><td>Specifies an alternate direction for text in textboxes. Overrides the direction property. The only allowed value is context.</td></tr><tr><td>mso-fit-shape-to-text</td><td>Specifies whether the shape stretches to fit the text in the textbox. Default is false.</td></tr><tr><td>mso-fit-text-to-shape</td><td>Specifies whether the text stretches to fit the textbox. Default is false.</td></tr><tr><td>mso-layout-</td><td>Specifies the alternate layout flow for text in textboxes. This</td></tr></table>

Attributes	Description
	flow-alt: property is used instead of layout-flow when the layout flow is from bottom to top for non-ideographic languages. Its only value is bottom-to-top.
	mso-next-textbox: Specifies the ID of the next textbox in a series. Used to keep track of a set of linked textboxes. Default is no value.
	mso-rotate: Specifies a specific rotation value for text in a textbox. Default is 0. Allowed values are: • 0 • 90 • 180 • -90
	mso-text-scale: Specifies the scaling factor for fitting text to shapes. Default is 0. This property is only used if mso-fit-text-to-shape is true.
	v-text-anchor: Specifies the vertical anchoring of text in a textbox. Default is top. The alignment of a text anchor only becomes evident if mso-fit-text-to-shape is false. This property is different from the vertical-align CSS property, which is used for ideographic languages. Allowed values are: • top • middle • bottom • top-center • middle-center • bottom-center • top-baseline • bottom-baseline • top-center-baseline • bottom-center-baseline

The following properties are only used by the textpath element (§6.1.2.23):

Property	Description
font	Specifies a compound value of font settings. Default is no value. The values are the same as those of the CSS font property. The order of definitions in the string is: font-style, font-variant, font-weight, font-size, line-height, font-family.
font-family	Specifies the family of the font. Default is no value. The values are the same as those of the CSS font-family property.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>font-size</td><td>Specifies the size of the font. Default is no value. The font size is defined in points. The values are the same as those of the CSS font-size property.</td></tr>
<tr><td>font-style</td><td>Specifies the amount of slant for a font. Default is normal. The values are the same as those of the CSS font-style property. Allowed values are:
normal
italic
oblique - Treated the same as italic.</td></tr>
<tr><td>font-variant</td><td>Specifies the variant style of a font. Default is normal. The values are the same as those of the CSS font-variant property. Allowed values are:
normal
small-caps</td></tr>
<tr><td>font-weight</td><td>Specifies the thickness of the letters of the font. Default is normal. The values are the same as those of the CSS font-weight property. Allowed values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>normal
lighter
100
200
300
400</td><td>Treated as non-bold.</td></tr>
<tr><td>bold
bolder
500
600
700
800
900</td><td>Treated as bold.</td></tr>
</table>
</td></tr>
<tr><td>mso-text-</td><td>Specifies whether a shadow is applied to the text on a text path.</td></tr>
</table>
</td></tr>
</table>

Attributes	Description
	shadow: Default is false.
	text-decoration: Specifies the style of text decoration. Default is none. The values are the same as those of the CSS text-decoration property. Allowed values are: • none • underline • overline • line-through • blink
	v-rotate-letters: Specifies whether the letters of the text are rotated counterclockwise by 90 degrees. Default is false.
	v-same-letter-heights: Specifies whether all letters will be the same height regardless of initial case. If true, the lowercase letters are stretched to the height of the uppercase letters. Default is false.
	v-text-align: Specifies the alignment of text. Default is left. Allowed values are: • left • right • center • justify • letter-justify - Distributes the extra space between the letters. • stretch-justify - Stretches the letters to fill in the space.
	v-text-kern: Specifies whether kerning is turned on. Default is false.
	v-text-reverse: Specifies whether the layout order of rows is reversed. Default is false. This is used for vertical text layout.
	v-text-spacing-mode: Specifies the mode for letter spacing. Default is tightening. This property determines whether space will be removed between each letter (tightening) or added between each letter (tracking). The amount of letter spacing change is defined by the v-text-spacing property. Allowed values are: • tightening • tracking
	v-text-spacing: Specifies the amount of spacing for text in 100ths of single line spacing. Default is 100.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>The line (§6.1.2.12), polyline (§6.1.2.15) and curve (§6.1.2.3) elements ignore the following properties:

• top
• left
• width
• height

The following properties are not inherited by an element that references a shapetype element (§6.1.2.20) via the id attribute:

• flip
• height
• left
• margin-left
• margin-top
• position
• rotation
• top
• visibility
• width
• z-index

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>target (Hyperlink Display Target)</td><td>Specifies a frame or window that a URL is displayed in. Default is no value. Allowed values are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td><targetname></td><td>String containing the name of the frame or window in which to load the document.</td></tr>
<tr><td>_blank</td><td>Specifies that the linked document is loaded into a new blank window. This window is not named.</td></tr>
<tr><td>_media</td><td>Specifies that the linked document is loaded into the Media Bar. Available in Microsoft® Internet Explorer 6 or later.</td></tr>
<tr><td>_parent</td><td>Specifies that the linked document is loaded into the immediate parent of the document containing the link.</td></tr>
<tr><td>_search</td><td>Specifies that the linked document is loaded into the browser's search pane. Available in Microsoft Internet Explorer 5 or greater.</td></tr>
<tr><td>_self</td><td>Specifies that the linked document is loaded into the window in which the link was clicked (the active window).</td></tr>
<tr><td>_top</td><td>Specifies that the linked document is loaded into the topmost window.</td></tr>
</table>
</td></tr>
</table>

Attributes	Description

	[*Example:*
	
	`<v:shape ...`
	`  href="http://www.openxmlformats.org"`
	`  target="_self" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

title (Shape Title)	Specifies the text displayed when the mouse pointer moves over the shape. Default is no value.
	
	[*Example:*
	
	`<v:shape ... title="tooltip" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

userdrawn (Exists In Master Slide)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies whether the user has added the shape to a master slide. Default is `false`. Used by PresentationML.
	
	[*Example:*
	
	`<v:shape ... o:userdrawn="true" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

userhidden (Hide Script Anchors)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies whether a script anchor is hidden. Default is `false`. If `true`, script anchors stay hidden even if the shape is otherwise visible. A script anchor is the visual representation of a script that when displayed in an application.
	
	[*Example:*
	
	`<v:shape ... o:userhidden="true" ... >`
	`</v:shape>`
	
	end example]

Attributes	Description
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

wrapcoords (Shape Bounding Polygon)	Specifies the bounding polygon that surrounds a shape. This is specified using a comma-delimited list of x and y coordinates: "x1,y1,x2,y2,x3,y3,..." This is used when text is tightly wrapped around a shape. Default is no value until the mso-wrap-mode style attribute is set to tight or through.
	
	[*Example:*
	
	`<v:shape ...`
	`  wrapcoords="0,0 0,200, 200,200, 200,0" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Arc">
   <sequence>
      <group ref="EG_ShapeElements" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attributeGroup ref="AG_AllCoreAttributes"/>
   <attributeGroup ref="AG_AllShapeAttributes"/>
   <attribute name="startAngle" type="xsd:decimal" use="optional"/>
   <attribute name="endAngle" type="xsd:decimal" use="optional"/>
</complexType>
```

6.1.2.2 background (Document Background)

This element describes the fill of the background of a page using vector graphics fills. Fills consist of simple colors, more advanced effects defined through the fill element (§6.1.2.5), or images.

[*Example:* The following shades the page background a pale red:

```
<v:background fillcolor="#c0504d">
</v:background>
```

This uses the fill element (§6.1.2.5) to create a gradient background fill:

```
<v:background>
 <v:fill type="gradient" color="#c0504d" color2="#f0504d" angle="45"/>
</v:background>
```

end example]

Parent Elements

Parent Elements
background (§2.2.1); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23)

Child Elements	Subclause
fill (Shape Fill Properties)	§6.1.2.5

Attributes	Description

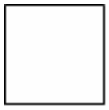
bwmode (Black-and-White Mode)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies how a shape will render for black-and-white output devices. When a shape is printed on a black-and-white printer or displayed in a black-and-white view in an application, several options are possible. Default is auto, which will use o:bwnormal for normal black-and-white rendering and o:bwpure for pure black-and-white rendering
	
	[*Example:* This shape renders in grayscale in a black-and-white environment:
	
	`<v:shape ... o:bwmode="grayscale" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).

bwnormal (Normal Black-and-White Mode)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies the black-and-white mode for normal black-and-white output devices. Default is auto.
	
	[*Example:* This shape renders in a pale grayscale in a normal black-and-white environment:
	
	`<v:shape ...`
	`  o:bwmode="lightgrayscale" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).

bwpure (Pure Black-and-White Mode)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies the black-and-white mode for pure black-and-white output devices. Default is auto.
	
	[*Example:* This shape renders in high contrast when in a pure black-and-white environment:
	
	`<v:shape ... o:bwmode="highcontrast" ... >`
	`</v:shape>`

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).

fillcolor (Fill Color)	Specifies the color to use for the fill. Default is `white`. If the fill element (§6.1.2.5) is present, its color attribute takes precedence. Colors are typically specified as either a named color, such as `red`, or six hexadecimal digits representing the red, green and blue values of the color, such as `#00FF30`. Full details are specified in the simple type description.
	
	[*Example:* This shape is red if its fill is visible:
	
	`<v:shape ... fillcolor="red" ... ></v:shape>`
	
	This is equivalent to:
	
	`<v:shape ... fillcolor="#ff0000" ... ></v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).

filled (Shape Fill Toggle)	Specifies whether the closed path will be filled. Default is `true`. This attribute is overridden by the fill on attribute. [*Example:* `<v:shape ... filled="f" fillcolor="red" ...></v:shape>`  *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
id (Unique Identifier)	Specifies a unique identifier that can be used to reference a VML object. Default is no value.

Attributes	Description
	[*Example:* `<v:shape ... id="myShape" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
targetscreensize (Target Screen Size) Namespace: urn:schemas-microsoft-com:office:office	Specifies the target resolution used for WordprocessingML documents with a gradient or picture filled background. Default is no value. Allowed values are: • 544,376 • 640,480 • 720,512 • 800,600 • 1024,768 • 1152,862 The possible values for this attribute are defined by the ST_ScreenSize simple type (§6.2.3.22).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Background">
   <sequence>
      <element ref="fill" minOccurs="0"/>
   </sequence>
   <attributeGroup ref="AG_Id"/>
   <attributeGroup ref="AG_Fill"/>
   <attribute ref="o:bwmode"/>
   <attribute ref="o:bwpure"/>
   <attribute ref="o:bwnormal"/>
   <attribute ref="o:targetscreensize"/>
</complexType>
```

6.1.2.3 curve (Bezier Curve)

This element is used to draw a cubic bézier curve.

The following properties of the style attribute are ignored:

- top
- margin-top
- center-y
- left
- margin-left
- center-x

- width
- height

[*Example:* The following specifies a simple curve that opens upward:

```
<v:curve id="mycurve"
from="10pt,10pt" to="100pt,10pt"
control1="40pt,30pt" control2="85pt,30pt">
</v:curve>
```

This shape is created:

end example]

Parent Elements
background (§2.2.1); group (§6.1.2.7); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23)

Child Elements	Subclause
anchorlock (Anchor Location Is Locked)	§6.3.2.1
borderbottom (Bottom Border)	§6.3.2.2
borderleft (Left Border)	§6.3.2.3
borderright (Right Border)	§6.3.2.4
bordertop (Top Border)	§6.3.2.5
callout (Callout)	§6.2.2.2
ClientData (Attached Object Data)	§6.4.2.12
clippath (Shape Clipping Path)	§6.2.2.3
extrusion (3D Extrusion)	§6.2.2.10
fill (Shape Fill Properties)	§6.1.2.5
formulas (Set of Formulas)	§6.1.2.6
handles (Set of Handles)	§6.1.2.9
imagedata (Image Data)	§6.1.2.11
lock (Shape Protections)	§6.2.2.17
path (Shape Path)	§6.1.2.14
shadow (Shadow Effect)	§6.1.2.18
signatureline (Digital Signature Line)	§6.2.2.29
skew (Skew Transform)	§6.2.2.30
stroke (Line Stroke Settings)	§6.1.2.21

Child Elements	Subclause
textbox (Text Box)	§6.1.2.22
textdata (VML Diagram Text)	§6.5.2.2
textpath (Text Layout Path)	§6.1.2.23
wrap (Text Wrapping)	§6.3.2.6

Attributes	Description
allowincell (Allow in Table Cell) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape can be placed in a table. Default is `false`. [*Example:* `<v:shape ... o:allowincell="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
allowoverlap (Allow Shape Overlap) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape can overlap another shape. Default is `true`. If `false`, the shape will shift left or right so as not to overlap another shape, similar to the behavior of the HTML float attribute. [*Example:* `<v:shape ... o:allowoverlap="false" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

alt (Alternate Text)	Specifies alternative text describing the graphical object. This text should provide a brief description of the shape for use by accessibility tools. Default is no value.
	
	[*Example:* The alt text describes the basic shape:
	
	`<v:shape ... fillcolor="red" alt="Red rectangle"></v:shape>`
	
	The alt text describes the contents of a shape displaying an image:
	
	`<v:shape ... alt="Picture of a sunset"></v:shape>`

Attributes	Description
	end example] The possible values for this attribute are defined by the XML Schema string datatype.
borderbottomcolor (Bottom Border Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the bottom border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:borderbottomcolor="red" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
borderleftcolor (Border Left Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the left border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:borderleftcolor="red" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
borderrightcolor (Border Right Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the right border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:borderrightcolor="red" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
bordertopcolor (Border Top Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the top border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:bordertopcolor="red" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
bullet (Graphical	Specifies whether the shape is a graphical bullet. Default is `false`.

Attributes	Description
Bullet) Namespace: urn:schemas-microsoft-com:office:office	[*Example:* `<v:shape ... o:bullet="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
button (Button Behavior Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape will exhibit button press behavior on click. Default is `false`. [*Example:* `<v:shape ... o:button="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
bwmode (Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies how a shape will render for black-and-white output devices. When a shape is printed on a black-and-white printer or displayed in a black-and-white view in an application, several options are possible. Default is `auto`, which will use o:bwnormal for normal black-and-white rendering and o:bwpure for pure black-and-white rendering. bwnormal and bwpure are subordinate to bwmode. If bwmode is "auto" then the value for bwnormal or bwpure is used depending on what the output format is. An application may define for itself what, if any, difference there is between normal B&W and pure B&W. For example, normal B&W might allow greyscale and pure B&W might not. [*Example:* This shape renders in grayscale in a black-and-white environment: `<v:shape ... o:bwmode="grayscale" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
bwnormal (Normal Black-and-White Mode) Namespace: urn:schemas-	Specifies the black-and-white mode for normal black-and-white output devices. Default is `auto`. [*Example:* This shape renders in a pale grayscale in a normal black-and-white environment:

Attributes	Description
microsoft-com:office:office	<v:shape ... o:bwmode="auto" o:bwnormal="lightgrayscale" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
bwpure (Pure Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies the black-and-white mode for pure black-and-white output devices. Default is auto. [*Example:* This shape renders in high contrast when in a pure black-and-white environment: <v:shape ... o:bwmode="auto" o:bwpure="highcontrast" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
chromakey (Image Transparency Color)	Specifies a color value that will be transparent and show anything behind the shape. Default is no value. [*Example:* <v:image ... chromakey="white" ...> </v:image> *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).
class (CSS Reference)	Specifies a reference to the definition of a CSS style. Default is no value. [*Example:* The snippets below are equivalent: ... .narrowstyle {width:50;height:100} ... <v:shape ... class="narrowstyle" style="top:1;left:1"> </v:shape> <v:shape ... style="top:1;left:1; width:50;height:100">

Attributes	Description
	</v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
clip (Clipping Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies that the clipping region is active. This is used in conjunction with the clippath (§6.2.2.3) element to create a clipping region. [*Example:* <v:shape ... o:clip="true"> </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
cliptowrap (Clip to Wrapping Polygon) Namespace: urn:schemas-microsoft-com:office:office	Specifies that the clipping region for the shape aligns with the wrapping polygon that tightly surrounds the entire shape (essentially, that the shape shall not be drawn beyond its wrapping polygon's extents – if it does, the shape shall be clipped). Default is `false`. [*Example:* <v:shape ... o:cliptowrap="true"> </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
connectortype (Shape Connector Type) Namespace: urn:schemas-microsoft-com:office:office	Specifies the type of connector used for joining shapes. Default is `straight`. [*Example:* <v:shape ... o:connectortype="elbow" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_ConnectorType simple type (§6.2.3.6).
control1 (First Curve Control Point)	Specifies the first control point for the curve, given in the coordinate space of the parent element. Default is "10,10". If the parent is not a VML element, the default unit is a pixel. Allowed units are in, cm, mm, pt, pc and px.

Attributes	Description
	[*Example:* `<v:curve ... control1="20,30" ... ></v:curve>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
control2 (Second Curve Control Point)	Specifies the second control point for the curve, given in the coordinate space of the parent element. Default is "20,0". If the parent is not a VML element, the default unit is a pixel. Allowed units are in, cm, mm, pt, pc and px. [*Example:* `<v:curve ... control2="50,20" ... ></v:curve>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
coordorigin (Coordinate Space Origin)	Specifies the coordinate of the top left corner of the shape's coordinate space. This determines the position of the (0,0) coordinate space origin within the shape's bounding box. Default is "0,0", which places the (0,0) origin at the top left corner of the bounding box. This affects shape properties that specify coordinate positions, such as the path attribute. Thus a path can be defined against a generic (0,0) origin and the coordorigin value translates the entire path within the shape's bounding space. [*Example:* The horizontal and vertical coordinate space ranges from -100 to +100 because the coordinate space (coordsize) is 200 by 200 and the top left coordinate is (-100,-100). The (0,0) origin lies at the center of the shape's bounding box, as evidenced by the position of the shape's path within the coordinate space: `<v:shape ... coordsize="200,200" coordorigin="-100,-100" path="m 0,0 l 0,50, 50,50, 50,0 x e"></v:shape>`  *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>coordsize
(Coordinate Space Size)</td><td>Specifies the size of the shape's coordinate space in coordinate units. Default is "1000,1000".

The physical size of a coordinate unit length is determined by both the size of the coordinate space (coordsize) and the size of the shape (style width and height). The coordsize attribute defines the number of horizontal and vertical subdivisions into which the shape's bounding box is divided. The combination of coordsize and style width/height effective scales the shape anisotropically.

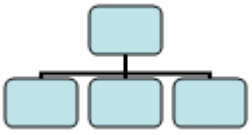
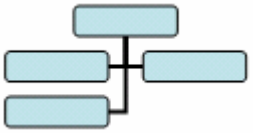
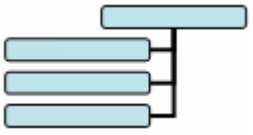
[Example: The path is 50 units wide and tall, which is 25% of the size of the coordinate space:

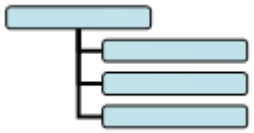
<pre><v:shape ... coordsize="200,200"
  coordorigin="-100,-100"
  path="m 0,0 l 0,50, 50,50, 50,0 x e">
</v:shape></pre>
 

end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>dgmlayout
(Diagram Node Layout Identifier)

Namespace: urn:schemas-microsoft-com:office:office</td><td>Specifies the type of automatic layout to apply to the diagram node. This is only meaningful for organization charts. Valid values for organization charts are:

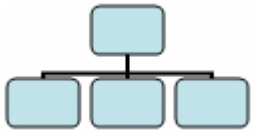
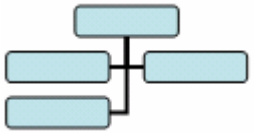
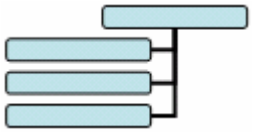
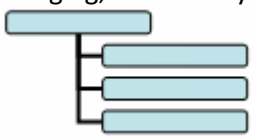
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Top-down, centered layout.
  </td></tr>
<tr><td>1</td><td>Hanging, both sides layout.
  </td></tr>
<tr><td>2</td><td>Hanging, right side layout.
  </td></tr>
<tr><td>3</td><td>Hanging, left side layout.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>  </td></tr>
</table>

[Example:

<pre><v:shape ... dgmlayout="1">
</v:shape></pre>
end example]

The possible values for this attribute are defined by the XML Schema integer datatype.
</td></tr>
<tr><td>dgmlayoutmru (Diagram Node Recent Layout Identifier)

Namespace: urn:schemas-microsoft-com:office:office</td><td>Specifies the type of automatic layout most recently used on the child elements of the diagram node. This is only meaningful for organization charts. Valid values for organization charts are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Top-down, centered layout.
  </td></tr>
<tr><td>1</td><td>Hanging, both sides layout.
  </td></tr>
<tr><td>2</td><td>Hanging, right side layout.
  </td></tr>
<tr><td>3</td><td>Hanging, left side layout.
  </td></tr>
</table>

[Example:

<pre><v:shape ... dgmlayout="1">
</v:shape></pre>
end example]</td></tr>
</table>

Attributes	Description
	The possible values for this attribute are defined by the XML Schema integer datatype.

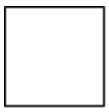
dgmnodekind (Diagram Node Identifier)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies an optional, application-specific parameter that is intended to be used by the application to tag different types of nodes in a diagram.
	
	[*Example:*
	
	`<v:shape ... dgmnodekind="1">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema integer datatype.

doubleclicknotify (Double-click Notification Toggle)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies that an event message is sent when a shape is double-clicked. Default is `false`.
	
	[*Example:*
	
	`<v:shape ... o:doubleclicknotify="true" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

fillcolor (Fill Color)	Specifies the color to use for the fill. Default is `white`. If the fill element (§6.1.2.5) is present, its color attribute takes precedence. Colors are typically specified as either a named color, such as `red`, or six hexadecimal digits representing the red, green and blue values of the color, such as `#00FF30`. Full details are specified in the simple type description.
	
	[*Example:* This shape is red if its fill is visible:
	
	`<v:shape ... fillcolor="red" ... >`
	`</v:shape>`
	
	This is equivalent to:
	
	`<v:shape ... fillcolor="#ff0000" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).

filled (Shape Fill	Specifies whether the closed path will be filled. Default is `true`. This attribute is

Attributes	Description

Toggle)	overridden by the fill on attribute.
	
	[*Example:*
	
	`<v:shape ... filled="f"`
	`  fillcolor="red" ...>`
	`</v:shape>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).

forcedash (Force Dashed Outline)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies whether a dashed outline is used to draw a shape when a shape has no line or fill. Default is `false`.
	
	Used by PresentationML placeholders to draw a dashed outline when there is no line and no fill for a shape.
	
	[*Example:*
	
	`<v:shape ... o:forcedash="true" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

from (Curve Starting Point)	Specifies the starting point of the line in the coordinate space of the parent element. Default is "0,0". If the parent is not a VML element, the default unit is a pixel. Allowed units are in, cm, mm, pt, pc and px.
	
	[*Example:*
	
	`<v:curve ... from="10,10" ... >`
	`</v:curve>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

hr (Horizontal Rule Toggle)	Specifies that a shape is a horizontal rule. Default is `false`.

Attributes	Description
Namespace: urn:schemas-microsoft-com:office:office	[*Example:* `<v:shape ... o:hr="true" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
hralign (Horizontal Rule Alignment) Namespace: urn:schemas-microsoft-com:office:office	Specifies the alignment of a horizontal rule. Default is `left`. [*Example:* `<v:shape ... o:hralign="center" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_HrAlign simple type (§6.2.3.14).
href (Hyperlink Target)	Specifies a hyperlink URL target for the shape. Default is no value. [*Example:* `<v:shape ... href="http://www.openxmlformats.org" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
hrnoshade (Horizontal Rule 3D Shading Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies that the horizontal rule does not have 3-D shading. Default is `false`. [*Example:* `<v:shape ... o:hrnoshade="true" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
hrpct (Horizontal Rule Length Percentage)	Specifies the length of a horizontal rule as a percentage of page width. Default is 0. [*Example:*

Attributes	Description
Namespace: urn:schemas-microsoft-com:office:office	<v:shape ... o:hrpct="85" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema float datatype.
hrstd (Horizontal Rule Standard Display Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape is displayed as a standard horizontal rule. Only applies if hr is true. Default is false. [*Example:* <v:shape ... o:hrstd="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
id (Unique Identifier)	Specifies a unique identifier that can be used to reference a VML object. Default is no value. [*Example:* <v:shape ... id="myShape" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
insetmode (Text Inset Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the application calculates the internal text margin instead of using the inset attribute. Default is custom. This attribute is only meaningful for text boxes. [*Example:* <v:shape ... o:insetmode="auto" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_InsetMode simple type (§6.2.3.15).
insetpen (Inset Border From Path)	Specifies that the border shall be displayed inside of the path defining the shape, rather than along the path (the default border placement), or outside of the path as might be done with an image.

Attributes	Description
	[*Example:* `<v:shape ... insetpen="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
ole (Embedded Object Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the shape is an embedded object. Default is `false`. [*Example:* `<v:shape ... o:ole="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalseBlank simple type (§6.2.3.24).
oleicon (Embedded Object Icon Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether an embedded object will be displayed as an icon. Default is `false`. [*Example:* `<v:shape ... o:oleicon="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
oned (Shape Handle Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the extra handles of a shape are hidden. If `true`, hides all shape handles except the top left and bottom right; that is, the same handles that are used for a straight line segment. Default is `false`. [*Example:* `<v:shape ... o:oned="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

Attributes	Description

opacity (Fill Color Opacity)	Specifies the opacity of the primary fill color. Default is 1.0. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8.
	
	[*Example:* The red color is 25% opaque:
	
	`<v:fill type="gradient" color="red"`
	`  color2="blue" opacity=".25">`
	`</v:fill>`
	
	  `opacity="1"`
	  `opacity=".25"`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

preferrelative (Relative Resize Toggle)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies whether the original size of an object is saved after reformatting. Default is `false`. If `true`, the original size of the object is stored and all resizing is based on a percentage of that original size. Otherwise, each resizing will reset the scale to 100%.
	
	[*Example:*
	
	`<v:shape ... o:preferrelative="true" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

print (Print Toggle)	Specifies whether the shape will be printed. Default is `true`.
	
	[*Example:*
	
	`<v:shape ... print="false" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).

regroupid (Regroup	Specifies a previous group for a shape. An ID number is used to identify groups of shapes

Attributes	Description
ID) Namespace: urn:schemas-microsoft-com:office:office	that are no longer grouped. This allows shapes to be regrouped programmatically. [*Example:* The shape was part of a group identified by the ID 040754: `<v:shape ... o:regroupid="040754" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema integer datatype.
spid (Optional String) Namespace: urn:schemas-microsoft-com:office:office	Specifies an optional string that an application may use to Identify the particular shape. Default is no value. The possible values for this attribute are defined by the XML Schema string datatype.
spt (Optional Number) Namespace: urn:schemas-microsoft-com:office:office	Specifies an optional number that an application may use to associate the particular shape with a defined shape type. Default is 0. The possible values for this attribute are defined by the XML Schema float datatype.
strokecolor (Shape Stroke Color)	Specifies the primary color of the brush to use to stroke the path of the shape. Default is `black`. The color attribute of the stroke element (§6.1.2.21) overrides this. Colors are typically specified as either a named color, such as `red`, or six hexadecimal digits representing the red, green and blue values of the color, such as `#00FF30`. Full details are specified in the simple type description. [*Example:* `<v:shape ... strokecolor="red" ...> </v:shape>`  *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).
stroked (Shape Stroke Toggle)	Specifies whether the path defining the shape is stroked with a solid line. The stroke element (§6.1.2.21) defines other strokes. The on attribute of the stroke element

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>overrides this attribute. Default is <code>true</code>.

[Example:

<pre><v:shape ... fillcolor="red"
  stroked="false" strokecolor="blue"...>
</v:shape></pre>
 

end example]

The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).</td></tr>
<tr><td>strokeweight (Shape Stroke Weight)</td><td>Specifies the width of the brush to use to stroke the path. Default is 1 point. If a number is given without units, the emu is used. The weight attribute of the stroke element (§6.1.2.21) overrides this attribute.

[Example:

<pre><v:shape ... strokeweight="3pt" ... >
</v:shape></pre>
 

end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>style (Shape Styling Properties)</td><td>Specifies the CSS2 styling properties of the shape. This uses the syntax described in the "Visual formatting model" of the Cascading Style Sheets, Level 2 specification, a Recommendation of the World Wide Web Consortium available here: http://www.w3.org/TR/REC-CSS2. Full descriptions of each property are not repeated here, but the VML treatment of each property is defined. Allowed properties include:

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td><code>flip</code></td><td>Specifies that the orientation of a shape is flipped. Default is no value. Allowed values are:

• x - Flip along the y-axis, reversing the x-coordinates.
• y - Flip along the x-axis, reversing the y-coordinates.
• xy - Flip along both the y- and x-axis.</td></tr>
</table>
</td></tr>
</table>

Attributes	Description
	• yx - Flip along both the x- and y-axis.
height	Specifies the height of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
left	Specifies the position of the left of the containing block of the shape relative to the element left of it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are: • auto - Default position of an element in the flow of the page. • <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
margin-bottom	Specifies the position of the bottom of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
margin-left	Specifies the position of the left of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page.

Attributes	Description
	<table><tr><td></td><td> <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. <percentage>- Value expressed as a percentage of the parent object's width. </td></tr><tr><td>margin-right</td><td>Specifies the position of the right of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: - auto - Default position of an element in the flow of the page. <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. <percentage>- Value expressed as a percentage of the parent object's width. </td></tr><tr><td>margin-top</td><td>Specifies the position of the top of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: - auto - Default position of an element in the flow of the page. <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. <percentage>- Value expressed as a percentage of the parent object's height. </td></tr><tr><td>mso-position-horizontal</td><td>Specifies the horizontal positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: - absolute - left - center - right - inside - outside </td></tr><tr><td>mso-position-horizontal-relative</td><td>Specifies relative horizontal position data for objects in WordprocessingML documents. This modifies the mso-position-horizontal property. Default is text. Allowed values are:</td></tr></table>

Attributes	Description
	• margin • page • text • char
mso-position-vertical	Specifies the vertical positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute • top • center • bottom • inside • outside
mso-position-vertical-relative	Specifies relative vertical position data for objects in WordprocessingML documents. This modifies the mso-position-vertical property. Default is text. Allowed values are: • margin • page • text • line
mso-wrap-distance-bottom	Specifies the distance from the bottom of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-distance-left	Specifies the distance from the left side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-distance-right	Specifies the distance from the right side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-distance-top	Specifies the distance from the top of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin

Attributes	Description
	of the shape to include the margin areas. This property does not change the origin.
mso-wrap-edited	Specifies whether the wrap coordinates were customized by the user. If the wrap coordinates are generated by an editor, this property is true; otherwise they were customized by a user. Default is false.
mso-wrap-style	Specifies the wrapping mode for text in shapes in WordprocessingML documents. Default is square. Allowed values are: • square - Wraps text inside the shape in a square. • none - Text does not wrap.
position	Specifies the type of positioning used to place an element. Default is static. When the element is contained inside a group, this property must be absolute. Allowed values are: • static - The element is positioned according to the normal flow of the page. The top and left properties are ignored. If the object is anchored inline, this value is used. • absolute - The element is positioned relative to the parent, using the top and left properties. • relative - The element is positioned according to the normal flow of the page, but the top and left properties are used. The overlap of overlapping elements is governed by the z-index property.
rotation	Specifies the angle that a shape is rotated, in degrees. Default is 0. Positive angles are clockwise.
top	Specifies the position of the top of the containing block of the shape relative to the element above it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
visibility	Specifies whether a shape is displayed. Only inherit and hidden are used; any other values are mapped to inherit. Default is

Attributes	Description
	`inherit`. Allowed values are: • `hidden` - The shape is not visible, but is still part of the flow of the objects in the browser. Mouse events are not processed. • `inherit` - The visibility state is inherited from the parent of the shape.
	`width` — Specifies the width of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	`z-index` — Specifies the display order of overlapping shapes. Default is 0. This property shall not be used for shapes anchored inline. Allowed values are: • `auto` - Uses the order that the shapes appear in the page, bottom to top. • <order>- A number that represents the stacking precedence. Shapes with higher numbers are placed on top of those with lower numbers. Negative numbers are allowed.

The following properties are only used by the textbox element (§6.1.2.22):

Property	Description
`direction`	Specifies the direction of the text in the textbox. Default is `ltr`. This property is superceded by the `mso-direction-alt` property if that is specified. Allowed values are: • `ltr` - Test is displayed left-to-right. • `rtl` - Test is displayed right-to-left.
`layout-flow`	Determines the flow of the text layout in a textbox. Default is `horizontal`. Allowed values are: • `horizontal` - Text is displayed horizontally.

Attributes	Description
	• `vertical` - Text is displayed vertically. • `vertical-ideographic` - Ideographic text is displayed vertically. • `horizontal-ideographic` - Ideographic text is displayed horizontally.
`mso-direction-alt`	Specifies an alternate direction for text in textboxes. Overrides the `direction` property. The only allowed value is `context`.
`mso-fit-shape-to-text`	Specifies whether the shape stretches to fit the text in the textbox. Default is `false`.
`mso-fit-text-to-shape`	Specifies whether the text stretches to fit the textbox. Default is `false`.
`mso-layout-flow-alt`	Specifies the alternate layout flow for text in textboxes. This property is used instead of `layout-flow` when the layout flow is from bottom to top for non-ideographic languages. Its only value is `bottom-to-top`.
`mso-next-textbox`	Specifies the ID of the next textbox in a series. Used to keep track of a set of linked textboxes. Default is no value.
`mso-rotate`	Specifies a specific rotation value for text in a textbox. Default is 0. Allowed values are: • 0 • 90 • 180 • -90
`mso-text-scale`	Specifies the scaling factor for fitting text to shapes. Default is 0. This property is only used if `mso-fit-text-to-shape` is `true`.
`v-text-anchor`	Specifies the vertical anchoring of text in a textbox. Default is `top`. The alignment of a text anchor only becomes evident if `mso-fit-text-to-shape` is `false`. This property is different from the `vertical-align` CSS property, which is used for ideographic languages. Allowed values are: • `top` • `middle` • `bottom` • `top-center` • `middle-center` • `bottom-center`

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>

top-baseline
bottom-baseline
top-center-baseline
bottom-center-baseline

</td></tr>
</table>
The following properties are only used by the textpath element (§6.1.2.23):
<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>font</td><td>Specifies a compound value of font settings. Default is no value. The values are the same as those of the CSS font property. The order of definitions in the string is: font-style, font-variant, font-weight, font-size, line-height, font-family.</td></tr>
<tr><td>font-family</td><td>Specifies the family of the font. Default is no value. The values are the same as those of the CSS font-family property.</td></tr>
<tr><td>font-size</td><td>Specifies the size of the font. Default is no value. The font size is defined in points. The values are the same as those of the CSS font-size property.</td></tr>
<tr><td>font-style</td><td>Specifies the amount of slant for a font. Default is normal. The values are the same as those of the CSS font-style property. Allowed values are:
normal
italic
oblique - Treated the same as italic.</td></tr>
<tr><td>font-variant</td><td>Specifies the variant style of a font. Default is normal. The values are the same as those of the CSS font-variant property. Allowed values are:
normal
small-caps</td></tr>
<tr><td>font-weight</td><td>Specifies the thickness of the letters of the font. Default is normal. The values are the same as those of the CSS font-weight property. Allowed values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>normal
lighter
100
200</td><td>Treated as non-bold.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>
<table>
<tr><td>300

400</td><td></td></tr>
<tr><td>bold

bolder

500

600

700

800

900</td><td>Treated as bold.</td></tr>
</table>
</td></tr>
<tr><td>mso-text-shadow</td><td>Specifies whether a shadow is applied to the text on a text path. Default is false.</td></tr>
<tr><td>text-decoration</td><td>Specifies the style of text decoration. Default is none. The values are the same as those of the CSS text-decoration property. Allowed values are:

• none
• underline
• overline
• line-through
• blink</td></tr>
<tr><td>v-rotate-letters</td><td>Specifies whether the letters of the text are rotated counterclockwise by 90 degrees. Default is false.</td></tr>
<tr><td>v-same-letter-heights</td><td>Specifies whether all letters will be the same height regardless of initial case. If true, the lowercase letters are stretched to the height of the uppercase letters. Default is false.</td></tr>
<tr><td>v-text-align</td><td>Specifies the alignment of text. Default is left. Allowed values are:

• left
• right
• center
• justify
• letter-justify - Distributes the extra space between the letters.
• stretch-justify - Stretches the letters to fill in the space.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>v-text-kern</td><td>Specifies whether kerning is turned on. Default is false.</td></tr>
<tr><td>v-text-reverse</td><td>Specifies whether the layout order of rows is reversed. Default is false. This is used for vertical text layout.</td></tr>
<tr><td>v-text-spacing-mode</td><td>Specifies the mode for letter spacing. Default is tightening. This property determines whether space will be removed between each letter (tightening) or added between each letter (tracking). The amount of letter spacing change is defined by the v-text-spacing property. Allowed values are:

• tightening
• tracking</td></tr>
<tr><td>v-text-spacing</td><td>Specifies the amount of spacing for text in 100ths of single line spacing. Default is 100.</td></tr>
</table>

The line (§6.1.2.12), polyline (§6.1.2.15) and curve (§6.1.2.3) elements ignore the following properties:

• top
• left
• width
• height

The following properties are not inherited by an element that references a shapetype element (§6.1.2.20) via the id attribute:

• flip
• height
• left
• margin-left
• margin-top
• position
• rotation
• top
• visibility
• width
• z-index

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>target (Hyperlink Display Target)</td><td>Specifies a frame or window that a URL is displayed in. Default is no value. Allowed values are:

<table>
<tr><th>Value</th><th>Description</th></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td><targetname></td><td>String containing the name of the frame or window in which to load the document.</td></tr>
<tr><td>_blank</td><td>Specifies that the linked document is loaded into a new blank window. This window is not named.</td></tr>
<tr><td>_media</td><td>Specifies that the linked document is loaded into the Media Bar. Available in Microsoft® Internet Explorer 6 or later.</td></tr>
<tr><td>_parent</td><td>Specifies that the linked document is loaded into the immediate parent of the document containing the link.</td></tr>
<tr><td>_search</td><td>Specifies that the linked document is loaded into the browser's search pane. Available in Microsoft Internet Explorer 5 or greater.</td></tr>
<tr><td>_self</td><td>Specifies that the linked document is loaded into the window in which the link was clicked (the active window).</td></tr>
<tr><td>_top</td><td>Specifies that the linked document is loaded into the topmost window.</td></tr>
</table>
[Example:
<pre>
  <v:shape ...
    href="http://www.openxmlformats.org"
    target="_self" ... >
  </v:shape>
</pre>
end example]
The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>title (Shape Title)</td><td>
Specifies the text displayed when the mouse pointer moves over the shape. Default is no value.
[Example:
<pre>
  <v:shape ... title="tooltip" ... >
  </v:shape>
</pre>
end example]
The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>to (Curve Ending Point)</td><td>
Specifies the ending point of the line in the coordinate space of the parent element. Default is "30,20". If the parent is not a VML element, the default unit is a pixel. Allowed units are in, cm, mm, pt, pc and px.
[Example:
<pre>
  <v:curve ... to="40,40" ... >
</pre>
</td></tr>
</table>

Attributes	Description
	</v:curve> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
userdrawn (Exists In Master Slide) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the user has added the shape to a master slide. Default is `false`. Used by PresentationML. [*Example:* <v:shape ... o:userdrawn="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
userhidden (Hide Script Anchors) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a script anchor is hidden. Default is `false`. If `true`, script anchors stay hidden even if the shape is otherwise visible. A script anchor is the visual representation of a script that when displayed in an application. [*Example:* <v:shape ... o:userhidden="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
wrapcoords (Shape Bounding Polygon)	Specifies the bounding polygon that surrounds a shape. This is specified using a comma-delimited list of x and y coordinates: "x1,y1,x2,y2,x3,y3,..." This is used when text is tightly wrapped around a shape. Default is no value until the mso-wrap-mode style attribute is set to `tight` or `through`. [*Example:* <v:shape ... wrapcoords="0,0 0,200, 200,200, 200,0" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Curve">
   <sequence>
      <group ref="EG_ShapeElements" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attributeGroup ref="AG_AllCoreAttributes"/>
   <attributeGroup ref="AG_AllShapeAttributes"/>
   <attribute name="from" type="xsd:string" use="optional"/>
   <attribute name="control1" type="xsd:string" use="optional"/>
   <attribute name="control2" type="xsd:string" use="optional"/>
   <attribute name="to" type="xsd:string" use="optional"/>
</complexType>
```

6.1.2.4 f (Single Formula)

This element defines a single value as the result of the evaluation of an expression. The expression is defined by the eqn attribute and has the general form of an operation followed by up to three arguments, which consist of adjustment values (see the adj attribute of the shape element (§6.1.2.19)), the results of earlier formulas, fixed numbers or pre-defined values. Each f value is referenced using "@" followed by a number corresponding to the zero-based index for that value in the list of f elements. For example, the value of the second f element is referenced as "@2".

[*Example:* The following defines a blue arrow pointing to the right:

```
<v:shape coordsize="21600,21600" adj="18000,5400,10800"
  path="m @0,0 l @0,@1 0,@1 0,@3 @0,@3 @0,21600 21600,10800 x e"
  style='left:50pt;top:50pt;width:90pt;height:30pt'
  fillcolor="#4f81bd" strokecolor="#4f81bd" strokeweight="2pt">
  <v:formulas>
   <v:f eqn="val #0"/>
   <v:f eqn="val #1"/>
   <v:f eqn="val #2"/>
   <v:f eqn="sum height 0 #1"/>
   <v:f eqn="sum #2 0 #1"/>
   <v:f eqn="sum width 0 #0"/>
   <v:f eqn="prod @5 @4 #2"/>
   <v:f eqn="sum width 0 @6"/>
  </v:formulas>
</v:shape>
```

The shape looks like this:

end example]

Parent Elements
formulas (§6.1.2.6)

Attributes	Description
eqn (Equation)	Specifies a single formula, which consists of a named operation followed by up to three parameters, typically described as v, P1 and P2. Up to 128 formulas may be specified. These operations are defined (calculation accuracy is discussed below):

Operation	Description
`val`	v Returns the supplied value. Exact.
`sum`	$\mathrm{v} + \mathrm{P1} - \mathrm{P2}$ Addition and subtraction. Exact.
`product`	$\mathrm{v} \times \mathrm{P1}/\mathrm{P2}$ Multiplication and division. Rounds up.
`mid`	$(\mathrm{v} + \mathrm{P1})/2$ Simple average. Rounds toward zero.
`abs`	$\lvert\mathrm{v}\rvert$ Absolute value. Exact.
`min`	$\min(\mathrm{v}, \mathrm{P1})$ The lesser of two values. Exact.
`max`	$\max(\mathrm{v}, \mathrm{P1})$ The greater of two values. Exact.
`if`	$\mathrm{v} > 0\ ?\ P1 : P2$ Conditional selection. Exact.
`mod`	$\sqrt{\mathrm{v}^2 + \mathrm{P1}^2 + \mathrm{P1}^2}$ Modulus. Inexact.
`atan2`	$\mathrm{atan2}(\mathrm{P1}, \mathrm{v})$ Trigonometric arc tangent of a quotient. Result is in "fd" units or fractional degrees - degrees $\times\ 2^{16}$. Inexact.
`sin`	$\mathrm{v} \times \sin(\mathrm{P1})$ Sine. Argument is in "fd" units or fractional degrees - degrees $\times\ 2^{16}$. Inexact.
`cos`	$\mathrm{v} \times \cos(\mathrm{P1})$ Cosine. Argument is in "fd" units or fractional degrees - degrees $\times\ 2^{16}$. Inexact.
`cosatan2`	$\mathrm{v} \times \cos\big(\mathrm{atan2}(\mathrm{P2}, \mathrm{P1})\big)$ Preserves full accuracy in the intermediate calculation. Inexact.

Attributes	Description

sinatan2	$v \times \sin(\text{atan2}(P2, P1))$ Preserves full accuracy in the intermediate calculation. Inexact.
sqrt	$\sqrt{v}$ Square root. Result is positive and rounds down. Inexact.
sumangle	$v + P1 \times 2^{16} - P2 \times 2^{16}$ Adds an existing angle in fd units (v) to two other angles specified in degrees. P1 and P2 are scaled by 2^{16}. Exact.
ellipse	$P2\sqrt{1 - \left(\frac{v}{P1}\right)^2}$ The eccentricity formula for an ellipse, where v is length of the semiminor axis and P1 is the length of the semimajor axis. Inexact.
tan	$v \times \tan(P1)$ Tangent. Argument is in "fd" units or fractional degrees - degrees $\times 2^{16}$. Inexact.

Formulas are evaluated to full precision, but the result is always a 32-bit integer. Formula authors should avoid formulas which are discontinuous - not only are many of the trigonometric operations inexact, the transformations within the coordinate spaces are also inexact. This can mean that a set of formulas which is discontinuous evaluates to give very different path values with the same input on two different systems.

When an operation is marked as exact then a conforming implementation must always generate the correct arithmetic answer (unless the calculations overflow internally). The product operation is required to round to the nearest integer. If the result is exactly 0.5 then it must be rounded up to the next numerically greater integer. The mid operation is required to round towards 0.

All other operations are inexact, but the implementation must round non-integral values down (towards -infinity) and should perform internal calculations with this form of rounding.

The arguments used in the evaluation of a formula are normally either fixed numbers, the result of the evaluation of a previous formula or an adjust value - the value of the corresponding entry in the shape adj attribute. Fixed numbers must be positive integral values in the range 0 to 65535 (unsigned 16-bit numbers). The following named values are defined:

Value	Description
@n	The value of formula n, where n is the zero-

Attributes	Description
	based index of the formula in the list of formulas. n must be less than the current formula index.
#n	Adjustment (adj) value n. n must be in the range 0 to 7.
width	The width defined by the coordsize attribute.
height	The height defined by the coordsize attribute.
xcenter	The x ordinate of the center of the coordinate space defined by coordorigin and coordsize.
ycenter	The y ordinate of the center of the coordinate space defined by coordorigin and coordsize.
xlimo	The x value of the limo attribute (see also the path element (§6.1.2.14)).
ylimo	The y value of the limo attribute (see also the path element (§6.1.2.14)).
hasstroke	1 if the shape has a stroke operation, 0 if it does not, as determined by the on attribute of the stroke element (§6.1.2.21).
hasfill	1 if the shape has a fill operation, 0 if it does not, as determined by the on attribute of the fill element (§6.1.2.5).
pixellinewidth	The line width in output device pixels. This is used to outset lines from the edge of a rectangle on the assumption that the implementation draws to lower right pixel in preference to the upper left pixel when a line is on a pixel boundary.
pixelwidth	The width of the shape in device pixels (i.e., the coordsize width transformed into device space).
pixelheight	The height of the coordsize in device pixels.
emuwidth	The width of the coordsize in EMUs.
emuheight	The height of the coordsize in EMUs.
emuwidth2	Half the width of the coordsize in EMUs.

Attributes	Description
	emuheight2 — Half the height of the coordsize in EMUs. The EMU, or English Metric Unit, is the smallest unit of measure in VML and corresponds to 914400 EMU per inch or 12700 EMU per point. See above for an example. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_F">
   <attribute name="eqn" type="xsd:string"/>
</complexType>
```

6.1.2.5 fill (Shape Fill Properties)

This element specifies how the path should be filled if something beyond a solid color fill is desired. The attributes of the fill element can be used to describe a powerful set of image- or gradient-based fill patterns. Extensions to the VML fill definition are encoded as sub-elements of fill.

Parent Elements
arc (§6.1.2.1); background (§2.2.1); background (§6.1.2.2); curve (§6.1.2.3); group (§6.1.2.7); image (§6.1.2.10); line (§6.1.2.12); object (§2.3.3.19); oval (§6.1.2.13); pict (§2.3.3.21); pict (§2.9.23); polyline (§6.1.2.15); rect (§6.1.2.16); roundrect (§6.1.2.17); shape (§6.1.2.19); shapedefaults (§6.2.2.27); shapetype (§6.1.2.20)

Child Elements	Subclause
fill (Shape Fill Extended Properties)	§6.2.2.12

Attributes	Description
alignshape (Align Image With Shape)	Specifies whether an image will align with the shape. Default is `true`. [*Example:* The image displayed in the shape is not rotated even though the shape is rotated 30 degrees: `<v:shape coordorigin="0,0" coordsize="200,200" style="top:1;left:1;width:50; height:50;rotation:30" path="m 1,1 l 1,200, 200,200, 200,1 x e"> <v:fill alignshape="false" type="frame" src="myimage.gif">`

Attributes	Description
	</v:fill> </v:shape> Applied to a simple square the fill looks like this:  *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
althref (Alternate Image Reference Location) Namespace: urn:schemas-microsoft-com:office:office	Defines an alternate reference for an image in Macintosh PICT format. [*Example:* <v:fill ... althref="myimage.pcz" ... > </v:fill> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
angle (Gradient Angle)	Specifies the direction of a gradient. The vector of a gradient is perpendicular to the vector of the blend direction from one color to another. The default value is zero degrees, which is a horizontal vector from left to right. Positive angles rotate the gradient in a counter-clockwise direction. [*Example:* The fill is composed of a 45-degree gradient of two colors. Blue is in the top left corner and red is in the bottom right corner. <v:fill type="gradient" color="red" color2="blue" angle="45"> </v:fill>  *end example*] The possible values for this attribute are defined by the XML Schema decimal datatype.
aspect (Image Aspect Ratio)	Specifies how the fill image aspect ratio will be preserved. Default is ignore. Allowed values are:

Attributes	Description

	• `ignore` - Ignore aspect ratio.
	• `atleast` - At least as large as defined by the size attribute.
	• `atmost` - No larger than that defined by the size attribute.
	
	In each case, the size attribute will be adjusted to preserve the aspect ratio of the image.
	
	[*Example:* The image that makes up the fill is no larger than 20 points by 20 points, limiting the size of the tiles inside the shape.
	
	`<v:fill type="tile" aspect="atmost"`
	`  size="20pt,20pt" src="myimage.gif">`
	`</v:fill>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_ImageAspect simple type (§6.1.3.6).

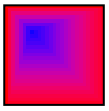
color (Primary Color)	Specifies the main fill color; functions the same as the fillcolor attribute of the shape element (§6.1.2.19). This attribute overrides the shape's fillcolor. Default is `white`. Colors are typically specified as either a named color, such as `red`, or six hexadecimal digits representing the red, green and blue values of the color, such as `#00FF30`. Full details are specified in the simple type description.
	
	[*Example:* The shape is blue:
	
	`<v:shape ... fillcolor="red" ... >`
	`  <v:fill color="blue"/>`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).

color2 (Secondary Color)	Specifies the secondary fill color, used when a fill type is a pattern or a gradient. Default is `white`. Colors are typically specified as either a named color, such as `red`, or six hexadecimal digits representing the red, green and blue values of the color, such as `#00FF30`. Full details are specified in the simple type description. [*Example:* The shape is filled with a horizontal gradient with red at the bottom and blue on top: `<v:fill type="gradient"`

Attributes	Description
	color="red" color2="blue"> </v:fill> *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).
colors (Intermediate Colors)	Specifies an array of comma-separated percentage-color pairs that define intermediate colors and their positions in a gradient fill. The primary color, specified either by the fillcolor attribute of the shape element (§6.1.2.19) or the color attribute of the fill element (§6.1.2.5), is used at the 0% endpoint. The secondary color, specified by the color2 attribute of the fill element (§6.1.2.5), is used at the 100% endpoint. The numeric values may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8. [*Example:* The shape is filled with a horizontal gradient colored, from bottom to top, red, yellow, green, blue: <v:fill type="gradient" color="red" color2="blue" colors="30% yellow,70% green"> </v:fill>  *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
detectmouseclick (Detect Mouse Click) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a mouse click is detected on the fill of a shape. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
focus (Gradient Center)	Specifies the center starting position of a gradient. Values range from 100% to -100%. Default is 0. A value of 100% or -100% reverses the direction of the gradient (in effect swapping color and color2). A value of 50% changes the gradient so that color is at both ends and color2 is in the middle. A value of -50% changes the gradient so that color2 is at both ends and color is in the middle.

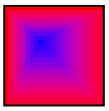
Attributes	Description

	[*Example:* The shape is filled with a horizontal gradient with red at both ends and blue in the middle:
	
	`<v:fill type="gradient"`
	`  color="red" color2="blue"`
	`  focus="50%">`
	`</v:fill>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

focusposition (Radial Gradient Center)	Specifies the position of the center rectangle of a radial gradient. The vector is a fraction of the width and height of the shape. The first is a percentage of the fill to the left edge; the second is a percentage of the fill to the top. Default is 0,0. To position a radial fill at the center of a shape, use a value of 50%,50%.
	
	[*Example:* The shape is filled with a rectangular gradient positioned in the top-left quadrant of the shape. The interior of the gradient is blue and the exterior is red:
	
	`<v:fill type="gradientradial"`
	`  color="red" color2="blue"`
	`  focusposition="25%,25%">`
	`</v:fill>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

focussize (Radial Gradient Size)	Specifies the size of the center rectangle of a radial gradient. The vector is a fraction of the width and height of the shape. The first is a percentage of the fill to the right edge; the second is a percentage of the fill to the bottom. Default is 0,0. A focussize value of 100%,100% and a focusposition of 0,0 makes color2 dominate the gradient completely. Small values of around 10%,10% are recommended for balanced gradients. [*Example:* The shape is filled with a rectangular gradient positioned in the top-left quadrant of the shape. The interior of the gradient is blue and the exterior is red. The

Attributes	Description

	red portion is wider on the bottom and right sides of the blue region. The pure blue region is 25% the width and 25% the height of the shape:
	
	`<v:fill type="gradientradial"`
	`color="red" color2="blue"`
	`focussize="25%,25%"`
	`focusposition="25%,25%">`
	`</v:fill>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

href (Hyperlink Target)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies the URL to the original image file. Used only if the picture has been linked and embedded. Default is no value.
	
	[*Example:*
	
	`<v:fill ... o:href="myimage.gif" ... >`
	`</v:fill>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

id (Unique Identifier)	Specifies a unique identifier that can be used to reference a VML object.
	
	Default is no value.
	
	[*Example:*
	
	`<v:shape ... id="myShape" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

id (Relationship to Part) Namespace: .../officeDocument/2006/relationships	Specifies the relationship ID of the relationship to the image used for this fill. The specified relationship shall be of type http://schemas.openxmlformats.org/officeDocument/2006/relationships/image or the document shall be considered non-conformant. [*Example*: The markup specifies the associated relationship part with relationship ID

Attributes	Description
	`rId10` contains the corresponding relationship information for the fill: `< ... r:id="rId10" />` *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

method (Gradient Fill Method)	Specifies the method used to generate the transition from color to color2 in a gradient fill. Default is `sigma`.
	
	[*Example:*
	
	`<v:fill type="gradient"`
	`  color="red" color2="blue" method="any">`
	`</v:fill>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_FillMethod simple type (§6.1.3.4).

on (Fill Toggle)	Specifies whether to fill the shape. Default is `true`. This attribute overrides the shape's fill attribute.
	
	[*Example:* The shape has a transparent fill:
	
	`<v:shape ... fill="true" ... >`
	`  <v:fill color="red" on="false">`
	`  </v:fill>`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).

opacity (Primary Color Opacity)	Specifies the opacity of the primary fill color. Default is 1.0. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8. [*Example:* The red color is 25% opaque:

Attributes	Description
	<v:fill type="gradient" color="red" color2="blue" opacity=".25"> </v:fill>  opacity="1"  opacity=".25" *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
opacity2 (Secondary Color Opacity) Namespace: urn:schemas-microsoft-com:office:office	Specifies the opacity of the secondary fill color. Default is 1.0. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8. [*Example:* The blue color is 25% opaque: <v:fill type="gradient" color="red" color2="blue" o:opacity2=".25"> </v:fill>  opacity2="1"  opacity2=".25" *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
origin (Fill Image Origin)	Specifies the position of the origin of a fill image as a point relative to the top left corner of the image. The vector is a fraction of the width and height of the image. Default is the center of the image. These numeric values may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8. [*Example:* The origin of the image is 25% to the right and 25% above the image's top left corner: <v:fill type="tile" src="myimage.gif" origin="0.25,-0.25"> </v:fill>

Attributes	Description
	 *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

position (Fill Image Position)	Specifies the position of the origin of a fill image as a point within its containing shape. The vector is a fraction of the width and height of the shape. These numeric values may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8.
	
	[*Example:* The origin of the image is positioned 25% to the right of the left edge of the shape and 25% down from the shape's top:
	
	`<v:fill type="tile" src="myimage.gif"`
	`  position="0.25,0.25">`
	`</v:fill>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

recolor (Recolor Fill as Picture)	Specifies that the fill uses an image. Default is `false`.
	
	[*Example:*
	
	`<v:fill r:id="rId4" o:title="MyPic" recolor="true"`
	`  type="frame">`
	`</v:fill>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).

relid (Relationship to Part) Namespace: urn:schemas-microsoft-com:office:office	Specifies the relationship ID of the relationship to the image. The specified relationship shall be of type http://schemas.openxmlformats.org/officeDocument/2006/relationships/image or the document shall be considered non-conformant. [*Example*: The markup specifies the associated relationship part with relationship ID

Attributes	Description

	`rId10` contains the corresponding relationship information:
	
	`<v:fill ... o:relid="rId10" ...>`
	`</v:fill>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_RelationshipId simple type (§6.2.3.20).

rotate (Rotate Fill with Shape)	Specifies whether the fill is rotated with the shape. Default is `false`.
	
	[*Example:* The gradient is rotated with the shape:
	
	`<v:fill color2="white" focus="100%" rotate="true"`
	`type="gradient">`
	`</v:fill>`
	
	 
	`rotate="true"`
	 
	`rotate="false"`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).

size (Fill Image Size)	Specifies the size of the fill image. Default is the native image pixel size.
	
	[*Example:* The image is reduced in size disproportionately:
	
	`<v:fill type="tile" src="myimage.gif"`
	`size="25pt,15pt">`
	`</v:fill>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

Attributes	Description
src (Fill Image Source)	Specifies the URL specifying the fill image to use. [*Example:* `<v:fill ... src="myimage.gif" ... ></v:fill>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
title (Title) Namespace: urn:schemas-microsoft-com:office:office	Specifies the title of an embedded fill image. This is typically set to the comment property of the image, which is often blank. [*Example:* `<v:fill ... o:title="alt text" ... ></v:fill>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
type (Fill Type)	Specifies the type of fill. Default is `solid`. Allowed values are: • `solid` • `gradient` • `gradientradial` • `tile` • `pattern` • `frame` [*Example:* Applied to a simple square using the following fill element, the three gradient types look like this: `<v:fill color="red" color2="blue" type="solid"></v:fill>`  `type="solid"`  `type="gradient"`

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>

type="gradientradial"

Applied to a simple square using the following fill elements, the three image types look like this:

<table>
<tr><td>  </td><td><v:fill src="myimage.gif"
type="tile" size="50%,50%">
</v:fill></td></tr>
<tr><td>  </td><td><v:fill src="myimage.gif"
type="frame" size="50%,50%">
</v:fill></td></tr>
<tr><td>  </td><td><v:fill src="myimage.gif"
color="red" color2="blue"
type="pattern">
</v:fill></td></tr>
</table>

end example]

The possible values for this attribute are defined by the ST_FillType simple type (§6.1.3.5).
</td></tr>
</table>

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Fill">
   <sequence>
      <element ref="o:fill" minOccurs="0"/>
   </sequence>
   <attributeGroup ref="AG_Id"/>
   <attribute name="type" type="ST_FillType" use="optional"/>
   <attribute name="on" type="ST_TrueFalse" use="optional"/>
   <attribute name="color" type="ST_ColorType" use="optional"/>
   <attribute name="opacity" type="xsd:string" use="optional"/>
   <attribute name="color2" type="ST_ColorType" use="optional"/>
   <attribute name="src" type="xsd:string" use="optional"/>
   <attribute ref="o:href"/>
   <attribute ref="o:althref"/>
   <attribute name="size" type="xsd:string" use="optional"/>
   <attribute name="origin" type="xsd:string" use="optional"/>
   <attribute name="position" type="xsd:string" use="optional"/>
   <attribute name="aspect" type="ST_ImageAspect" use="optional"/>
   <attribute name="colors" type="xsd:string" use="optional"/>
   <attribute name="angle" type="xsd:decimal" use="optional"/>
   <attribute name="alignshape" type="ST_TrueFalse" use="optional"/>
   <attribute name="focus" type="xsd:string" use="optional"/>
   <attribute name="focussize" type="xsd:string" use="optional"/>
   <attribute name="focusposition" type="xsd:string" use="optional"/>
   <attribute name="method" type="ST_FillMethod" use="optional"/>
   <attribute ref="o:detectmouseclick"/>
   <attribute ref="o:title"/>
   <attribute ref="o:opacity2"/>
   <attribute name="recolor" type="ST_TrueFalse" use="optional"/>
   <attribute name="rotate" type="ST_TrueFalse" use="optional"/>
   <attribute ref="r:id" use="optional"/>
   <attribute ref="o:relid" use="optional"/>
</complexType>
```

6.1.2.6 formulas (Set of Formulas)

This element defines a set of formulas whose calculated values are referenced by other attributes. Each formula is contained in a child f element (§6.1.2.4).

Parent Elements
arc (§6.1.2.1); background (§2.2.1); curve (§6.1.2.3); group (§6.1.2.7); image (§6.1.2.10); line (§6.1.2.12); object (§2.3.3.19); oval (§6.1.2.13); pict (§2.3.3.21); pict (§2.9.23); polyline (§6.1.2.15); rect (§6.1.2.16); roundrect (§6.1.2.17); shape (§6.1.2.19); shapetype (§6.1.2.20)

Child Elements	Subclause
f (Single Formula)	§6.1.2.4

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Formulas">
   <sequence>
      <element name="f" type="CT_F" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

6.1.2.7 group (Shape Group)

This element is used to collect shapes and groups so they can be positioned and transformed as a single unit. A group contains group, shapetype, shape, pre-defined shape - arc, curve, image, line, oval, polyline, rect, roundrect - and lock elements.

[*Example:* The following example defines a few basic parts of a flying saucer graphic. The group consists of five shapes. Each shape's position is determined within the coordinate space of the group, which is defined by the group's attributes.

```
<v:group id="saucer"
  style='position:relative;left:200;top:200;width:50;height:50'
  coordorigin="0,0" coordsize="6000,6000">
  <v:shape id="body"
    style='position:relative;left:234.75pt;top:208.875pt;
    width:235.25pt;height:128.875pt' coordsize="3765,2060"
    path="m1285,251l1126,469,580,1009,,1285,25,1412,93,1547,194,1673,
    1017,2026,2312,2060,3209,1756,3765,1388,3278,680,3059,319,2976,,
    1285,251,1285,251xe"
    fillcolor="#bcbcd6" stroked="f">
    <v:path arrowok="t"/>
  </v:shape>
  <v:shape id="canopy"
    style='position:relative;left:314.625pt;top:140.5pt;
    width:104pt;height:102pt' coordsize="1663,1633"
    path="m0,1355l177,1498,353,1582,840,1633,1378,1498,1663,1295,
    1545,456,1260,10,1025,,656,260,253,874,,1355,,1355xe"
    fillcolor="#99ebff" stroked="f">
    <v:path arrowok="t"/>
  </v:shape>
  <v:shape id="light1"
    style='position:relative;left:408.625pt;top:268.75pt;
    width:24.25pt;height:27.375pt' coordsize="388,437"
    path="m209,0l34,101,,302,125,437,329,327,388,152,209,,209,0xe"
    fillcolor="#fff27f" stroked="f">
    <v:path arrowok="t"/>
  </v:shape>
  <v:shape id="light2"
```

```
      style='position:relative;left:356.625pt;top:279.25pt;
      width:28.875pt;height:30pt' coordsize="462,479"
      path="m135,0l0,186,59,422,344,479,462,228,135,,135,0xe"
      fillcolor="#fff27f" stroked="f">
      <v:path arrowok="t"/>
    </v:shape>
    <v:shape id="light3"
      style='position:relative;left:302.625pt;top:274pt;
      width:23pt;height:23.625pt' coordsize="369,378"
      path="m0,591226,,369,186,243,378,32,363,,59,,59xe"
      fillcolor="#fff27f" stroked="f">
      <v:path arrowok="t"/>
    </v:shape>
  </v:group>
```

end example]

Parent Elements
background (§2.2.1); group (§6.1.2.7); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23)

Child Elements	Subclause
anchorlock (Anchor Location Is Locked)	§6.3.2.1
arc (Arc Segment)	§6.1.2.1
borderbottom (Bottom Border)	§6.3.2.2
borderleft (Left Border)	§6.3.2.3
borderright (Right Border)	§6.3.2.4
bordertop (Top Border)	§6.3.2.5
callout (Callout)	§6.2.2.2
ClientData (Attached Object Data)	§6.4.2.12
clippath (Shape Clipping Path)	§6.2.2.3
curve (Bezier Curve)	§6.1.2.3
diagram (VML Diagram)	§6.2.2.8
extrusion (3D Extrusion)	§6.2.2.10
fill (Shape Fill Properties)	§6.1.2.5
formulas (Set of Formulas)	§6.1.2.6

Child Elements	Subclause
group (Shape Group)	§6.1.2.7
handles (Set of Handles)	§6.1.2.9
image (Image File)	§6.1.2.10
imagedata (Image Data)	§6.1.2.11
line (Line)	§6.1.2.12
lock (Shape Protections)	§6.2.2.17
oval (Oval)	§6.1.2.13
path (Shape Path)	§6.1.2.14
polyline (Multiple Path Line)	§6.1.2.15
rect (Rectangle)	§6.1.2.16
roundrect (Rounded Rectangle)	§6.1.2.17
shadow (Shadow Effect)	§6.1.2.18
shape (Shape Definition)	§6.1.2.19
shapetype (Shape Template)	§6.1.2.20
signatureline (Digital Signature Line)	§6.2.2.29
skew (Skew Transform)	§6.2.2.30
stroke (Line Stroke Settings)	§6.1.2.21
textbox (Text Box)	§6.1.2.22
textdata (VML Diagram Text)	§6.5.2.2
textpath (Text Layout Path)	§6.1.2.23
wrap (Text Wrapping)	§6.3.2.6

Attributes	Description
allowincell (Allow in Table Cell) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape can be placed in a table. Default is `false`. [*Example:* `<v:shape ... o:allowincell="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
allowoverlap (Allow Shape Overlap)	Specifies whether a shape can overlap another shape. Default is `true`. If `false`, the shape will shift left or right so as not to overlap another shape, similar to the behavior of the HTML float attribute.

Attributes	Description
Namespace: urn:schemas-microsoft-com:office:office	[*Example:* `<v:shape ... o:allowoverlap="false" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

alt (Alternate Text)	Specifies alternative text describing the graphical object. This text should provide a brief description of the shape for use by accessibility tools. Default is no value.
	
	[*Example:* The alt text describes the basic shape:
	
	`<v:shape ... fillcolor="red" alt="Red rectangle"></v:shape>`
	
	The alt text describes the contents of a shape displaying an image:
	
	`<v:shape ... alt="Picture of a sunset"></v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

borderbottomcolor (Bottom Border Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the bottom border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:borderbottomcolor="red" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
borderleftcolor (Border Left Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the left border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:borderleftcolor="red" ... ></v:shape>` *end example*]

Attributes	Description
	The possible values for this attribute are defined by the XML Schema string datatype.
borderrightcolor (Border Right Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the right border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:borderrightcolor="red" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
bordertopcolor (Border Top Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the top border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:bordertopcolor="red" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
bullet (Graphical Bullet) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the shape is a graphical bullet. Default is `false`. [*Example:* `<v:shape ... o:bullet="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
button (Button Behavior Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape will exhibit button press behavior on click. Default is `false`. [*Example:* `<v:shape ... o:button="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
class (CSS	Specifies a reference to the definition of a CSS style. Default is no value.

Attributes	Description

Reference)	[*Example:* The snippets below are equivalent:
	
	`...`
	`.narrowstyle {width:50;height:100}`
	`...`
	`<v:shape ... class="narrowstyle"`
	`  style="top:1;left:1">`
	`</v:shape>`
	
	`<v:shape ... style="top:1;left:1;`
	`  width:50;height:100">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

coordorigin (Coordinate Space Origin)	Specifies the coordinate of the top left corner of the shape's coordinate space. This determines the position of the (0,0) coordinate space origin within the shape's bounding box. Default is "0,0", which places the (0,0) origin at the top left corner of the bounding box.
	
	This affects shape properties that specify coordinate positions, such as the path attribute. Thus a path can be defined against a generic (0,0) origin and the coordorigin value translates the entire path within the shape's bounding space.
	
	[*Example:* The horizontal and vertical coordinate space ranges from -100 to +100 because the coordinate space (coordsize) is 200 by 200 and the top left coordinate is (-100,-100). The (0,0) origin lies at the center of the shape's bounding box, as evidenced by the position of the shape's path within the coordinate space:
	
	`<v:shape ... coordsize="200,200"`
	`  coordorigin="-100,-100"`
	`  path="m 0,0 l 0,50, 50,50, 50,0 x e">`
	`</v:shape>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

coordsize (Coordinate Space	Specifies the size of the shape's coordinate space in coordinate units. Default is "1000,1000".

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>Size)</td><td>The physical size of a coordinate unit length is determined by both the size of the coordinate space (coordsize) and the size of the shape (style width and height). The coordsize attribute defines the number of horizontal and vertical subdivisions into which the shape's bounding box is divided. The combination of coordsize and style width/height effective scales the shape anisotropically.

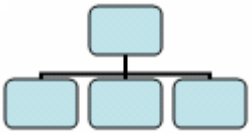
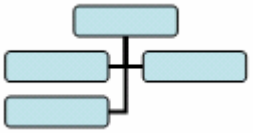
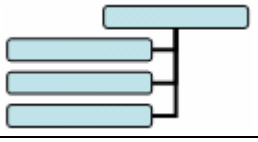
[Example: The path is 50 units wide and tall, which is 25% of the size of the coordinate space:

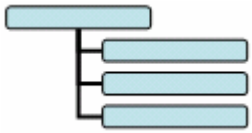
<pre><v:shape ... coordsize="200,200"
  coordorigin="-100,-100"
  path="m 0,0 l 0,50, 50,50, 50,0 x e">
</v:shape></pre>
 

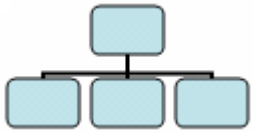
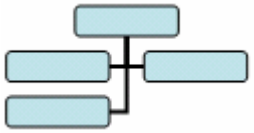
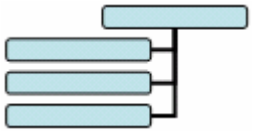
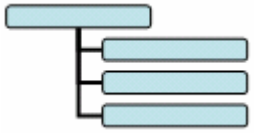
end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>dgmlayout
(Diagram Node Layout Identifier)

Namespace: urn:schemas-microsoft-com:office:office</td><td>Specifies the type of automatic layout to apply to the diagram node. This is only meaningful for organization charts. Valid values for organization charts are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Top-down, centered layout.
  </td></tr>
<tr><td>1</td><td>Hanging, both sides layout.
  </td></tr>
<tr><td>2</td><td>Hanging, right side layout.
  </td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>3</td><td>Hanging, left side layout.
  </td></tr>
</table>
[Example:
<pre><v:shape ... dgmlayout="1">
</v:shape></pre>
end example]
The possible values for this attribute are defined by the XML Schema integer datatype.
</td></tr>
<tr><td>dgmlayoutmru
(Diagram Node Recent Layout Identifier)

Namespace:
urn:schemas-microsoft-com:office:office</td><td>
Specifies the type of automatic layout most recently used on the child elements of the diagram node. This is only meaningful for organization charts. Valid values for organization charts are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Top-down, centered layout.
  </td></tr>
<tr><td>1</td><td>Hanging, both sides layout.
  </td></tr>
<tr><td>2</td><td>Hanging, right side layout.
  </td></tr>
<tr><td>3</td><td>Hanging, left side layout.
  </td></tr>
</table>
[Example:
<pre><v:shape ... dgmlayout="1">
</v:shape></pre>
</td></tr>
</table>

Attributes	Description
	end example] The possible values for this attribute are defined by the XML Schema integer datatype.

dgmnodekind (Diagram Node Identifier)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies an optional, application-specific parameter that is intended to be used by the application to tag different types of nodes in a diagram.
	
	[*Example:*
	
	`<v:shape ... dgmnodekind="1">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema integer datatype.

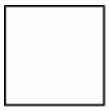
doubleclicknotify (Double-click Notification Toggle)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies that an event message is sent when a shape is double-clicked. Default is `false`.
	
	[*Example:*
	
	`<v:shape ... o:doubleclicknotify="true" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

editas (Group Diagram Type)	Specifies which diagram type the contained shapes represent. This is used in conjunction with the diagram element (§6.2.2.8). A value of `canvas` indicates that the group is a regular group of shapes and does not represent a diagram. Other values indicate that the diagram element and its children contain semantic information relevant to that type of diagram, which is represented by the shapes in the group.
	
	[Example:
	
	`<v:group ... editas="orgchart">`
	`</v:group>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_EditAs simple type (§6.1.3.2).

fillcolor (Fill Color)	Specifies the color to use for the fill. Default is `white`. If the fill element (§6.1.2.5) is present, its color attribute takes precedence. Colors are typically specified as either a named color, such as `red`, or six hexadecimal digits representing the red, green and blue values of the color, such as `#00FF30`. Full details are specified in the simple type description.

Attributes	Description

	[*Example:* This shape is red if its fill is visible:
	
	`<v:shape ... fillcolor="red" ... ></v:shape>`
	
	This is equivalent to:
	
	`<v:shape ... fillcolor="#ff0000" ... ></v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).

filled (Shape Fill Toggle)	Specifies whether the closed path will be filled. Default is `true`. This attribute is overridden by the fill on attribute. [*Example:* `<v:shape ... filled="f" fillcolor="red" ...></v:shape>`  *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
hr (Horizontal Rule Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies that a shape is a horizontal rule. Default is `false`. [*Example:* `<v:shape ... o:hr="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
hralign (Horizontal Rule Alignment)	Specifies the alignment of a horizontal rule. Default is `left`. [*Example:*

Attributes	Description
Namespace: urn:schemas-microsoft-com:office:office	<v:shape ... o:hralign="center" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_HrAlign simple type (§6.2.3.14).
href (Hyperlink Target)	Specifies a hyperlink URL target for the shape. Default is no value. [*Example:* <v:shape ... href="http://www.openxmlformats.org" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
hrnoshade (Horizontal Rule 3D Shading Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies that the horizontal rule does not have 3-D shading. Default is `false`. [*Example:* <v:shape ... o:hrnoshade="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
hrpct (Horizontal Rule Length Percentage) Namespace: urn:schemas-microsoft-com:office:office	Specifies the length of a horizontal rule as a percentage of page width. Default is 0. [*Example:* <v:shape ... o:hrpct="85" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema float datatype.
hrstd (Horizontal Rule Standard Display Toggle) Namespace: urn:schemas-	Specifies whether a shape is displayed as a standard horizontal rule. Only applies if hr is `true`. Default is `false`. [*Example:* <v:shape ... o:hrstd="true" ... >

Attributes	Description
microsoft-com:office:office	</v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
id (Unique Identifier)	Specifies a unique identifier that can be used to reference a VML object. Default is no value. [*Example:* <v:shape ... id="myShape" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
insetmode (Text Inset Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the application calculates the internal text margin instead of using the inset attribute. Default is `custom`. This attribute is only meaningful for text boxes. [*Example:* <v:shape ... o:insetmode="auto" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_InsetMode simple type (§6.2.3.15).
oned (Shape Handle Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the extra handles of a shape are hidden. If `true`, hides all shape handles except the top left and bottom right; that is, the same handles that are used for a straight line segment. Default is `false`. [*Example:* <v:shape ... o:oned="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
print (Print Toggle)	Specifies whether the shape will be printed. Default is `true`.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>[Example:

<pre><v:shape ... print="false" ... >
</v:shape></pre>
end example]

The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).</td></tr>
<tr><td>regroupid (Regroup ID)

Namespace: urn:schemas-microsoft-com:office:office</td><td>Specifies a previous group for a shape. An ID number is used to identify groups of shapes that are no longer grouped. This allows shapes to be regrouped programmatically.

[Example: The shape was part of a group identified by the ID 040754:

<pre><v:shape ... o:regroupid="040754" ... >
</v:shape></pre>
end example]

The possible values for this attribute are defined by the XML Schema integer datatype.</td></tr>
<tr><td>spid (Optional String)

Namespace: urn:schemas-microsoft-com:office:office</td><td>Specifies an optional string that an application may use to Identify the particular shape. Default is no value.

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>style (Shape Styling Properties)</td><td>Specifies the CSS2 styling properties of the shape. This uses the syntax described in the "Visual formatting model" of the Cascading Style Sheets, Level 2 specification, a Recommendation of the World Wide Web Consortium available here: http://www.w3.org/TR/REC-CSS2. Full descriptions of each property are not repeated here, but the VML treatment of each property is defined. Allowed properties include:

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>flip</td><td>Specifies that the orientation of a shape is flipped. Default is no value. Allowed values are:

• x - Flip along the y-axis, reversing the x-coordinates.
• y - Flip along the x-axis, reversing the y-coordinates.
• xy - Flip along both the y- and x-axis.
• yx - Flip along both the x- and y-axis.</td></tr>
<tr><td>height</td><td>Specifies the height of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are:</td></tr>
</table>
</td></tr>
</table>

Attributes	Description
	• auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
left	Specifies the position of the left of the containing block of the shape relative to the element left of it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are: • auto - Default position of an element in the flow of the page. • <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
margin-bottom	Specifies the position of the bottom of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
margin-left	Specifies the position of the left of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the

Attributes	Description
	parent object's width.
margin-right	Specifies the position of the right of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
margin-top	Specifies the position of the top of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
mso-position-horizontal	Specifies the horizontal positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute • left • center • right • inside • outside
mso-position-horizontal-relative	Specifies relative horizontal position data for objects in WordprocessingML documents. This modifies the mso-position-horizontal property. Default is text. Allowed values are: • margin • page • text

Attributes	Description
	• char
mso-position-vertical	Specifies the vertical positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute • top • center • bottom • inside • outside
mso-position-vertical-relative	Specifies relative vertical position data for objects in WordprocessingML documents. This modifies the mso-position-vertical property. Default is text. Allowed values are: • margin • page • text • line
mso-wrap-distance-bottom	Specifies the distance from the bottom of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-distance-left	Specifies the distance from the left side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-distance-right	Specifies the distance from the right side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-distance-top	Specifies the distance from the top of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-	Specifies whether the wrap coordinates were customized by the user. If the wrap coordinates are generated by an editor, this

Attributes	Description	
	edited	property is true; otherwise they were customized by a user. Default is false.
	mso-wrap-style	Specifies the wrapping mode for text in shapes in WordprocessingML documents. Default is square. Allowed values are: • square - Wraps text inside the shape in a square. • none - Text does not wrap.
	position	Specifies the type of positioning used to place an element. Default is static. When the element is contained inside a group, this property must be absolute. Allowed values are: • static - The element is positioned according to the normal flow of the page. The top and left properties are ignored. If the object is anchored inline, this value is used. • absolute - The element is positioned relative to the parent, using the top and left properties. • relative - The element is positioned according to the normal flow of the page, but the top and left properties are used. The overlap of overlapping elements is governed by the z-index property.
	rotation	Specifies the angle that a shape is rotated, in degrees. Default is 0. Positive angles are clockwise.
	top	Specifies the position of the top of the containing block of the shape relative to the element above it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	visibility	Specifies whether a shape is displayed. Only inherit and hidden are used; any other values are mapped to inherit. Default is inherit. Allowed values are: • hidden - The shape is not visible, but is still part of the flow of the objects in the browser. Mouse events are not

Attributes	Description
	processed. • `inherit` - The visibility state is inherited from the parent of the shape.
`width`	Specifies the width of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
`z-index`	Specifies the display order of overlapping shapes. Default is 0. This property shall not be used for shapes anchored inline. Allowed values are: • `auto` - Uses the order that the shapes appear in the page, bottom to top. • <order>- A number that represents the stacking precedence. Shapes with higher numbers are placed on top of those with lower numbers. Negative numbers are allowed.

The following properties are only used by the textbox element (§6.1.2.22):

Property	Description
`direction`	Specifies the direction of the text in the textbox. Default is `ltr`. This property is superceded by the `mso-direction-alt` property if that is specified. Allowed values are: • `ltr` - Test is displayed left-to-right. • `rtl` - Test is displayed right-to-left.
`layout-flow`	Determines the flow of the text layout in a textbox. Default is `horizontal`. Allowed values are: • `horizontal` - Text is displayed horizontally. • `vertical` - Text is displayed vertically. • `vertical-ideographic` - Ideographic text is displayed vertically. • `horizontal-ideographic` - Ideographic text is displayed

Attributes	Description
	horizontally.
mso-direction-alt	Specifies an alternate direction for text in textboxes. Overrides the direction property. The only allowed value is context.
mso-fit-shape-to-text	Specifies whether the shape stretches to fit the text in the textbox. Default is false.
mso-fit-text-to-shape	Specifies whether the text stretches to fit the textbox. Default is false.
mso-layout-flow-alt	Specifies the alternate layout flow for text in textboxes. This property is used instead of layout-flow when the layout flow is from bottom to top for non-ideographic languages. Its only value is bottom-to-top.
mso-next-textbox	Specifies the ID of the next textbox in a series. Used to keep track of a set of linked textboxes. Default is no value.
mso-rotate	Specifies a specific rotation value for text in a textbox. Default is 0. Allowed values are: • 0 • 90 • 180 • -90
mso-text-scale	Specifies the scaling factor for fitting text to shapes. Default is 0. This property is only used if mso-fit-text-to-shape is true.
v-text-anchor	Specifies the vertical anchoring of text in a textbox. Default is top. The alignment of a text anchor only becomes evident if mso-fit-text-to-shape is false. This property is different from the vertical-align CSS property, which is used for ideographic languages. Allowed values are: • top • middle • bottom • top-center • middle-center • bottom-center • top-baseline • bottom-baseline • top-center-baseline • bottom-center-baseline

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
The following properties are only used by the textpath element (§6.1.2.23):
<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>font</td><td>Specifies a compound value of font settings. Default is no value. The values are the same as those of the CSS font property. The order of definitions in the string is: font-style, font-variant, font-weight, font-size, line-height, font-family.</td></tr>
<tr><td>font-family</td><td>Specifies the family of the font. Default is no value. The values are the same as those of the CSS font-family property.</td></tr>
<tr><td>font-size</td><td>Specifies the size of the font. Default is no value. The font size is defined in points. The values are the same as those of the CSS font-size property.</td></tr>
<tr><td>font-style</td><td>Specifies the amount of slant for a font. Default is normal. The values are the same as those of the CSS font-style property. Allowed values are:
normal
italic
oblique - Treated the same as italic.</td></tr>
<tr><td>font-variant</td><td>Specifies the variant style of a font. Default is normal. The values are the same as those of the CSS font-variant property. Allowed values are:
normal
small-caps</td></tr>
<tr><td>font-weight</td><td>Specifies the thickness of the letters of the font. Default is normal. The values are the same as those of the CSS font-weight property. Allowed values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>normal
lighter
100
200
300
400</td><td>Treated as non-bold.</td></tr>
<tr><td>bold</td><td>Treated as bold.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Attributes	Description
	<table><tr><td>bolder 500 600 700 800 900</td><td></td></tr></table>
	mso-text-shadow: Specifies whether a shadow is applied to the text on a text path. Default is false.
	text-decoration: Specifies the style of text decoration. Default is none. The values are the same as those of the CSS text-decoration property. Allowed values are: • none • underline • overline • line-through • blink
	v-rotate-letters: Specifies whether the letters of the text are rotated counterclockwise by 90 degrees. Default is false.
	v-same-letter-heights: Specifies whether all letters will be the same height regardless of initial case. If true, the lowercase letters are stretched to the height of the uppercase letters. Default is false.
	v-text-align: Specifies the alignment of text. Default is left. Allowed values are: • left • right • center • justify • letter-justify - Distributes the extra space between the letters. • stretch-justify - Stretches the letters to fill in the space.
	v-text-kern: Specifies whether kerning is turned on. Default is false.
	v-text-reverse: Specifies whether the layout order of rows is reversed. Default is false. This is used for vertical text layout.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>v-text-spacing-mode</td><td>Specifies the mode for letter spacing. Default is tightening. This property determines whether space will be removed between each letter (tightening) or added between each letter (tracking). The amount of letter spacing change is defined by the v-text-spacing property. Allowed values are:

• tightening
• tracking</td></tr>
<tr><td>v-text-spacing</td><td>Specifies the amount of spacing for text in 100ths of single line spacing. Default is 100.</td></tr>
</table>

The line (§6.1.2.12), polyline (§6.1.2.15) and curve (§6.1.2.3) elements ignore the following properties:

• top
• left
• width
• height

The following properties are not inherited by an element that references a shapetype element (§6.1.2.20) via the id attribute:

• flip
• height
• left
• margin-left
• margin-top
• position
• rotation
• top
• visibility
• width
• z-index

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>tablelimits (Table Row Height Limits)

Namespace: urn:schemas-microsoft-com:office:office</td><td>Specifies a list of minimum height values for each row in a table. Default is no value.

Used by PresentationML for native tables. This attribute is only useful when the table is made up of shapes that are grouped. When text is added to table cells, the row height may increase. The tablelimits attribute stores the original row height so that if text is deleted, the row height will not fall below the original value.

[Example:</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>

```
<v:shape ... o:tablelimits="30pt 20pt" ... >
</v:shape>
```

end example]

The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>tableproperties (Table Properties)

Namespace: urn:schemas-microsoft-com:office:office</td><td>

Specifies a bitmask, represented as an integer, that determines table properties. Only the first three bits of this integer are used. Default is 0.

Used by PresentationML for native tables. This attribute is only useful when the table is made up of shapes that are grouped. Allowed values are:

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>1</td><td>Set if the group of shapes is a table.</td></tr>
<tr><td>2</td><td>Set if the shape is a placeholder.</td></tr>
<tr><td>3</td><td>Set if the table text is bi-directional.</td></tr>
</table>

[*Example:* Decimal 3 means that bits 1 and 2 are set.

```
<v:shape ... o:tableproperties="3" ... >
</v:shape>
```

end example]

The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>target (Hyperlink Display Target)</td><td>

Specifies a frame or window that a URL is displayed in. Default is no value. Allowed values are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td><targetname></td><td>String containing the name of the frame or window in which to load the document.</td></tr>
<tr><td>_blank</td><td>Specifies that the linked document is loaded into a new blank window. This window is not named.</td></tr>
<tr><td>_media</td><td>Specifies that the linked document is loaded into the Media Bar. Available in Microsoft® Internet Explorer 6 or later.</td></tr>
<tr><td>_parent</td><td>Specifies that the linked document is loaded into the immediate parent of the document containing the link.</td></tr>
<tr><td>_search</td><td>Specifies that the linked document is loaded into the browser's search pane. Available in Microsoft Internet Explorer 5 or greater.</td></tr>
<tr><td>_self</td><td>Specifies that the linked document is loaded into the window in which the link was clicked (the active window).</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>_top</td><td>Specifies that the linked document is loaded into the topmost window.</td></tr>
</table>

[Example:

<pre>
<v:shape ...
  href="http://www.openxmlformats.org"
  target="_self" ... >
</v:shape>
</pre>

end example]

The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>title (Shape Title)</td><td>Specifies the text displayed when the mouse pointer moves over the shape. Default is no value.

[Example:

<pre>
<v:shape ... title="tooltip" ... >
</v:shape>
</pre>

end example]

The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>userdrawn (Exists In Master Slide)

Namespace: urn:schemas-microsoft-com:office:office</td><td>Specifies whether the user has added the shape to a master slide. Default is <code>false</code>. Used by PresentationML.

[Example:

<pre>
<v:shape ... o:userdrawn="true" ... >
</v:shape>
</pre>

end example]

The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
</td></tr>
<tr><td>userhidden (Hide Script Anchors)

Namespace: urn:schemas-microsoft-com:office:office</td><td>Specifies whether a script anchor is hidden. Default is <code>false</code>. If <code>true</code>, script anchors stay hidden even if the shape is otherwise visible. A script anchor is the visual representation of a script that when displayed in an application.

[Example:

<pre>
<v:shape ... o:userhidden="true" ... >
</v:shape>
</pre>
</td></tr>
</table>

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

wrapcoords (Shape Bounding Polygon)	Specifies the bounding polygon that surrounds a shape. This is specified using a comma-delimited list of x and y coordinates: "x1,y1,x2,y2,x3,y3,..." This is used when text is tightly wrapped around a shape. Default is no value until the mso-wrap-mode style attribute is set to `tight` or `through`.
	
	[*Example:*
	
	`<v:shape ...`
	`  wrapcoords="0,0 0,200, 200,200, 200,0" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Group">
   <choice maxOccurs="unbounded">
      <group ref="EG_ShapeElements"/>
      <element ref="group"/>
      <element ref="shape"/>
      <element ref="shapetype"/>
      <element ref="arc"/>
      <element ref="curve"/>
      <element ref="image"/>
      <element ref="line"/>
      <element ref="oval"/>
      <element ref="polyline"/>
      <element ref="rect"/>
      <element ref="roundrect"/>
      <element ref="o:diagram"/>
   </choice>
   <attributeGroup ref="AG_AllCoreAttributes"/>
   <attributeGroup ref="AG_Fill"/>
   <attribute name="editas" type="ST_EditAs" use="optional"/>
   <attribute ref="o:tableproperties"/>
   <attribute ref="o:tablelimits"/>
</complexType>
```

6.1.2.8 h (Shape Handle)

This element defines a single handle, which is a user interface element tied to one or two adj values. Moving the handle changes its linked adj values, which in turn changes formulas and attributes that depend on them.

The handle is optionally constrained vertically or horizontally. The linked adj values store the position of the handle in the shape's coordinate space.

[*Example:* The example below defines a simple kite shape with a resizable width:

```
<v:shape coordsize="200,200" coordorigin="-100,-100" adj="100"
style="width:50;height:50;position:relative"
 path="m @1,-50 l 0,-200 @0,-50 0,200 x e">
 <v:formulas>
  <v:f eqn="val #0"/>
  <v:f eqn="sum 0 0 @0"/>
 </v:formulas>
 <v:handles>
  <v:h position="#0,0"/>
 </v:handles>
</v:shape>
```

end example]

Parent Elements
handles (§6.1.2.9)

Attributes	Description

invx (Invert Handle's X Position)	Specifies whether the x position of the handle should be inverted according to: $x_{new} = coordorigin_x + coordsize_x - x_{old}$
	
	Default is `false`.
	
	[*Example:*
	
	`<v:handles>`
	`  <v:h ... invx="true" ... />`
	`</v:handles>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).

invy (Invert Handle's Y Position)	Specifies whether the y position of the handle should be inverted according to: $y_{new} = coordorigin_y + coordsize_y - y_{old}$ Default is `false`. [*Example:*

Attributes	Description
	<v:handles> <v:h ... invy="true" ... /> </v:handles> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
map (Handle Coordinate Mapping)	Specifies how the x and y positions of the handle are mapped from the coordsize range into the specified range. Default is "0,1000". [*Example:* <v:handles> <v:h ... map="-1000,1000" ... /> </v:handles> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
polar (Handle Polar Center)	Specifies the center position of a handle that uses polar coordinates. If specified, the position attribute is assumed to contain radius and angle values. If omitted, the position attribute is assumed to contain x and y positions. Default is no value. [*Example:* <v:handles> <v:h ... polar="0,0" ... /> </v:handles> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
position (Handle Position)	Specifies the x and y position of the handle. If the polar attribute is present, defines the handle position using radius and angle values. Default is "0,0". Each value in the vector is one of the following: • constant • formula (e.g., @2) • adj value (e.g., #2) • center • topleft • bottomright

Attributes	Description

	Each of the above except for an adj value reference fixes the handle position for that dimension. Specifying an adj value allows the handle to move in that dimension and the handle position for that dimension is stored in the adj value.
	
	[*Example:* The handle's x position is fixed but it is free to move in the y dimension:
	
	`<v:handles>`
	`  <v:h ... position="topleft,#2" ... />`
	`</v:handles>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

radiusrange (Handle Polar Radius Range)	Specifies a range of minimum and maximum values that constrain the radius of a handle using polar coordinates. Default is "0,0". Each value is either a constant or a formula reference. Omitting a value leaves that bound unconstrained.
	
	[*Example:* The polar handle may only be moved within a radius range of 25 to 50.
	
	`<v:handles>`
	`  <v:h ... radiusrange="25,50" ... />`
	`</v:handles>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

switch (Handle Inversion Toggle)	Specifies whether the x and y dimensions of the handle are switched when the shape is taller than it is wide. Default is `false`. This is useful for shapes with limo stretch behavior.
	
	[*Example:*
	
	`<v:handles>`
	`  <v:h ... switch="true" ... />`
	`</v:handles>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalseBlank simple type (§6.1.3.15).

xrange (Handle X Position Range)	Specifies a range of minimum and maximum values that constrain the x position of a handle. Default is "0,0". Each value is either a constant or a formula reference. Omitting a value leaves that bound unconstrained.

Attributes	Description

	[*Example:* The handle's x position has a maximum bound of 500 and no minimum bound:
	
	`<v:handles>`
	`  <v:h ... xrange=",500" ... />`
	`</v:handles>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

yrange (Handle Y Position Range)	Specifies a range of minimum and maximum values that constrain the y position of a handle. Default is "0,0". Each value is either a constant or a formula reference. Omitting a value leaves that bound unconstrained.
	
	[*Example:* The handle's y position has a minimum bound of -500 and no maximum bound:
	
	`<v:handles>`
	`  <v:h ... yrange="-500," ... />`
	`</v:handles>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_H">
   <attribute name="position" type="xsd:string"/>
   <attribute name="polar" type="xsd:string"/>
   <attribute name="map" type="xsd:string"/>
   <attribute name="invx" type="ST_TrueFalse"/>
   <attribute name="invy" type="ST_TrueFalse"/>
   <attribute name="switch" type="ST_TrueFalseBlank"/>
   <attribute name="xrange" type="xsd:string"/>
   <attribute name="yrange" type="xsd:string"/>
   <attribute name="radiusrange" type="xsd:string"/>
</complexType>
```

6.1.2.9 handles (Set of Handles)

This element defines a set of user interface elements which can vary a shape's adj values. All dependent formulas and attributes are recalculated. Each handle is defined by a child h element.

Parent Elements
arc (§6.1.2.1); background (§2.2.1); curve (§6.1.2.3); group (§6.1.2.7); image (§6.1.2.10); line (§6.1.2.12); object (§2.3.3.19); oval (§6.1.2.13); pict (§2.3.3.21); pict (§2.9.23); polyline (§6.1.2.15); rect (§6.1.2.16); roundrect (§6.1.2.17); shape (§6.1.2.19); shapetype (§6.1.2.20)

Child Elements	Subclause
h (Shape Handle)	§6.1.2.8

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Handles">
   <sequence>
      <element name="h" type="CT_H" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

6.1.2.10 image (Image File)

This element is used to draw an image that has been loaded from an external source. There is an implied rectangle that is the same size as the image. Any stroke or fill is applied to this implied rectangle. The stroke is drawn on top of the image. The fill is behind the image and therefore only visible through transparent areas of the image. Image transparency is either encoded in the file or defined via a color value using the chromakey attribute. Unlike the imagedata element (§6.1.2.11), the image element does not have a parent element.

[*Example:*

```
<v:image src="myimage.gif"
  style="position:relative;top:1;left:1;width:50;height:45"
  cropbottom="10%" gamma="0.5" gain="2">
</v:image>
```

Original image: 

Modified image: 

end example]

Parent Elements
background (§2.2.1); group (§6.1.2.7); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23)

Child Elements	Subclause
anchorlock (Anchor Location Is Locked)	§6.3.2.1
borderbottom (Bottom Border)	§6.3.2.2
borderleft (Left Border)	§6.3.2.3
borderright (Right Border)	§6.3.2.4
bordertop (Top Border)	§6.3.2.5

Child Elements	Subclause
callout (Callout)	§6.2.2.2
ClientData (Attached Object Data)	§6.4.2.12
clippath (Shape Clipping Path)	§6.2.2.3
extrusion (3D Extrusion)	§6.2.2.10
fill (Shape Fill Properties)	§6.1.2.5
formulas (Set of Formulas)	§6.1.2.6
handles (Set of Handles)	§6.1.2.9
imagedata (Image Data)	§6.1.2.11
lock (Shape Protections)	§6.2.2.17
path (Shape Path)	§6.1.2.14
shadow (Shadow Effect)	§6.1.2.18
signatureline (Digital Signature Line)	§6.2.2.29
skew (Skew Transform)	§6.2.2.30
stroke (Line Stroke Settings)	§6.1.2.21
textbox (Text Box)	§6.1.2.22
textdata (VML Diagram Text)	§6.5.2.2
textpath (Text Layout Path)	§6.1.2.23
wrap (Text Wrapping)	§6.3.2.6

Attributes	Description
allowincell (Allow in Table Cell) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape can be placed in a table. Default is `false`. [*Example:* `<v:shape ... o:allowincell="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
allowoverlap (Allow Shape Overlap) Namespace: urn:schemas-microsoft-	Specifies whether a shape can overlap another shape. Default is `true`. If `false`, the shape will shift left or right so as not to overlap another shape, similar to the behavior of the HTML float attribute. [*Example:* `<v:shape ... o:allowoverlap="false" ... >`

Attributes	Description
com:office:office	</v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
alt (Alternate Text)	Specifies alternative text describing the graphical object. This text should provide a brief description of the shape for use by accessibility tools. Default is no value. [*Example:* The alt text describes the basic shape: <v:shape ... fillcolor="red" alt="Red rectangle"> </v:shape> The alt text describes the contents of a shape displaying an image: <v:shape ... alt="Picture of a sunset"> </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
bilevel (Image Bilevel Toggle)	Specifies that all colors in the picture shall be converted to either 0 or full intensity component values. This converts a color bitmap to 8 colors and a grayscale bitmap to black and white. Default is false. [*Example:* <v:image ... bilevel="true" ...> </v:image>  *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
blacklevel (Image Brightness)	Specifies the image brightness. Default is 0. [*Example:* <v:image ... blacklevel="0.1" ...> </v:image>

Attributes	Description
	 *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
borderbottomcolor (Bottom Border Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the bottom border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:borderbottomcolor="red" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
borderleftcolor (Border Left Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the left border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:borderleftcolor="red" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
borderrightcolor (Border Right Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the right border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:borderrightcolor="red" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
bordertopcolor (Border Top Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the top border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:bordertopcolor="red" ... > </v:shape>`

Attributes	Description
	end example] The possible values for this attribute are defined by the XML Schema string datatype.
bullet (Graphical Bullet) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the shape is a graphical bullet. Default is `false`. [*Example:* `<v:shape ... o:bullet="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
button (Button Behavior Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape will exhibit button press behavior on click. Default is `false`. [*Example:* `<v:shape ... o:button="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
bwmode (Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies how a shape will render for black-and-white output devices. When a shape is printed on a black-and-white printer or displayed in a black-and-white view in an application, several options are possible. Default is `auto`, which will use o:bwnormal for normal black-and-white rendering and o:bwpure for pure black-and-white rendering. bwnormal and bwpure are subordinate to bwmode. If bwmode is "auto" then the value for bwnormal or bwpure is used depending on what the output format is. An application may define for itself what, if any, difference there is between normal B&W and pure B&W. For example, normal B&W might allow greyscale and pure B&W might not. [*Example:* This shape renders in grayscale in a black-and-white environment: `<v:shape ... o:bwmode="grayscale" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
bwnormal (Normal Black-and-White	Specifies the black-and-white mode for normal black-and-white output devices. Default is `auto`.

Attributes	Description
Mode) Namespace: urn:schemas-microsoft-com:office:office	[*Example:* This shape renders in a pale grayscale in a normal black-and-white environment: `<v:shape ... o:bwmode="auto" o:bwnormal="lightgrayscale" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
bwpure (Pure Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies the black-and-white mode for pure black-and-white output devices. Default is auto. [*Example:* This shape renders in high contrast when in a pure black-and-white environment: `<v:shape ... o:bwmode="auto" o:bwpure="highcontrast" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
chromakey (Image Transparency Color)	Specifies a color value that will be transparent and show anything behind the shape. Default is no value. [*Example:* `<v:image ... chromakey="white" ...></v:image>` *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).

class (CSS Reference)	Specifies a reference to the definition of a CSS style. Default is no value.
	
	[*Example:* The snippets below are equivalent:
	
	`...`
	`.narrowstyle {width:50;height:100}`
	`...`
	`<v:shape ... class="narrowstyle"`
	`style="top:1;left:1">`

Attributes	Description
	</v:shape> <v:shape ... style="top:1;left:1; width:50;height:100"> </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
clip (Clipping Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies that the clipping region is active. This is used in conjunction with the clippath (§6.2.2.3) element to create a clipping region. [*Example:* <v:shape ... o:clip="true"> </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
cliptowrap (Clip to Wrapping Polygon) Namespace: urn:schemas-microsoft-com:office:office	Specifies that the clipping region for the shape aligns with the wrapping polygon that tightly surrounds the entire shape (essentially, that the shape shall not be drawn beyond its wrapping polygon's extents – if it does, the shape shall be clipped). Default is `false`. [*Example:* <v:shape ... o:cliptowrap="true"> </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
connectortype (Shape Connector Type) Namespace: urn:schemas-microsoft-com:office:office	Specifies the type of connector used for joining shapes. Default is `straight`. [*Example:* <v:shape ... o:connectortype="elbow" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_ConnectorType simple type

Attributes	Description
	(§6.2.3.6).

coordorigin (Coordinate Space Origin)	Specifies the coordinate of the top left corner of the shape's coordinate space. This determines the position of the (0,0) coordinate space origin within the shape's bounding box. Default is "0,0", which places the (0,0) origin at the top left corner of the bounding box.
	
	This affects shape properties that specify coordinate positions, such as the path attribute. Thus a path can be defined against a generic (0,0) origin and the coordorigin value translates the entire path within the shape's bounding space.
	
	[*Example:* The horizontal and vertical coordinate space ranges from -100 to +100 because the coordinate space (coordsize) is 200 by 200 and the top left coordinate is (-100,-100). The (0,0) origin lies at the center of the shape's bounding box, as evidenced by the position of the shape's path within the coordinate space:
	
	`<v:shape ... coordsize="200,200"`
	`  coordorigin="-100,-100"`
	`  path="m 0,0 l 0,50, 50,50, 50,0 x e">`
	`</v:shape>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

coordsize (Coordinate Space Size)	Specifies the size of the shape's coordinate space in coordinate units. Default is "1000,1000".
	
	The physical size of a coordinate unit length is determined by both the size of the coordinate space (coordsize) and the size of the shape (style width and height). The coordsize attribute defines the number of horizontal and vertical subdivisions into which the shape's bounding box is divided. The combination of coordsize and style width/height effective scales the shape anisotropically.
	
	[*Example:* The path is 50 units wide and tall, which is 25% of the size of the coordinate space:
	
	`<v:shape ... coordsize="200,200"`
	`  coordorigin="-100,-100"`
	`  path="m 0,0 l 0,50, 50,50, 50,0 x e">`
	`</v:shape>`

Attributes	Description
	 *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

cropbottom (Image Bottom Crop)	Specifies the how much to crop the image from the bottom up as a fraction of picture size. Default is 0. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8.
	
	[*Example:*
	
	`<v:image ... cropbottom="10%" ...>`
	`</v:image>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

cropleft (Image Left Crop)	Specifies how much to crop the image from the left in as a fraction of picture size. Default is 0. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8.
	
	[*Example:*
	
	`<v:image ... cropleft="10%" ...>`
	`</v:image>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

cropright (Image Right Crop)	Specifies how much to crop the image from the right in as a fraction of picture size. Default is 0. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8. [*Example:* `<v:image ... cropright="10%" ...>`

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>

```
</v:image>
```

end example]

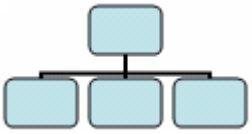
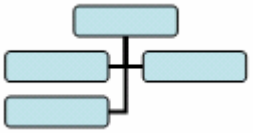
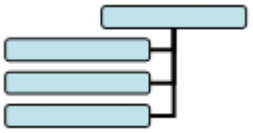
The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>croptop (Image Top Crop)</td><td>Specifies how much to crop the image from the top down as a fraction of picture size. Default is 0. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8.

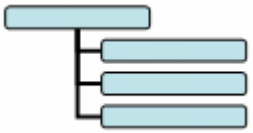
[*Example:*

```
<v:image ... croptop="10%" ...>
</v:image>
```

end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>dgmlayout (Diagram Node Layout Identifier)

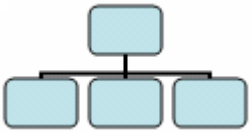
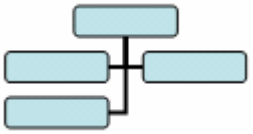
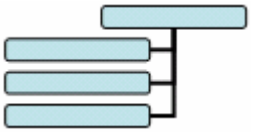
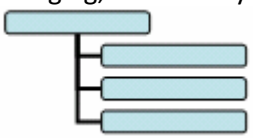
Namespace: urn:schemas-microsoft-com:office:office</td><td>Specifies the type of automatic layout to apply to the diagram node. This is only meaningful for organization charts. Valid values for organization charts are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Top-down, centered layout.
  </td></tr>
<tr><td>1</td><td>Hanging, both sides layout.
  </td></tr>
<tr><td>2</td><td>Hanging, right side layout.
  </td></tr>
<tr><td>3</td><td>Hanging, left side layout.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>  </td></tr>
</table>

[Example:

<pre><v:shape ... dgmlayout="1">
</v:shape></pre>
end example]

The possible values for this attribute are defined by the XML Schema integer datatype.
</td></tr>
<tr><td>dgmlayoutmru (Diagram Node Recent Layout Identifier)

Namespace: urn:schemas-microsoft-com:office:office</td><td>
Specifies the type of automatic layout most recently used on the child elements of the diagram node. This is only meaningful for organization charts. Valid values for organization charts are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Top-down, centered layout.
  </td></tr>
<tr><td>1</td><td>Hanging, both sides layout.
  </td></tr>
<tr><td>2</td><td>Hanging, right side layout.
  </td></tr>
<tr><td>3</td><td>Hanging, left side layout.
  </td></tr>
</table>
[Example:

<pre><v:shape ... dgmlayout="1">
</v:shape></pre>
end example]
</td></tr>
</table>

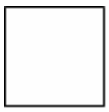
Attributes	Description
	The possible values for this attribute are defined by the XML Schema integer datatype.

dgmnodekind (Diagram Node Identifier)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies an optional, application-specific parameter that is intended to be used by the application to tag different types of nodes in a diagram.
	
	[*Example:*
	
	`<v:shape ... dgmnodekind="1">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema integer datatype.

doubleclicknotify (Double-click Notification Toggle)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies that an event message is sent when a shape is double-clicked. Default is `false`.
	
	[*Example:*
	
	`<v:shape ... o:doubleclicknotify="true" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

fillcolor (Fill Color)	Specifies the color to use for the fill. Default is `white`. If the fill element (§6.1.2.5) is present, its color attribute takes precedence. Colors are typically specified as either a named color, such as `red`, or six hexadecimal digits representing the red, green and blue values of the color, such as `#00FF30`. Full details are specified in the simple type description.
	
	[*Example:* This shape is red if its fill is visible:
	
	`<v:shape ... fillcolor="red" ... >`
	`</v:shape>`
	
	This is equivalent to:
	
	`<v:shape ... fillcolor="#ff0000" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).

filled (Shape Fill	Specifies whether the closed path will be filled. Default is `true`. This attribute is

Attributes	Description
Toggle)	overridden by the fill on attribute. [*Example:* `<v:shape ... filled="f" fillcolor="red" ...> </v:shape>`  *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
forcedash (Force Dashed Outline) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a dashed outline is used to draw a shape when a shape has no line or fill. Default is `false`. Used by PresentationML placeholders to draw a dashed outline when there is no line and no fill for a shape. [*Example:* `<v:shape ... o:forcedash="true" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
gain (Image Intensity)	Specifies an adjustment for the intensity of all colors. Essentially sets how bright white will be. Default is 1. [*Example:* `<v:image ... gain="0.5" ...> </v:image>`  *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

Attributes	Description

gamma (Image Gamma Correction)	Specifies the gamma correction. Default is 1.
	
	Gamma correction is a factor by which the intended target display gamma differs from the sRGB profile. It can be used to correct for images not prepared for sRGB displays and to adjust overall image contrast. Decreasing it below 1 gives a higher contrast image.
	
	[*Example:*
	
	`<v:image ... gamma="0.5" ...>`
	`</v:image>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

grayscale (Image Grayscale Toggle)	Specifies to display the image in grayscale. Default is `false`.
	
	[*Example:*
	
	`<v:image ... gamma="0.5" ...>`
	`</v:image>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).

hr (Horizontal Rule Toggle)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies that a shape is a horizontal rule. Default is `false`.
	
	[*Example:*
	
	`<v:shape ... o:hr="true" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

hralign (Horizontal Rule Alignment)	Specifies the alignment of a horizontal rule. Default is `left`.

Attributes	Description
Namespace: urn:schemas-microsoft-com:office:office	[*Example:* `<v:shape ... o:hralign="center" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_HrAlign simple type (§6.2.3.14).
href (Hyperlink Target)	Specifies a hyperlink URL target for the shape. Default is no value. [*Example:* `<v:shape ... href="http://www.openxmlformats.org" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
hrnoshade (Horizontal Rule 3D Shading Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies that the horizontal rule does not have 3-D shading. Default is `false`. [*Example:* `<v:shape ... o:hrnoshade="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
hrpct (Horizontal Rule Length Percentage) Namespace: urn:schemas-microsoft-com:office:office	Specifies the length of a horizontal rule as a percentage of page width. Default is 0. [*Example:* `<v:shape ... o:hrpct="85" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema float datatype.
hrstd (Horizontal Rule Standard Display Toggle) Namespace:	Specifies whether a shape is displayed as a standard horizontal rule. Only applies if hr is `true`. Default is `false`. [*Example:*

Attributes	Description
urn:schemas-microsoft-com:office:office	<v:shape ... o:hrstd="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
id (Unique Identifier)	Specifies a unique identifier that can be used to reference a VML object. Default is no value. [*Example:* <v:shape ... id="myShape" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
insetmode (Text Inset Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the application calculates the internal text margin instead of using the inset attribute. Default is `custom`. This attribute is only meaningful for text boxes. [*Example:* <v:shape ... o:insetmode="auto" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_InsetMode simple type (§6.2.3.15).
insetpen (Inset Border From Path)	Specifies that the border shall be displayed inside of the path defining the shape, rather than along the path (the default border placement), or outside of the path as might be done with an image. [*Example:* <v:shape ... insetpen="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
ole (Embedded	Specifies whether the shape is an embedded object. Default is `false`.

Attributes	Description
Object Toggle) Namespace: urn:schemas-microsoft-com:office:office	[*Example:* `<v:shape ... o:ole="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalseBlank simple type (§6.2.3.24).
oleicon (Embedded Object Icon Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether an embedded object will be displayed as an icon. Default is `false`. [*Example:* `<v:shape ... o:oleicon="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
oned (Shape Handle Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the extra handles of a shape are hidden. If `true`, hides all shape handles except the top left and bottom right; that is, the same handles that are used for a straight line segment. Default is `false`. [*Example:* `<v:shape ... o:oned="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
opacity (Fill Color Opacity)	Specifies the opacity of the primary fill color. Default is 1.0. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8. [*Example:* The red color is 25% opaque: `<v:fill type="gradient" color="red" color2="blue" opacity=".25"></v:fill>`

Attributes	Description
	 opacity="1"  opacity=".25" *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
preferrelative (Relative Resize Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the original size of an object is saved after reformatting. Default is `false`. If `true`, the original size of the object is stored and all resizing is based on a percentage of that original size. Otherwise, each resizing will reset the scale to 100%. [*Example:* `<v:shape ... o:preferrelative="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
print (Print Toggle)	Specifies whether the shape will be printed. Default is `true`. [*Example:* `<v:shape ... print="false" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
regroupid (Regroup ID) Namespace: urn:schemas-microsoft-com:office:office	Specifies a previous group for a shape. An ID number is used to identify groups of shapes that are no longer grouped. This allows shapes to be regrouped programmatically. [*Example:* The shape was part of a group identified by the ID 040754: `<v:shape ... o:regroupid="040754" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema integer datatype.

Attributes	Description
spid (Optional String) Namespace: urn:schemas-microsoft-com:office:office	Specifies an optional string that an application may use to Identify the particular shape. Default is no value. The possible values for this attribute are defined by the XML Schema string datatype.
spt (Optional Number) Namespace: urn:schemas-microsoft-com:office:office	Specifies an optional number that an application may use to associate the particular shape with a defined shape type. Default is 0. The possible values for this attribute are defined by the XML Schema float datatype.

src (Image Source)	Specifies the URL of the image to use.
	
	[*Example:*
	
	`<v:image ... src="myimage.gif" ...>`
	`</v:image>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

strokecolor (Shape Stroke Color)	Specifies the primary color of the brush to use to stroke the path of the shape. Default is `black`. The color attribute of the stroke element (§6.1.2.21) overrides this. Colors are typically specified as either a named color, such as `red`, or six hexadecimal digits representing the red, green and blue values of the color, such as `#00FF30`. Full details are specified in the simple type description.
	
	[*Example:*
	
	`<v:shape ... strokecolor="red" ...>`
	`</v:shape>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_ColorType simple type

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>(§6.1.3.1).</td></tr>
<tr><td>stroked (Shape Stroke Toggle)</td><td>Specifies whether the path defining the shape is stroked with a solid line. The stroke element (§6.1.2.21) defines other strokes. The on attribute of the stroke element overrides this attribute. Default is true.

[Example:

<pre><v:shape ... fillcolor="red"
  stroked="false" strokecolor="blue"...>
</v:shape></pre>
 

end example]

The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).</td></tr>
<tr><td>strokeweight (Shape Stroke Weight)</td><td>Specifies the width of the brush to use to stroke the path. Default is 1 point. If a number is given without units, the emu is used. The weight attribute of the stroke element (§6.1.2.21) overrides this attribute.

[Example:

<pre><v:shape ... strokeweight="3pt" ... >
</v:shape></pre>
 

end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>style (Shape Styling Properties)</td><td>Specifies the CSS2 styling properties of the shape. This uses the syntax described in the "Visual formatting model" of the Cascading Style Sheets, Level 2 specification, a Recommendation of the World Wide Web Consortium available here: http://www.w3.org/TR/REC-CSS2. Full descriptions of each property are not repeated here, but the VML treatment of each property is defined. Allowed properties include:

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>flip</td><td>Specifies that the orientation of a shape is flipped. Default is no value. Allowed values are:</td></tr>
</table>
</td></tr>
</table>

Attributes	Description
	• x - Flip along the y-axis, reversing the x-coordinates. • y - Flip along the x-axis, reversing the y-coordinates. • xy - Flip along both the y- and x-axis. • yx - Flip along both the x- and y-axis.
`height`	Specifies the height of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
`left`	Specifies the position of the left of the containing block of the shape relative to the element left of it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
`margin-bottom`	Specifies the position of the bottom of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
`margin-left`	Specifies the position of the left of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are:

Attributes	Description	
		• auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	margin-right	Specifies the position of the right of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	margin-top	Specifies the position of the top of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	mso-position-horizontal	Specifies the horizontal positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute • left • center • right • inside • outside
	mso-position-	Specifies relative horizontal position data for objects in

Attributes	Description	
	horizontal-relative	WordprocessingML documents. This modifies the mso-position-horizontal property. Default is text. Allowed values are: • margin • page • text • char
	mso-position-vertical	Specifies the vertical positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute • top • center • bottom • inside • outside
	mso-position-vertical-relative	Specifies relative vertical position data for objects in WordprocessingML documents. This modifies the mso-position-vertical property. Default is text. Allowed values are: • margin • page • text • line
	mso-wrap-distance-bottom	Specifies the distance from the bottom of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
	mso-wrap-distance-left	Specifies the distance from the left side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
	mso-wrap-distance-right	Specifies the distance from the right side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.

Attributes	Description	
	mso-wrap-distance-top	Specifies the distance from the top of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
	mso-wrap-edited	Specifies whether the wrap coordinates were customized by the user. If the wrap coordinates are generated by an editor, this property is true; otherwise they were customized by a user. Default is false.
	mso-wrap-style	Specifies the wrapping mode for text in shapes in WordprocessingML documents. Default is square. Allowed values are: • square - Wraps text inside the shape in a square. • none - Text does not wrap.
	position	Specifies the type of positioning used to place an element. Default is static. When the element is contained inside a group, this property must be absolute. Allowed values are: • static - The element is positioned according to the normal flow of the page. The top and left properties are ignored. If the object is anchored inline, this value is used. • absolute - The element is positioned relative to the parent, using the top and left properties. • relative - The element is positioned according to the normal flow of the page, but the top and left properties are used. The overlap of overlapping elements is governed by the z-index property.
	rotation	Specifies the angle that a shape is rotated, in degrees. Default is 0. Positive angles are clockwise.

Attributes	Description	
	top	Specifies the position of the top of the containing block of the shape relative to the element above it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	visibility	Specifies whether a shape is displayed. Only inherit and hidden are used; any other values are mapped to inherit. Default is inherit. Allowed values are: • hidden - The shape is not visible, but is still part of the flow of the objects in the browser. Mouse events are not processed. • inherit - The visibility state is inherited from the parent of the shape.
	width	Specifies the width of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	z-index	Specifies the display order of overlapping shapes. Default is 0. This property shall not be used for shapes anchored inline. Allowed values are: • auto - Uses the order that the shapes appear in the page, bottom to top. • <order>- A number that represents the stacking precedence. Shapes with higher numbers are placed on top of those with lower numbers. Negative numbers are allowed.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
The following properties are only used by the textbox element (§6.1.2.22):
<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>direction</td><td>Specifies the direction of the text in the textbox. Default is ltr. This property is superceded by the mso-direction-alt property if that is specified. Allowed values are:
ltr - Test is displayed left-to-right.
rtl - Test is displayed right-to-left.</td></tr>
<tr><td>layout-flow</td><td>Determines the flow of the text layout in a textbox. Default is horizontal. Allowed values are:
horizontal - Text is displayed horizontally.
vertical - Text is displayed vertically.
vertical-ideographic - Ideographic text is displayed vertically.
horizontal-ideographic - Ideographic text is displayed horizontally.</td></tr>
<tr><td>mso-direction-alt</td><td>Specifies an alternate direction for text in textboxes. Overrides the direction property. The only allowed value is context.</td></tr>
<tr><td>mso-fit-shape-to-text</td><td>Specifies whether the shape stretches to fit the text in the textbox. Default is false.</td></tr>
<tr><td>mso-fit-text-to-shape</td><td>Specifies whether the text stretches to fit the textbox. Default is false.</td></tr>
<tr><td>mso-layout-flow-alt</td><td>Specifies the alternate layout flow for text in textboxes. This property is used instead of layout-flow when the layout flow is from bottom to top for non-ideographic languages. Its only value is bottom-to-top.</td></tr>
<tr><td>mso-next-textbox</td><td>Specifies the ID of the next textbox in a series. Used to keep track of a set of linked textboxes. Default is no value.</td></tr>
<tr><td>mso-rotate</td><td>Specifies a specific rotation value for text in a textbox. Default is 0. Allowed values are:
0
90
180</td></tr>
</table>
</td></tr>
</table>

Attributes	Description
	<table><tr><td></td><td>• -90</td></tr><tr><td>mso-text-scale</td><td>Specifies the scaling factor for fitting text to shapes. Default is 0. This property is only used if mso-fit-text-to-shape is true.</td></tr><tr><td>v-text-anchor</td><td>Specifies the vertical anchoring of text in a textbox. Default is top. The alignment of a text anchor only becomes evident if mso-fit-text-to-shape is false. This property is different from the vertical-align CSS property, which is used for ideographic languages. Allowed values are: • top • middle • bottom • top-center • middle-center • bottom-center • top-baseline • bottom-baseline • top-center-baseline • bottom-center-baseline</td></tr></table> The following properties are only used by the textpath element (§6.1.2.23): <table><tr><th>Property</th><th>Description</th></tr><tr><td>font</td><td>Specifies a compound value of font settings. Default is no value. The values are the same as those of the CSS font property. The order of definitions in the string is: font-style, font-variant, font-weight, font-size, line-height, font-family.</td></tr><tr><td>font-family</td><td>Specifies the family of the font. Default is no value. The values are the same as those of the CSS font-family property.</td></tr><tr><td>font-size</td><td>Specifies the size of the font. Default is no value. The font size is defined in points. The values are the same as those of the CSS font-size property.</td></tr><tr><td>font-style</td><td>Specifies the amount of slant for a font. Default is normal. The values are the same as those of the CSS font-style property. Allowed values are: • normal • italic • oblique - Treated the same as italic.</td></tr><tr><td>font-variant</td><td>Specifies the variant style of a font. Default is normal. The values are the same as those of the CSS font-variant property. Allowed values are:</td></tr></table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>

normal
small-caps

</td></tr>
<tr><td>font-weight</td><td>Specifies the thickness of the letters of the font. Default is normal. The values are the same as those of the CSS font-weight property. Allowed values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>normal
lighter
100
200
300
400</td><td>Treated as non-bold.</td></tr>
<tr><td>bold
bolder
500
600
700
800
900</td><td>Treated as bold.</td></tr>
</table>
</td></tr>
<tr><td>mso-text-shadow</td><td>Specifies whether a shadow is applied to the text on a text path. Default is false.</td></tr>
<tr><td>text-decoration</td><td>Specifies the style of text decoration. Default is none. The values are the same as those of the CSS text-decoration property. Allowed values are:

none
underline
overline
line-through
blink

</td></tr>
<tr><td>v-rotate-</td><td>Specifies whether the letters of the text are rotated counterclockwise by 90 degrees. Default is false.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>letters</td><td></td></tr>
<tr><td>v-same-letter-heights</td><td>Specifies whether all letters will be the same height regardless of initial case. If true, the lowercase letters are stretched to the height of the uppercase letters. Default is false.</td></tr>
<tr><td>v-text-align</td><td>Specifies the alignment of text. Default is left. Allowed values are:
left
right
center
justify
letter-justify - Distributes the extra space between the letters.
stretch-justify - Stretches the letters to fill in the space.</td></tr>
<tr><td>v-text-kern</td><td>Specifies whether kerning is turned on. Default is false.</td></tr>
<tr><td>v-text-reverse</td><td>Specifies whether the layout order of rows is reversed. Default is false. This is used for vertical text layout.</td></tr>
<tr><td>v-text-spacing-mode</td><td>Specifies the mode for letter spacing. Default is tightening. This property determines whether space will be removed between each letter (tightening) or added between each letter (tracking). The amount of letter spacing change is defined by the v-text-spacing property. Allowed values are:
tightening
tracking</td></tr>
<tr><td>v-text-spacing</td><td>Specifies the amount of spacing for text in 100ths of single line spacing. Default is 100.</td></tr>
</table>
The line (§6.1.2.12), polyline (§6.1.2.15) and curve (§6.1.2.3) elements ignore the following properties:
top
left
width
height
The following properties are not inherited by an element that references a shapetype element (§6.1.2.20) via the id attribute:
flip
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>

height
left
margin-left
margin-top
position
rotation
top
visibility
width
z-index

The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>target (Hyperlink Display Target)</td><td>
Specifies a frame or window that a URL is displayed in. Default is no value. Allowed values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td><targetname></td><td>String containing the name of the frame or window in which to load the document.</td></tr>
<tr><td>_blank</td><td>Specifies that the linked document is loaded into a new blank window. This window is not named.</td></tr>
<tr><td>_media</td><td>Specifies that the linked document is loaded into the Media Bar. Available in Microsoft® Internet Explorer 6 or later.</td></tr>
<tr><td>_parent</td><td>Specifies that the linked document is loaded into the immediate parent of the document containing the link.</td></tr>
<tr><td>_search</td><td>Specifies that the linked document is loaded into the browser's search pane. Available in Microsoft Internet Explorer 5 or greater.</td></tr>
<tr><td>_self</td><td>Specifies that the linked document is loaded into the window in which the link was clicked (the active window).</td></tr>
<tr><td>_top</td><td>Specifies that the linked document is loaded into the topmost window.</td></tr>
</table>
[Example:
<pre>
  <v:shape ...
    href="http://www.openxmlformats.org"
    target="_self" ... >
  </v:shape>
</pre>
end example]
The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>title (Shape Title)</td><td>Specifies the text displayed when the mouse pointer moves over the shape. Default is no</td></tr>
</table>

Attributes	Description
	value. [*Example:* `<v:shape ... title="tooltip" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
userdrawn (Exists In Master Slide) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the user has added the shape to a master slide. Default is `false`. Used by PresentationML. [*Example:* `<v:shape ... o:userdrawn="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
userhidden (Hide Script Anchors) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a script anchor is hidden. Default is `false`. If `true`, script anchors stay hidden even if the shape is otherwise visible. A script anchor is the visual representation of a script that when displayed in an application. [*Example:* `<v:shape ... o:userhidden="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
wrapcoords (Shape Bounding Polygon)	Specifies the bounding polygon that surrounds a shape. This is specified using a comma-delimited list of x and y coordinates: "x1,y1,x2,y2,x3,y3,..." This is used when text is tightly wrapped around a shape. Default is no value until the mso-wrap-mode style attribute is set to `tight` or `through`. [*Example:* `<v:shape ... wrapcoords="0,0 0,200, 200,200, 200,0" ... ></v:shape>`

Attributes	Description
	end example] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Image">
   <sequence>
      <group ref="EG_ShapeElements" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attributeGroup ref="AG_AllCoreAttributes"/>
   <attributeGroup ref="AG_AllShapeAttributes"/>
   <attributeGroup ref="AG_ImageAttributes"/>
</complexType>
```

6.1.2.11 imagedata (Image Data)

This element is used to draw an image that has been loaded from an external source. There is an implied rectangle that is the same size as the image. Any stroke or fill is applied to this implied rectangle. The stroke is drawn on top of the image. The fill is behind the image and therefore only visible through transparent areas of the image. Image transparency is either encoded in the file or defined via a color value using the chromakey attribute. Unlike the image element (§6.1.2.10), the imagedata element must have a parent element.

[*Example:*

```
<v:shape style="position:relative;top:1;left:1;width:50;height:50"
  path="m 0,0 l 1000,0 1000,1000 0,1000 x e" fillcolor="blue">
  <v:imagedata src="myimage.gif"/>
</v:shape>
```

end example]

Parent Elements
arc (§6.1.2.1); background (§2.2.1); curve (§6.1.2.3); group (§6.1.2.7); image (§6.1.2.10); line (§6.1.2.12); object (§2.3.3.19); oval (§6.1.2.13); pict (§2.3.3.21); pict (§2.9.23); polyline (§6.1.2.15); rect (§6.1.2.16); roundrect (§6.1.2.17); shape (§6.1.2.19); shapetype (§6.1.2.20)

Attributes	Description
althref (Alternate Image Reference)	Defines an alternate reference for an image in Macintosh PICT format. [*Example:*

Attributes	Description
Namespace: urn:schemas-microsoft-com:office:office	<v:imagedata ... althref="myimage.pcz" ... > </v:imagedata> *end example* The possible values for this attribute are defined by the XML Schema string datatype.
bilevel (Image Bilevel Toggle)	Specifies that all colors in the picture shall be converted to either 0 or full intensity component values. This converts a color bitmap to 8 colors and a grayscale bitmap to black and white. Default is `false`. [*Example:* <v:image ... bilevel="true" ...> </v:image>  *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
blacklevel (Image Brightness)	Specifies the image brightness. Default is 0. [*Example:* <v:image ... blacklevel="0.1" ...> </v:image>  *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
chromakey (Image Transparency Color)	Specifies a color value that will be transparent and show anything behind the shape. Default is no value. [*Example:* <v:image ... chromakey="white" ...> </v:image>

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).

cropbottom (Image Bottom Crop)	Specifies the how much to crop the image from the bottom up as a fraction of picture size. Default is 0. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8.
	
	[*Example:*
	
	`<v:image ... cropbottom="10%" ...>`
	`</v:image>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

cropleft (Image Left Crop)	Specifies how much to crop the image from the left in as a fraction of picture size. Default is 0. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8.
	
	[*Example:*
	
	`<v:image ... cropleft="10%" ...>`
	`</v:image>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

cropright (Image Right Crop)	Specifies how much to crop the image from the right in as a fraction of picture size. Default is 0. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8.
	
	[*Example:*
	
	`<v:image ... cropright="10%" ...>`
	`</v:image>`

Attributes	Description
	 *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

croptop (Image Top Crop)	Specifies how much to crop the image from the top down as a fraction of picture size. Default is 0. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8.
	
	[*Example:*
	
	`<v:image ... croptop="10%" ...>`
	`</v:image>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

detectmouseclick (Detect Mouse Click) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a mouse click is detected on the fill of a shape. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
embosscolor (Embossed Color)	Specifies the color to use to create an embossed effect in the image. Default is no value. This can be set to a percentage of the shadow color to create an embossed picture effect. The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).

gain (Image Intensity)	Specifies an adjustment for the intensity of all colors. Essentially sets how bright white will be. Default is 1.
	
	[*Example:*
	
	`<v:image ... gain="0.5" ...>`
	`</v:image>`

Attributes	Description
	 *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

gamma (Image Gamma Correction)	Specifies the gamma correction. Default is 1.
	
	Gamma correction is a factor by which the intended target display gamma differs from the sRGB profile. It can be used to correct for images not prepared for sRGB displays and to adjust overall image contrast. Decreasing it below 1 gives a higher contrast image.
	
	[*Example:*
	
	`<v:image ... gamma="0.5" ...>`
	`</v:image>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

grayscale (Image Grayscale Toggle)	Specifies to display the image in grayscale. Default is `false`.
	
	[*Example:*
	
	`<v:image ... gamma="0.5" ...>`
	`</v:image>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).

href (Explicit Relationship to Hyperlink Target) Namespace: …/officeDocument	Specifies the relationship ID of the relationship to the hyperlink used for this VML object. The specified relationship shall be of type http://schemas.openxmlformats.org/officeDocument/2006/relationships/image or the document shall be considered non-conformant. [*Example*: The markup specifies the associated relationship part with relationship ID

Attributes	Description
/2006/relationships	`rId10` contains the corresponding relationship information for the image data: `< ... r:href="rId5" />` *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

href (Original Image Reference)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies the URL to the original image file. Used only if the picture has been linked and embedded. Default is no value.
	
	[*Example:*
	
	`<v:fill ... o:href="myimage.gif" ... >`
	`</v:fill>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

id (Unique Identifier)	Specifies a unique identifier that can be used to reference a VML object.
	
	Default is no value.
	
	[*Example:*
	
	`<v:shape ... id="myShape" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

id (Explicit Relationship to Image Data) Namespace: .../officeDocument/2006/relationships	Specifies the relationship ID of the relationship to the image used for this VML object. The specified relationship shall be of type http://schemas.openxmlformats.org/officeDocument/2006/relationships/image or the document shall be considered non-conformant. [*Example*: The markup specifies the associated relationship part with relationship ID `rId10` contains the corresponding relationship information for the image data: `< ... r:id="rId10" />` *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

Attributes	Description

movie (Movie Reference)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies a pointer to a movie image. This is a data block that contains a pointer to a pointer to movie data.
	
	[*Example:*
	
	`<v:imagedata ... o:movie="1434" ...>`
	`</v:imagedata>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema float datatype.

oleid (Image Embedded Object ID) Namespace: urn:schemas-microsoft-com:office:office	Specifies the embedded object ID of an image. The possible values for this attribute are defined by the XML Schema float datatype.
pict (Explicit Relationship to Alternate Image Data) Namespace: .../officeDocument/2006/relationships	Specifies the relationship ID of the relationship to an alternate format image used for this VML object. The specified relationship shall be of type http://schemas.openxmlformats.org/officeDocument/2006/relationships/image or the document shall be considered non-conformant. If this attribute is specified, the application should attempt to display the image defined by the relationship. If the application cannot display the format of that image, the r:id attribute is used. [*Example*: The markup specifies the associated relationship part with relationship ID `rId7` contains the corresponding relationship information for the image data. The relationship part with relationship ID `rId10` is used if the application cannot display the image referenced by `rId7`.: `< ... r:id="rId10" r:pict="rId7"/>` *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

recolortarget (Black Recoloring Color)	Specifies the color to which black should be recolored.
	
	[*Example:*
	
	`<v:imagedata r:id="rId4" recolortarget="red">`
	`</v:imagedata>`

Attributes	Description
	 `no recolor`  `recolortarget="red"` *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).

relid (Relationship to Part)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies the relationship ID of the relationship to the image. The specified relationship shall be of type http://schemas.openxmlformats.org/officeDocument/2006/relationships/image or the document shall be considered non-conformant.
	
	[*Example*: The markup specifies the associated relationship part with relationship ID `rId10` contains the corresponding relationship information:
	
	`<v:imagedata ... o:relid="rId10" ...>`
	`</v:imagedata>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_RelationshipId simple type (§6.2.3.20).

src (Image Source)	Specifies the URL of the image to use.
	
	[*Example:*
	
	`<v:image ... src="myimage.gif" ...>`
	`</v:image>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

title (Image Data Title) Namespace:	Specifies the title of an embedded image. This is typically set to the comment property of the image, which is often blank. [*Example:*

Attributes	Description
urn:schemas-microsoft-com:office:office	<v:fill ... o:title="alt text" ... > </v:fill> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ImageData">
   <attributeGroup ref="AG_Id"/>
   <attributeGroup ref="AG_ImageAttributes"/>
   <attributeGroup ref="AG_Chromakey"/>
   <attribute name="embosscolor" type="ST_ColorType" use="optional"/>
   <attribute name="recolortarget" type="ST_ColorType"/>
   <attribute ref="o:href"/>
   <attribute ref="o:althref"/>
   <attribute ref="o:title"/>
   <attribute ref="o:oleid"/>
   <attribute ref="o:detectmouseclick"/>
   <attribute ref="o:movie"/>
   <attribute ref="o:relid"/>
   <attribute ref="r:id"/>
   <attribute ref="r:pict"/>
   <attribute ref="r:href"/>
</complexType>
```

6.1.2.12 line (Line)

This element draws a straight line.

[*Example:*

```
<v:line from="10pt,10pt" to="75pt,35pt"
  strokecolor="blue" strokeweight="3pt">
</v:line>
```

end example]

Parent Elements
background (§2.2.1); group (§6.1.2.7); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23)

Child Elements	Subclause

Child Elements	Subclause
anchorlock (Anchor Location Is Locked)	§6.3.2.1
borderbottom (Bottom Border)	§6.3.2.2
borderleft (Left Border)	§6.3.2.3
borderright (Right Border)	§6.3.2.4
bordertop (Top Border)	§6.3.2.5
callout (Callout)	§6.2.2.2
ClientData (Attached Object Data)	§6.4.2.12
clippath (Shape Clipping Path)	§6.2.2.3
extrusion (3D Extrusion)	§6.2.2.10
fill (Shape Fill Properties)	§6.1.2.5
formulas (Set of Formulas)	§6.1.2.6
handles (Set of Handles)	§6.1.2.9
imagedata (Image Data)	§6.1.2.11
lock (Shape Protections)	§6.2.2.17
path (Shape Path)	§6.1.2.14
shadow (Shadow Effect)	§6.1.2.18
signatureline (Digital Signature Line)	§6.2.2.29
skew (Skew Transform)	§6.2.2.30
stroke (Line Stroke Settings)	§6.1.2.21
textbox (Text Box)	§6.1.2.22
textdata (VML Diagram Text)	§6.5.2.2
textpath (Text Layout Path)	§6.1.2.23
wrap (Text Wrapping)	§6.3.2.6

Attributes	Description
allowincell (Allow in Table Cell) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape can be placed in a table. Default is `false`. [*Example:* `<v:shape ... o:allowincell="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

Attributes	Description
allowoverlap (Allow Shape Overlap) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape can overlap another shape. Default is `true`. If `false`, the shape will shift left or right so as not to overlap another shape, similar to the behavior of the HTML float attribute. [*Example:* `<v:shape ... o:allowoverlap="false" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

alt (Alternate Text)	Specifies alternative text describing the graphical object. This text should provide a brief description of the shape for use by accessibility tools. Default is no value.
	
	[*Example:* The alt text describes the basic shape:
	
	`<v:shape ... fillcolor="red" alt="Red rectangle"></v:shape>`
	
	The alt text describes the contents of a shape displaying an image:
	
	`<v:shape ... alt="Picture of a sunset"></v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

borderbottomcolor (Bottom Border Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the bottom border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:borderbottomcolor="red" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
borderleftcolor (Border Left Color) Namespace: urn:schemas-	Specifies the left border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:borderleftcolor="red" ... >`

Attributes	Description
microsoft-com:office:office	</v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
borderrightcolor (Border Right Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the right border color of an inline shape. Default is no value. [*Example:* <v:shape ... o:borderrightcolor="red" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
bordertopcolor (Border Top Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the top border color of an inline shape. Default is no value. [*Example:* <v:shape ... o:bordertopcolor="red" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
bullet (Graphical Bullet) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the shape is a graphical bullet. Default is `false`. [*Example:* <v:shape ... o:bullet="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
button (Button Behavior Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape will exhibit button press behavior on click. Default is `false`. [*Example:* <v:shape ... o:button="true" ... > </v:shape> *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
bwmode (Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies how a shape will render for black-and-white output devices. When a shape is printed on a black-and-white printer or displayed in a black-and-white view in an application, several options are possible. Default is auto, which will use o:bwnormal for normal black-and-white rendering and o:bwpure for pure black-and-white rendering. bwnormal and bwpure are subordinate to bwmode. If bwmode is "auto" then the value for bwnormal or bwpure is used depending on what the output format is. An application may define for itself what, if any, difference there is between normal B&W and pure B&W. For example, normal B&W might allow greyscale and pure B&W might not. [*Example:* This shape renders in grayscale in a black-and-white environment: `<v:shape ... o:bwmode="grayscale" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
bwnormal (Normal Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies the black-and-white mode for normal black-and-white output devices. Default is auto. [*Example:* This shape renders in a pale grayscale in a normal black-and-white environment: `<v:shape ... o:bwmode="auto" o:bwnormal="lightgrayscale" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
bwpure (Pure Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies the black-and-white mode for pure black-and-white output devices. Default is auto. [*Example:* This shape renders in high contrast when in a pure black-and-white environment: `<v:shape ... o:bwmode="auto" o:bwpure="highcontrast" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type

Attributes	Description
	(§6.2.3.2).

chromakey (Image Transparency Color)	Specifies a color value that will be transparent and show anything behind the shape. Default is no value.
	
	[*Example:*
	
	`<v:image ... chromakey="white" ...>`
	`</v:image>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).

class (CSS Reference)	Specifies a reference to the definition of a CSS style. Default is no value.
	
	[*Example:* The snippets below are equivalent:
	
	`...`
	`.narrowstyle {width:50;height:100}`
	`...`
	`<v:shape ... class="narrowstyle"`
	`style="top:1;left:1">`
	`</v:shape>`
	
	`<v:shape ... style="top:1;left:1;`
	`width:50;height:100">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

clip (Clipping Toggle)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies that the clipping region is active. This is used in conjunction with the clippath (§6.2.2.3) element to create a clipping region.
	
	[*Example:*
	
	`<v:shape ... o:clip="true">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

cliptowrap (Clip to	Specifies that the clipping region for the shape aligns with the wrapping polygon that

Attributes	Description

Wrapping Polygon)
 	
Namespace: urn:schemas-microsoft-com:office:office
	tightly surrounds the entire shape (essentially, that the shape shall not be drawn beyond its wrapping polygon's extents – if it does, the shape shall be clipped). Default is `false`.
	
	[*Example:*
	
	`<v:shape ... o:cliptowrap="true">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

connectortype (Shape Connector Type)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies the type of connector used for joining shapes. Default is `straight`.
	
	[*Example:*
	
	`<v:shape ... o:connectortype="elbow" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_ConnectorType simple type (§6.2.3.6).

coordorigin (Coordinate Space Origin)	Specifies the coordinate of the top left corner of the shape's coordinate space. This determines the position of the (0,0) coordinate space origin within the shape's bounding box. Default is "0,0", which places the (0,0) origin at the top left corner of the bounding box.
	
	This affects shape properties that specify coordinate positions, such as the path attribute. Thus a path can be defined against a generic (0,0) origin and the coordorigin value translates the entire path within the shape's bounding space.
	
	[*Example:* The horizontal and vertical coordinate space ranges from -100 to +100 because the coordinate space (coordsize) is 200 by 200 and the top left coordinate is (-100,-100). The (0,0) origin lies at the center of the shape's bounding box, as evidenced by the position of the shape's path within the coordinate space:
	
	`<v:shape ... coordsize="200,200"`
	`coordorigin="-100,-100"`
	`path="m 0,0 l 0,50, 50,50, 50,0 x e">`
	`</v:shape>`
	
	 

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>coordsize
(Coordinate Space Size)</td><td>Specifies the size of the shape's coordinate space in coordinate units. Default is "1000,1000".

The physical size of a coordinate unit length is determined by both the size of the coordinate space (coordsize) and the size of the shape (style width and height). The coordsize attribute defines the number of horizontal and vertical subdivisions into which the shape's bounding box is divided. The combination of coordsize and style width/height effective scales the shape anisotropically.

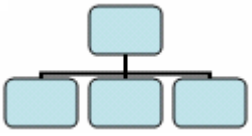
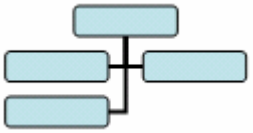
[Example: The path is 50 units wide and tall, which is 25% of the size of the coordinate space:

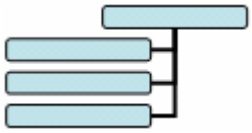
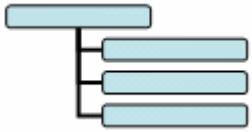
<pre><v:shape ... coordsize="200,200"
  coordorigin="-100,-100"
  path="m 0,0 l 0,50, 50,50, 50,0 x e">
</v:shape></pre>
 

end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>dgmlayout
(Diagram Node Layout Identifier)

Namespace:
urn:schemas-microsoft-com:office:office</td><td>Specifies the type of automatic layout to apply to the diagram node. This is only meaningful for organization charts. Valid values for organization charts are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Top-down, centered layout.
  </td></tr>
<tr><td>1</td><td>Hanging, both sides layout.
  </td></tr>
<tr><td>2</td><td>Hanging, right side layout.</td></tr>
</table>
</td></tr>
</table>

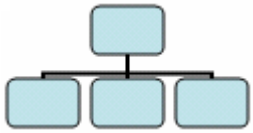
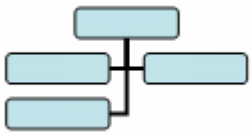
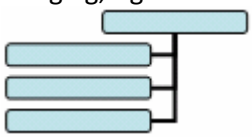
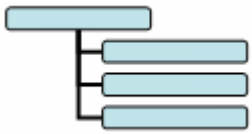
<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>  </td></tr>
<tr><td>3</td><td>Hanging, left side layout.
  </td></tr>
</table>

[Example:

<pre><v:shape ... dgmlayout="1">
</v:shape></pre>
end example]

The possible values for this attribute are defined by the XML Schema integer datatype.</td></tr>
<tr><td>dgmlayoutmru (Diagram Node Recent Layout Identifier)

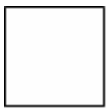
Namespace: urn:schemas-microsoft-com:office:office</td><td>Specifies the type of automatic layout most recently used on the child elements of the diagram node. This is only meaningful for organization charts. Valid values for organization charts are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Top-down, centered layout.
  </td></tr>
<tr><td>1</td><td>Hanging, both sides layout.
  </td></tr>
<tr><td>2</td><td>Hanging, right side layout.
  </td></tr>
<tr><td>3</td><td>Hanging, left side layout.
  </td></tr>
</table>

[Example:</td></tr>
</table>

Attributes	Description
	<v:shape ... dgmlayout="1"> </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema integer datatype.
dgmnodekind (Diagram Node Identifier) Namespace: urn:schemas-microsoft-com:office:office	Specifies an optional, application-specific parameter that is intended to be used by the application to tag different types of nodes in a diagram. [*Example:* <v:shape ... dgmnodekind="1"> </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema integer datatype.
doubleclicknotify (Double-click Notification Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies that an event message is sent when a shape is double-clicked. Default is `false`. [*Example:* <v:shape ... o:doubleclicknotify="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
fillcolor (Fill Color)	Specifies the color to use for the fill. Default is `white`. If the fill element (§6.1.2.5) is present, its color attribute takes precedence. Colors are typically specified as either a named color, such as `red`, or six hexadecimal digits representing the red, green and blue values of the color, such as `#00FF30`. Full details are specified in the simple type description. [*Example:* This shape is red if its fill is visible: <v:shape ... fillcolor="red" ... > </v:shape> This is equivalent to: <v:shape ... fillcolor="#ff0000" ... > </v:shape>

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).

filled (Shape Fill Toggle)	Specifies whether the closed path will be filled. Default is `true`. This attribute is overridden by the fill on attribute.
	
	[*Example:*
	
	`<v:shape ... filled="f"`
	`  fillcolor="red" ...>`
	`</v:shape>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).

forcedash (Force Dashed Outline)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies whether a dashed outline is used to draw a shape when a shape has no line or fill. Default is `false`.
	
	Used by PresentationML placeholders to draw a dashed outline when there is no line and no fill for a shape.
	
	[*Example:*
	
	`<v:shape ... o:forcedash="true" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

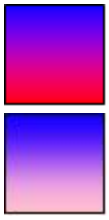
from (Line Start)	Specifies the starting point of the line in the coordinate space of the parent element. Default is "0,0". If the parent is not a VML element, the default unit is a pixel. Allowed units are in, cm, mm, pt, pc and px.
	
	[*Example:*
	
	`<v:line from="10pt,10pt" to="50pt,50pt">`
	`</v:line>`

Attributes	Description
	 *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
hr (Horizontal Rule Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies that a shape is a horizontal rule. Default is `false`. [*Example:* `<v:shape ... o:hr="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
hralign (Horizontal Rule Alignment) Namespace: urn:schemas-microsoft-com:office:office	Specifies the alignment of a horizontal rule. Default is `left`. [*Example:* `<v:shape ... o:hralign="center" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_HrAlign simple type (§6.2.3.14).
href (Hyperlink Target)	Specifies a hyperlink URL target for the shape. Default is no value. [*Example:* `<v:shape ... href="http://www.openxmlformats.org" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
hrnoshade (Horizontal Rule 3D Shading Toggle) Namespace: urn:schemas-	Specifies that the horizontal rule does not have 3-D shading. Default is `false`. [*Example:* `<v:shape ... o:hrnoshade="true" ... ></v:shape>`

Attributes	Description
microsoft-com:office:office	*end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
hrpct (Horizontal Rule Length Percentage) Namespace: urn:schemas-microsoft-com:office:office	Specifies the length of a horizontal rule as a percentage of page width. Default is 0. [*Example:* `<v:shape ... o:hrpct="85" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema float datatype.
hrstd (Horizontal Rule Standard Display Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape is displayed as a standard horizontal rule. Only applies if hr is `true`. Default is `false`. [*Example:* `<v:shape ... o:hrstd="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
id (Unique Identifier)	Specifies a unique identifier that can be used to reference a VML object. Default is no value. [*Example:* `<v:shape ... id="myShape" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
insetmode (Text Inset Mode) Namespace: urn:schemas-microsoft-	Specifies whether the application calculates the internal text margin instead of using the inset attribute. Default is `custom`. This attribute is only meaningful for text boxes. [*Example:* `<v:shape ... o:insetmode="auto" ... >`

Attributes	Description
com:office:office	</v:shape> *end example*] The possible values for this attribute are defined by the ST_InsetMode simple type (§6.2.3.15).
insetpen (Inset Border From Path)	Specifies that the border shall be displayed inside of the path defining the shape, rather than along the path (the default border placement), or outside of the path as might be done with an image. [*Example:* <v:shape ... insetpen="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
ole (Embedded Object Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the shape is an embedded object. Default is `false`. [*Example:* <v:shape ... o:ole="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalseBlank simple type (§6.2.3.24).
oleicon (Embedded Object Icon Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether an embedded object will be displayed as an icon. Default is `false`. [*Example:* <v:shape ... o:oleicon="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
oned (Shape Handle Toggle) Namespace:	Specifies whether the extra handles of a shape are hidden. If `true`, hides all shape handles except the top left and bottom right; that is, the same handles that are used for a straight line segment. Default is `false`.

Attributes	Description
urn:schemas-microsoft-com:office:office	[*Example:* `<v:shape ... o:oned="true" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

opacity (Fill Color Opacity)	Specifies the opacity of the primary fill color. Default is 1.0. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8.
	
	[*Example:* The red color is 25% opaque:
	
	`<v:fill type="gradient" color="red"`
	`color2="blue" opacity=".25">`
	`</v:fill>`
	
	  `opacity="1"`
	
	 `opacity=".25"`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

preferrelative (Relative Resize Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the original size of an object is saved after reformatting. Default is `false`. If `true`, the original size of the object is stored and all resizing is based on a percentage of that original size. Otherwise, each resizing will reset the scale to 100%. [*Example:* `<v:shape ... o:preferrelative="true" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
print (Print Toggle)	Specifies whether the shape will be printed. Default is `true`. [*Example:*

Attributes	Description
	<v:shape ... print="false" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
regroupid (Regroup ID) Namespace: urn:schemas-microsoft-com:office:office	Specifies a previous group for a shape. An ID number is used to identify groups of shapes that are no longer grouped. This allows shapes to be regrouped programmatically. [*Example:* The shape was part of a group identified by the ID 040754: <v:shape ... o:regroupid="040754" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema integer datatype.
spid (Optional String) Namespace: urn:schemas-microsoft-com:office:office	Specifies an optional string that an application may use to Identify the particular shape. Default is no value. The possible values for this attribute are defined by the XML Schema string datatype.
spt (Optional Number) Namespace: urn:schemas-microsoft-com:office:office	Specifies an optional number that an application may use to associate the particular shape with a defined shape type. Default is 0. The possible values for this attribute are defined by the XML Schema float datatype.
strokecolor (Shape Stroke Color)	Specifies the primary color of the brush to use to stroke the path of the shape. Default is black. The color attribute of the stroke element (§6.1.2.21) overrides this. Colors are typically specified as either a named color, such as red, or six hexadecimal digits representing the red, green and blue values of the color, such as #00FF30. Full details are specified in the simple type description. [*Example:* <v:shape ... strokecolor="red" ...> </v:shape>

Attributes	Description
	 *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).

stroked (Shape Stroke Toggle)	Specifies whether the path defining the shape is stroked with a solid line. The stroke element (§6.1.2.21) defines other strokes. The on attribute of the stroke element overrides this attribute. Default is `true`.
	
	[*Example:*
	
	`<v:shape ... fillcolor="red"`
	`  stroked="false" strokecolor="blue"...>`
	`</v:shape>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).

strokeweight (Shape Stroke Weight)	Specifies the width of the brush to use to stroke the path. Default is 1 point. If a number is given without units, the emu is used. The weight attribute of the stroke element (§6.1.2.21) overrides this attribute.
	
	[*Example:*
	
	`<v:shape ... strokeweight="3pt" ... >`
	`</v:shape>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

style (Shape Styling Properties)	Specifies the CSS2 styling properties of the shape. This uses the syntax described in the "Visual formatting model" of the Cascading Style Sheets, Level 2 specification, a Recommendation of the World Wide Web Consortium available here:

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>http://www.w3.org/TR/REC-CSS2. Full descriptions of each property are not repeated here, but the VML treatment of each property is defined. Allowed properties include:

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>flip</td><td>Specifies that the orientation of a shape is flipped. Default is no value. Allowed values are:

• x - Flip along the y-axis, reversing the x-coordinates.
• y - Flip along the x-axis, reversing the y-coordinates.
• xy - Flip along both the y- and x-axis.
• yx - Flip along both the x- and y-axis.</td></tr>
<tr><td>height</td><td>Specifies the height of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are:

• auto - Default position of an element in the flow of the page.
• <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed.
• <percentage>- Value expressed as a percentage of the parent object's height.</td></tr>
<tr><td>left</td><td>Specifies the position of the left of the containing block of the shape relative to the element left of it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are:

• auto - Default position of an element in the flow of the page.
• <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed.
• <percentage>- Value expressed as a percentage of the parent object's width.</td></tr>
<tr><td>margin-bottom</td><td>Specifies the position of the bottom of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are:

• auto - Default position of an element in the flow of the page.
• <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or</td></tr>
</table>
</td></tr>
</table>

Attributes	Description	
		ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	margin-left	Specifies the position of the left of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	margin-right	Specifies the position of the right of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	margin-top	Specifies the position of the top of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	mso-position-horizontal	Specifies the horizontal positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute

Attributes	Description
	• left • center • right • inside • outside
mso-position-horizontal-relative	Specifies relative horizontal position data for objects in WordprocessingML documents. This modifies the mso-position-horizontal property. Default is text. Allowed values are: • margin • page • text • char
mso-position-vertical	Specifies the vertical positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute • top • center • bottom • inside • outside
mso-position-vertical-relative	Specifies relative vertical position data for objects in WordprocessingML documents. This modifies the mso-position-vertical property. Default is text. Allowed values are: • margin • page • text • line
mso-wrap-distance-bottom	Specifies the distance from the bottom of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-distance-left	Specifies the distance from the left side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin

Attributes	Description
	of the shape to include the margin areas. This property does not change the origin.
mso-wrap-distance-right	Specifies the distance from the right side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-distance-top	Specifies the distance from the top of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-edited	Specifies whether the wrap coordinates were customized by the user. If the wrap coordinates are generated by an editor, this property is true; otherwise they were customized by a user. Default is false.
mso-wrap-style	Specifies the wrapping mode for text in shapes in WordprocessingML documents. Default is square. Allowed values are: • square - Wraps text inside the shape in a square. • none - Text does not wrap.
position	Specifies the type of positioning used to place an element. Default is static. When the element is contained inside a group, this property must be absolute. Allowed values are: • static - The element is positioned according to the normal flow of the page. The top and left properties are ignored. If the object is anchored inline, this value is used. • absolute - The element is positioned relative to the parent, using the top and left properties. • relative - The element is positioned according to the normal flow of the page, but the top and left properties are used. The overlap of overlapping elements is governed by the z-index property.
rotation	Specifies the angle that a shape is rotated, in degrees. Default is 0. Positive angles are clockwise.
top	Specifies the position of the top of the containing block of the shape relative to the element above it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are:

Attributes	Description
	<table><tr><td></td><td> - auto - Default position of an element in the flow of the page. <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. <percentage>- Value expressed as a percentage of the parent object's height. </td></tr><tr><td>visibility</td><td>Specifies whether a shape is displayed. Only inherit and hidden are used; any other values are mapped to inherit. Default is inherit. Allowed values are: - hidden - The shape is not visible, but is still part of the flow of the objects in the browser. Mouse events are not processed. - inherit - The visibility state is inherited from the parent of the shape. </td></tr><tr><td>width</td><td>Specifies the width of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: - auto - Default position of an element in the flow of the page. <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. <percentage>- Value expressed as a percentage of the parent object's width. </td></tr><tr><td>z-index</td><td>Specifies the display order of overlapping shapes. Default is 0. This property shall not be used for shapes anchored inline. Allowed values are: - auto - Uses the order that the shapes appear in the page, bottom to top. <order>- A number that represents the stacking precedence. Shapes with higher numbers are placed on top of those with lower numbers. Negative numbers are allowed. </td></tr></table> The following properties are only used by the textbox element (§6.1.2.22): <table><tr><th>Property</th><th>Description</th></tr></table>

Attributes	Description	
	direction	Specifies the direction of the text in the textbox. Default is ltr. This property is superceded by the mso-direction-alt property if that is specified. Allowed values are: • ltr - Test is displayed left-to-right. • rtl - Test is displayed right-to-left.
	layout-flow	Determines the flow of the text layout in a textbox. Default is horizontal. Allowed values are: • horizontal - Text is displayed horizontally. • vertical - Text is displayed vertically. • vertical-ideographic - Ideographic text is displayed vertically. • horizontal-ideographic - Ideographic text is displayed horizontally.
	mso-direction-alt	Specifies an alternate direction for text in textboxes. Overrides the direction property. The only allowed value is context.
	mso-fit-shape-to-text	Specifies whether the shape stretches to fit the text in the textbox. Default is false.
	mso-fit-text-to-shape	Specifies whether the text stretches to fit the textbox. Default is false.
	mso-layout-flow-alt	Specifies the alternate layout flow for text in textboxes. This property is used instead of layout-flow when the layout flow is from bottom to top for non-ideographic languages. Its only value is bottom-to-top.
	mso-next-textbox	Specifies the ID of the next textbox in a series. Used to keep track of a set of linked textboxes. Default is no value.
	mso-rotate	Specifies a specific rotation value for text in a textbox. Default is 0. Allowed values are: • 0 • 90 • 180 • -90
	mso-text-scale	Specifies the scaling factor for fitting text to shapes. Default is 0. This property is only used if mso-fit-text-to-shape is true.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>v-text-anchor</td><td>Specifies the vertical anchoring of text in a textbox. Default is top. The alignment of a text anchor only becomes evident if mso-fit-text-to-shape is false. This property is different from the vertical-align CSS property, which is used for ideographic languages. Allowed values are:
top
middle
bottom
top-center
middle-center
bottom-center
top-baseline
bottom-baseline
top-center-baseline
bottom-center-baseline</td></tr>
</table>
The following properties are only used by the textpath element (§6.1.2.23):
<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>font</td><td>Specifies a compound value of font settings. Default is no value. The values are the same as those of the CSS font property. The order of definitions in the string is: font-style, font-variant, font-weight, font-size, line-height, font-family.</td></tr>
<tr><td>font-family</td><td>Specifies the family of the font. Default is no value. The values are the same as those of the CSS font-family property.</td></tr>
<tr><td>font-size</td><td>Specifies the size of the font. Default is no value. The font size is defined in points. The values are the same as those of the CSS font-size property.</td></tr>
<tr><td>font-style</td><td>Specifies the amount of slant for a font. Default is normal. The values are the same as those of the CSS font-style property. Allowed values are:
normal
italic
oblique - Treated the same as italic.</td></tr>
<tr><td>font-variant</td><td>Specifies the variant style of a font. Default is normal. The values are the same as those of the CSS font-variant property. Allowed values are:
normal
small-caps</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>font-weight</td><td>Specifies the thickness of the letters of the font. Default is normal. The values are the same as those of the CSS font-weight property. Allowed values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>normal
lighter
100
200
300
400</td><td>Treated as non-bold.</td></tr>
<tr><td>bold
bolder
500
600
700
800
900</td><td>Treated as bold.</td></tr>
</table>
</td></tr>
<tr><td>mso-text-shadow</td><td>Specifies whether a shadow is applied to the text on a text path. Default is false.</td></tr>
<tr><td>text-decoration</td><td>Specifies the style of text decoration. Default is none. The values are the same as those of the CSS text-decoration property. Allowed values are:
none
underline
overline
line-through
blink</td></tr>
<tr><td>v-rotate-letters</td><td>Specifies whether the letters of the text are rotated counterclockwise by 90 degrees. Default is false.</td></tr>
<tr><td>v-same-letter-heights</td><td>Specifies whether all letters will be the same height regardless of initial case. If true, the lowercase letters are stretched to the</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>height of the uppercase letters. Default is <code>false</code>.</td></tr>
<tr><td><code>v-text-align</code></td><td>Specifies the alignment of text. Default is <code>left</code>. Allowed values are:
<code>left</code>
<code>right</code>
<code>center</code>
<code>justify</code>
<code>letter-justify</code> - Distributes the extra space between the letters.
<code>stretch-justify</code> - Stretches the letters to fill in the space.</td></tr>
<tr><td><code>v-text-kern</code></td><td>Specifies whether kerning is turned on. Default is <code>false</code>.</td></tr>
<tr><td><code>v-text-reverse</code></td><td>Specifies whether the layout order of rows is reversed. Default is <code>false</code>. This is used for vertical text layout.</td></tr>
<tr><td><code>v-text-spacing-mode</code></td><td>Specifies the mode for letter spacing. Default is <code>tightening</code>. This property determines whether space will be removed between each letter (<code>tightening</code>) or added between each letter (<code>tracking</code>). The amount of letter spacing change is defined by the <code>v-text-spacing</code> property. Allowed values are:
<code>tightening</code>
<code>tracking</code></td></tr>
<tr><td><code>v-text-spacing</code></td><td>Specifies the amount of spacing for text in 100ths of single line spacing. Default is 100.</td></tr>
</table>
The line (§6.1.2.12), polyline (§6.1.2.15) and curve (§6.1.2.3) elements ignore the following properties:
<code>top</code>
<code>left</code>
<code>width</code>
<code>height</code>
The following properties are not inherited by an element that references a shapetype element (§6.1.2.20) via the id attribute:
<code>flip</code>
<code>height</code>
<code>left</code>
<code>margin-left</code>
<code>margin-top</code>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>

position
rotation
top
visibility
width
z-index

The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>target (Hyperlink Display Target)</td><td>
Specifies a frame or window that a URL is displayed in. Default is no value. Allowed values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td><targetname></td><td>String containing the name of the frame or window in which to load the document.</td></tr>
<tr><td>_blank</td><td>Specifies that the linked document is loaded into a new blank window. This window is not named.</td></tr>
<tr><td>_media</td><td>Specifies that the linked document is loaded into the Media Bar. Available in Microsoft® Internet Explorer 6 or later.</td></tr>
<tr><td>_parent</td><td>Specifies that the linked document is loaded into the immediate parent of the document containing the link.</td></tr>
<tr><td>_search</td><td>Specifies that the linked document is loaded into the browser's search pane. Available in Microsoft Internet Explorer 5 or greater.</td></tr>
<tr><td>_self</td><td>Specifies that the linked document is loaded into the window in which the link was clicked (the active window).</td></tr>
<tr><td>_top</td><td>Specifies that the linked document is loaded into the topmost window.</td></tr>
</table>
[Example:
<pre>
  <v:shape ...
    href="http://www.openxmlformats.org"
    target="_self" ... >
  </v:shape>
</pre>
end example]
The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>title (Shape Title)</td><td>
Specifies the text displayed when the mouse pointer moves over the shape. Default is no value.
[Example:
</td></tr>
</table>

Attributes	Description
	<v:shape ... title="tooltip" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
to (Line End Point)	Specifies the ending point of the line in the coordinate space of the parent element. Default is "10,10". If the parent is not a VML element, the default unit is a pixel. Allowed units are in, cm, mm, pt, pc and px. [*Example:* <v:line from="10pt,10pt" to="50pt,50pt"> </v:line>  *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
userdrawn (Exists In Master Slide) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the user has added the shape to a master slide. Default is `false`. Used by PresentationML. [*Example:* <v:shape ... o:userdrawn="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
userhidden (Hide Script Anchors) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a script anchor is hidden. Default is `false`. If `true`, script anchors stay hidden even if the shape is otherwise visible. A script anchor is the visual representation of a script that when displayed in an application. [*Example:* <v:shape ... o:userhidden="true" ... > </v:shape> *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

wrapcoords (Shape Bounding Polygon)	Specifies the bounding polygon that surrounds a shape. This is specified using a comma-delimited list of x and y coordinates: "x1,y1,x2,y2,x3,y3,..." This is used when text is tightly wrapped around a shape. Default is no value until the mso-wrap-mode style attribute is set to tight or through.
	
	[*Example:*
	
	`<v:shape ...`
	`  wrapcoords="0,0 0,200, 200,200, 200,0" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Line">
   <sequence>
      <group ref="EG_ShapeElements" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attributeGroup ref="AG_AllCoreAttributes"/>
   <attributeGroup ref="AG_AllShapeAttributes"/>
   <attribute name="from" type="xsd:string" use="optional"/>
   <attribute name="to" type="xsd:string" use="optional"/>
</complexType>
```

6.1.2.13 oval (Oval)

This element draws an oval sized according to the CSS2 style content width and height.

[*Example:*

```
<v:oval fillcolor="blue"
  style="position:relative;top:1;left:1;width:150;height:50">
</v:oval>
```

end example]

Parent Elements
background (§2.2.1); group (§6.1.2.7); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23)

Child Elements	Subclause
anchorlock (Anchor Location Is Locked)	§6.3.2.1
borderbottom (Bottom Border)	§6.3.2.2
borderleft (Left Border)	§6.3.2.3
borderright (Right Border)	§6.3.2.4
bordertop (Top Border)	§6.3.2.5
callout (Callout)	§6.2.2.2
ClientData (Attached Object Data)	§6.4.2.12
clippath (Shape Clipping Path)	§6.2.2.3
extrusion (3D Extrusion)	§6.2.2.10
fill (Shape Fill Properties)	§6.1.2.5
formulas (Set of Formulas)	§6.1.2.6
handles (Set of Handles)	§6.1.2.9
imagedata (Image Data)	§6.1.2.11
lock (Shape Protections)	§6.2.2.17
path (Shape Path)	§6.1.2.14
shadow (Shadow Effect)	§6.1.2.18
signatureline (Digital Signature Line)	§6.2.2.29
skew (Skew Transform)	§6.2.2.30
stroke (Line Stroke Settings)	§6.1.2.21
textbox (Text Box)	§6.1.2.22
textdata (VML Diagram Text)	§6.5.2.2
textpath (Text Layout Path)	§6.1.2.23
wrap (Text Wrapping)	§6.3.2.6

Attributes	Description
allowincell (Allow in Table Cell) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape can be placed in a table. Default is `false`. [*Example:* `<v:shape ... o:allowincell="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

Attributes	Description
allowoverlap (Allow Shape Overlap) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape can overlap another shape. Default is `true`. If `false`, the shape will shift left or right so as not to overlap another shape, similar to the behavior of the HTML float attribute. [*Example:* `<v:shape ... o:allowoverlap="false" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

alt (Alternate Text)	Specifies alternative text describing the graphical object. This text should provide a brief description of the shape for use by accessibility tools. Default is no value.
	
	[*Example:* The alt text describes the basic shape:
	
	`<v:shape ... fillcolor="red" alt="Red rectangle"></v:shape>`
	
	The alt text describes the contents of a shape displaying an image:
	
	`<v:shape ... alt="Picture of a sunset"></v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

borderbottomcolor (Bottom Border Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the bottom border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:borderbottomcolor="red" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
borderleftcolor (Border Left Color) Namespace: urn:schemas-	Specifies the left border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:borderleftcolor="red" ... >`

Attributes	Description
microsoft-com:office:office	</v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
borderrightcolor (Border Right Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the right border color of an inline shape. Default is no value. [*Example:* <v:shape ... o:borderrightcolor="red" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
bordertopcolor (Border Top Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the top border color of an inline shape. Default is no value. [*Example:* <v:shape ... o:bordertopcolor="red" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
bullet (Graphical Bullet) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the shape is a graphical bullet. Default is `false`. [*Example:* <v:shape ... o:bullet="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
button (Button Behavior Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape will exhibit button press behavior on click. Default is `false`. [*Example:* <v:shape ... o:button="true" ... > </v:shape> *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
bwmode (Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies how a shape will render for black-and-white output devices. When a shape is printed on a black-and-white printer or displayed in a black-and-white view in an application, several options are possible. Default is auto, which will use o:bwnormal for normal black-and-white rendering and o:bwpure for pure black-and-white rendering. bwnormal and bwpure are subordinate to bwmode. If bwmode is "auto" then the value for bwnormal or bwpure is used depending on what the output format is. An application may define for itself what, if any, difference there is between normal B&W and pure B&W. For example, normal B&W might allow greyscale and pure B&W might not. [*Example:* This shape renders in grayscale in a black-and-white environment: `<v:shape ... o:bwmode="grayscale" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
bwnormal (Normal Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies the black-and-white mode for normal black-and-white output devices. Default is auto. [*Example:* This shape renders in a pale grayscale in a normal black-and-white environment: `<v:shape ... o:bwmode="auto" o:bwnormal="lightgrayscale" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
bwpure (Pure Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies the black-and-white mode for pure black-and-white output devices. Default is auto. [*Example:* This shape renders in high contrast when in a pure black-and-white environment: `<v:shape ... o:bwmode="auto" o:bwpure="highcontrast" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type

Attributes	Description
	(§6.2.3.2).

chromakey (Image Transparency Color)	Specifies a color value that will be transparent and show anything behind the shape. Default is no value.
	
	[*Example:*
	
	`<v:image ... chromakey="white" ...>`
	`</v:image>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).

class (CSS Reference)	Specifies a reference to the definition of a CSS style. Default is no value.
	
	[*Example:* The snippets below are equivalent:
	
	`...`
	`.narrowstyle {width:50;height:100}`
	`...`
	`<v:shape ... class="narrowstyle"`
	`style="top:1;left:1">`
	`</v:shape>`
	
	`<v:shape ... style="top:1;left:1;`
	`width:50;height:100">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

clip (Clipping Toggle)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies that the clipping region is active. This is used in conjunction with the clippath (§6.2.2.3) element to create a clipping region.
	
	[*Example:*
	
	`<v:shape ... o:clip="true">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

cliptowrap (Clip to	Specifies that the clipping region for the shape aligns with the wrapping polygon that

Attributes	Description

Wrapping Polygon)
 	
Namespace: urn:schemas-microsoft-com:office:office
	tightly surrounds the entire shape (essentially, that the shape shall not be drawn beyond its wrapping polygon's extents – if it does, the shape shall be clipped). Default is `false`.
	
	[*Example:*
	
	`<v:shape ... o:cliptowrap="true">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

connectortype (Shape Connector Type)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies the type of connector used for joining shapes. Default is `straight`.
	
	[*Example:*
	
	`<v:shape ... o:connectortype="elbow" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_ConnectorType simple type (§6.2.3.6).

coordorigin (Coordinate Space Origin)	Specifies the coordinate of the top left corner of the shape's coordinate space. This determines the position of the (0,0) coordinate space origin within the shape's bounding box. Default is "0,0", which places the (0,0) origin at the top left corner of the bounding box.
	
	This affects shape properties that specify coordinate positions, such as the path attribute. Thus a path can be defined against a generic (0,0) origin and the coordorigin value translates the entire path within the shape's bounding space.
	
	[*Example:* The horizontal and vertical coordinate space ranges from -100 to +100 because the coordinate space (coordsize) is 200 by 200 and the top left coordinate is (-100,-100). The (0,0) origin lies at the center of the shape's bounding box, as evidenced by the position of the shape's path within the coordinate space:
	
	`<v:shape ... coordsize="200,200"`
	`coordorigin="-100,-100"`
	`path="m 0,0 l 0,50, 50,50, 50,0 x e">`
	`</v:shape>`
	
	 

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>coordsize
(Coordinate Space Size)</td><td>Specifies the size of the shape's coordinate space in coordinate units. Default is "1000,1000".

The physical size of a coordinate unit length is determined by both the size of the coordinate space (coordsize) and the size of the shape (style width and height). The coordsize attribute defines the number of horizontal and vertical subdivisions into which the shape's bounding box is divided. The combination of coordsize and style width/height effective scales the shape anisotropically.

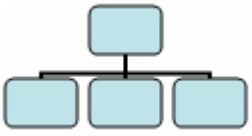
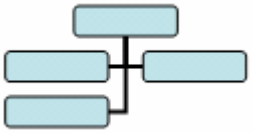
[Example: The path is 50 units wide and tall, which is 25% of the size of the coordinate space:

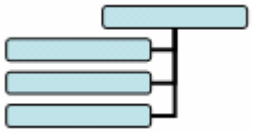
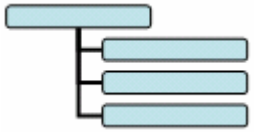
<pre><v:shape ... coordsize="200,200"
  coordorigin="-100,-100"
  path="m 0,0 l 0,50, 50,50, 50,0 x e">
</v:shape></pre>
 

end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>dgmlayout
(Diagram Node Layout Identifier)

Namespace:
urn:schemas-microsoft-com:office:office</td><td>Specifies the type of automatic layout to apply to the diagram node. This is only meaningful for organization charts. Valid values for organization charts are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Top-down, centered layout.
  </td></tr>
<tr><td>1</td><td>Hanging, both sides layout.
  </td></tr>
<tr><td>2</td><td>Hanging, right side layout.</td></tr>
</table>
</td></tr>
</table>

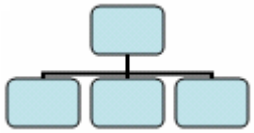
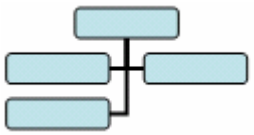
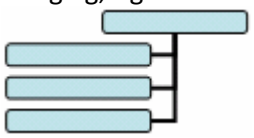
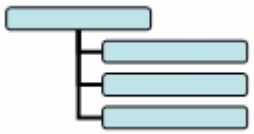
<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>  </td></tr>
<tr><td>3</td><td>Hanging, left side layout.
  </td></tr>
</table>

[Example:

<pre><v:shape ... dgmlayout="1">
</v:shape></pre>
end example]

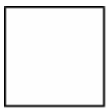
The possible values for this attribute are defined by the XML Schema integer datatype.</td></tr>
<tr><td>dgmlayoutmru (Diagram Node Recent Layout Identifier)

Namespace: urn:schemas-microsoft-com:office:office</td><td>Specifies the type of automatic layout most recently used on the child elements of the diagram node. This is only meaningful for organization charts. Valid values for organization charts are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Top-down, centered layout.
  </td></tr>
<tr><td>1</td><td>Hanging, both sides layout.
  </td></tr>
<tr><td>2</td><td>Hanging, right side layout.
  </td></tr>
<tr><td>3</td><td>Hanging, left side layout.
  </td></tr>
</table>

[Example:</td></tr>
</table>

Attributes	Description
	<v:shape ... dgmlayout="1"> </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema integer datatype.
dgmnodekind (Diagram Node Identifier) Namespace: urn:schemas-microsoft-com:office:office	Specifies an optional, application-specific parameter that is intended to be used by the application to tag different types of nodes in a diagram. [*Example:* <v:shape ... dgmnodekind="1"> </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema integer datatype.
doubleclicknotify (Double-click Notification Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies that an event message is sent when a shape is double-clicked. Default is `false`. [*Example:* <v:shape ... o:doubleclicknotify="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
fillcolor (Fill Color)	Specifies the color to use for the fill. Default is `white`. If the fill element (§6.1.2.5) is present, its color attribute takes precedence. Colors are typically specified as either a named color, such as `red`, or six hexadecimal digits representing the red, green and blue values of the color, such as `#00FF30`. Full details are specified in the simple type description. [*Example:* This shape is red if its fill is visible: <v:shape ... fillcolor="red" ... > </v:shape> This is equivalent to: <v:shape ... fillcolor="#ff0000" ... > </v:shape>

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).
filled (Shape Fill Toggle)	Specifies whether the closed path will be filled. Default is `true`. This attribute is overridden by the fill on attribute. [*Example:* `<v:shape ... filled="f" fillcolor="red" ...> </v:shape>`  *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
forcedash (Force Dashed Outline) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a dashed outline is used to draw a shape when a shape has no line or fill. Default is `false`. Used by PresentationML placeholders to draw a dashed outline when there is no line and no fill for a shape. [*Example:* `<v:shape ... o:forcedash="true" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
hr (Horizontal Rule Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies that a shape is a horizontal rule. Default is `false`. [*Example:* `<v:shape ... o:hr="true" ... > </v:shape>` *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
hralign (Horizontal Rule Alignment) Namespace: urn:schemas-microsoft-com:office:office	Specifies the alignment of a horizontal rule. Default is `left`. [*Example:* `<v:shape ... o:hralign="center" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_HrAlign simple type (§6.2.3.14).
href (Hyperlink Target)	Specifies a hyperlink URL target for the shape. Default is no value. [*Example:* `<v:shape ... href="http://www.openxmlformats.org" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
hrnoshade (Horizontal Rule 3D Shading Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies that the horizontal rule does not have 3-D shading. Default is `false`. [*Example:* `<v:shape ... o:hrnoshade="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
hrpct (Horizontal Rule Length Percentage) Namespace: urn:schemas-microsoft-com:office:office	Specifies the length of a horizontal rule as a percentage of page width. Default is 0. [*Example:* `<v:shape ... o:hrpct="85" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema float datatype.
hrstd (Horizontal	Specifies whether a shape is displayed as a standard horizontal rule. Only applies if hr is

Attributes	Description
Rule Standard Display Toggle) Namespace: urn:schemas-microsoft-com:office:office	`true`. Default is `false`. [*Example:* `<v:shape ... o:hrstd="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
id (Unique Identifier)	Specifies a unique identifier that can be used to reference a VML object. Default is no value. [*Example:* `<v:shape ... id="myShape" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
insetmode (Text Inset Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the application calculates the internal text margin instead of using the inset attribute. Default is `custom`. This attribute is only meaningful for text boxes. [*Example:* `<v:shape ... o:insetmode="auto" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_InsetMode simple type (§6.2.3.15).
insetpen (Inset Border From Path)	Specifies that the border shall be displayed inside of the path defining the shape, rather than along the path (the default border placement), or outside of the path as might be done with an image. [*Example:* `<v:shape ... insetpen="true" ... ></v:shape>` *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
ole (Embedded Object Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the shape is an embedded object. Default is `false`. [*Example:* `<v:shape ... o:ole="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalseBlank simple type (§6.2.3.24).
oleicon (Embedded Object Icon Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether an embedded object will be displayed as an icon. Default is `false`. [*Example:* `<v:shape ... o:oleicon="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
oned (Shape Handle Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the extra handles of a shape are hidden. If `true`, hides all shape handles except the top left and bottom right; that is, the same handles that are used for a straight line segment. Default is `false`. [*Example:* `<v:shape ... o:oned="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
opacity (Fill Color Opacity)	Specifies the opacity of the primary fill color. Default is 1.0. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8. [*Example:* The red color is 25% opaque: `<v:fill type="gradient" color="red"`

Attributes	Description
	color2="blue" opacity=".25"> </v:fill>  opacity="1"  opacity=".25" *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
preferrelative (Relative Resize Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the original size of an object is saved after reformatting. Default is `false`. If `true`, the original size of the object is stored and all resizing is based on a percentage of that original size. Otherwise, each resizing will reset the scale to 100%. [*Example:* <v:shape ... o:preferrelative="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
print (Print Toggle)	Specifies whether the shape will be printed. Default is `true`. [*Example:* <v:shape ... print="false" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
regroupid (Regroup ID) Namespace: urn:schemas-microsoft-com:office:office	Specifies a previous group for a shape. An ID number is used to identify groups of shapes that are no longer grouped. This allows shapes to be regrouped programmatically. [*Example:* The shape was part of a group identified by the ID 040754: <v:shape ... o:regroupid="040754" ... > </v:shape>

Attributes	Description
	end example] The possible values for this attribute are defined by the XML Schema integer datatype.
spid (Optional String) Namespace: urn:schemas-microsoft-com:office:office	Specifies an optional string that an application may use to Identify the particular shape. Default is no value. The possible values for this attribute are defined by the XML Schema string datatype.
spt (Optional Number) Namespace: urn:schemas-microsoft-com:office:office	Specifies an optional number that an application may use to associate the particular shape with a defined shape type. Default is 0. The possible values for this attribute are defined by the XML Schema float datatype.

strokecolor (Shape Stroke Color)	Specifies the primary color of the brush to use to stroke the path of the shape. Default is `black`. The color attribute of the stroke element (§6.1.2.21) overrides this. Colors are typically specified as either a named color, such as `red`, or six hexadecimal digits representing the red, green and blue values of the color, such as `#00FF30`. Full details are specified in the simple type description.
	
	[*Example:*
	
	`<v:shape ... strokecolor="red" ...>`
	`</v:shape>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).

stroked (Shape Stroke Toggle)	Specifies whether the path defining the shape is stroked with a solid line. The stroke element (§6.1.2.21) defines other strokes. The on attribute of the stroke element overrides this attribute. Default is `true`.
	
	[*Example:*
	
	`<v:shape ... fillcolor="red"`
	`  stroked="false" strokecolor="blue"...>`
	`</v:shape>`

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td> 

end example]

The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).</td></tr>
<tr><td>strokeweight (Shape Stroke Weight)</td><td>Specifies the width of the brush to use to stroke the path. Default is 1 point. If a number is given without units, the emu is used. The weight attribute of the stroke element (§6.1.2.21) overrides this attribute.

[*Example:*

<pre><v:shape ... strokeweight="3pt" ... >
</v:shape></pre>
 

end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>style (Shape Styling Properties)</td><td>Specifies the CSS2 styling properties of the shape. This uses the syntax described in the "Visual formatting model" of the Cascading Style Sheets, Level 2 specification, a Recommendation of the World Wide Web Consortium available here: http://www.w3.org/TR/REC-CSS2. Full descriptions of each property are not repeated here, but the VML treatment of each property is defined. Allowed properties include:

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>flip</td><td>Specifies that the orientation of a shape is flipped. Default is no value. Allowed values are:

• x - Flip along the y-axis, reversing the x-coordinates.
• y - Flip along the x-axis, reversing the y-coordinates.
• xy - Flip along both the y- and x-axis.
• yx - Flip along both the x- and y-axis.</td></tr>
<tr><td>height</td><td>Specifies the height of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are:

• auto - Default position of an element in the flow of the</td></tr>
</table>
</td></tr>
</table>

Attributes	Description
	page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	`left` — Specifies the position of the left of the containing block of the shape relative to the element left of it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	`margin-bottom` — Specifies the position of the bottom of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	`margin-left` — Specifies the position of the left of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.

Attributes	Description	
	margin-right	Specifies the position of the right of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	margin-top	Specifies the position of the top of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	mso-position-horizontal	Specifies the horizontal positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute • left • center • right • inside • outside
	mso-position-horizontal-relative	Specifies relative horizontal position data for objects in WordprocessingML documents. This modifies the mso-position-horizontal property. Default is text. Allowed values are: • margin • page • text • char

Attributes	Description	
	mso-position-vertical	Specifies the vertical positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute • top • center • bottom • inside • outside
	mso-position-vertical-relative	Specifies relative vertical position data for objects in WordprocessingML documents. This modifies the mso-position-vertical property. Default is text. Allowed values are: • margin • page • text • line
	mso-wrap-distance-bottom	Specifies the distance from the bottom of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
	mso-wrap-distance-left	Specifies the distance from the left side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
	mso-wrap-distance-right	Specifies the distance from the right side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
	mso-wrap-distance-top	Specifies the distance from the top of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
	mso-wrap-edited	Specifies whether the wrap coordinates were customized by the user. If the wrap coordinates are generated by an editor, this property is true; otherwise they were customized by a user.

Attributes	Description
	Default is `false`.
`mso-wrap-style`	Specifies the wrapping mode for text in shapes in WordprocessingML documents. Default is `square`. Allowed values are: • `square` - Wraps text inside the shape in a square. • `none` - Text does not wrap.
`position`	Specifies the type of positioning used to place an element. Default is `static`. When the element is contained inside a group, this property must be `absolute`. Allowed values are: • `static` - The element is positioned according to the normal flow of the page. The `top` and `left` properties are ignored. If the object is anchored inline, this value is used. • `absolute` - The element is positioned relative to the parent, using the `top` and `left` properties. • `relative` - The element is positioned according to the normal flow of the page, but the `top` and `left` properties are used. The overlap of overlapping elements is governed by the `z-index` property.
`rotation`	Specifies the angle that a shape is rotated, in degrees. Default is 0. Positive angles are clockwise.
`top`	Specifies the position of the top of the containing block of the shape relative to the element above it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
`visibility`	Specifies whether a shape is displayed. Only `inherit` and `hidden` are used; any other values are mapped to `inherit`. Default is `inherit`. Allowed values are: • `hidden` - The shape is not visible, but is still part of the flow of the objects in the browser. Mouse events are not processed.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>• inherit - The visibility state is inherited from the parent of the shape.</td></tr>
<tr><td>width</td><td>Specifies the width of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are:

• auto - Default position of an element in the flow of the page.
• <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed.
• <percentage>- Value expressed as a percentage of the parent object's width.</td></tr>
<tr><td>z-index</td><td>Specifies the display order of overlapping shapes. Default is 0. This property shall not be used for shapes anchored inline. Allowed values are:

• auto - Uses the order that the shapes appear in the page, bottom to top.
• <order>- A number that represents the stacking precedence. Shapes with higher numbers are placed on top of those with lower numbers. Negative numbers are allowed.</td></tr>
</table>

The following properties are only used by the textbox element (§6.1.2.22):

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>direction</td><td>Specifies the direction of the text in the textbox. Default is ltr. This property is superceded by the mso-direction-alt property if that is specified. Allowed values are:

• ltr - Test is displayed left-to-right.
• rtl - Test is displayed right-to-left.</td></tr>
<tr><td>layout-flow</td><td>Determines the flow of the text layout in a textbox. Default is horizontal. Allowed values are:

• horizontal - Text is displayed horizontally.
• vertical - Text is displayed vertically.
• vertical-ideographic - Ideographic text is displayed vertically.
• horizontal-ideographic - Ideographic text is displayed horizontally.</td></tr>
</table>
</td></tr>
</table>

Attributes	Description
mso-direction-alt	Specifies an alternate direction for text in textboxes. Overrides the direction property. The only allowed value is context.
mso-fit-shape-to-text	Specifies whether the shape stretches to fit the text in the textbox. Default is false.
mso-fit-text-to-shape	Specifies whether the text stretches to fit the textbox. Default is false.
mso-layout-flow-alt	Specifies the alternate layout flow for text in textboxes. This property is used instead of layout-flow when the layout flow is from bottom to top for non-ideographic languages. Its only value is bottom-to-top.
mso-next-textbox	Specifies the ID of the next textbox in a series. Used to keep track of a set of linked textboxes. Default is no value.
mso-rotate	Specifies a specific rotation value for text in a textbox. Default is 0. Allowed values are: • 0 • 90 • 180 • -90
mso-text-scale	Specifies the scaling factor for fitting text to shapes. Default is 0. This property is only used if mso-fit-text-to-shape is true.
v-text-anchor	Specifies the vertical anchoring of text in a textbox. Default is top. The alignment of a text anchor only becomes evident if mso-fit-text-to-shape is false. This property is different from the vertical-align CSS property, which is used for ideographic languages. Allowed values are: • top • middle • bottom • top-center • middle-center • bottom-center • top-baseline • bottom-baseline • top-center-baseline • bottom-center-baseline

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>The following properties are only used by the textpath element (§6.1.2.23):
<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>font</td><td>Specifies a compound value of font settings. Default is no value. The values are the same as those of the CSS font property. The order of definitions in the string is: font-style, font-variant, font-weight, font-size, line-height, font-family.</td></tr>
<tr><td>font-family</td><td>Specifies the family of the font. Default is no value. The values are the same as those of the CSS font-family property.</td></tr>
<tr><td>font-size</td><td>Specifies the size of the font. Default is no value. The font size is defined in points. The values are the same as those of the CSS font-size property.</td></tr>
<tr><td>font-style</td><td>Specifies the amount of slant for a font. Default is normal. The values are the same as those of the CSS font-style property. Allowed values are:

• normal
• italic
• oblique - Treated the same as italic.</td></tr>
<tr><td>font-variant</td><td>Specifies the variant style of a font. Default is normal. The values are the same as those of the CSS font-variant property. Allowed values are:

• normal
• small-caps</td></tr>
<tr><td>font-weight</td><td>Specifies the thickness of the letters of the font. Default is normal. The values are the same as those of the CSS font-weight property. Allowed values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>normal

lighter

100

200

300

400</td><td>Treated as non-bold.</td></tr>
<tr><td>bold</td><td>Treated as bold.</td></tr>
</table></td></tr>
</table></td></tr>
</table>

Attributes	Description
	bolder 500 600 700 800 900
mso-text-shadow	Specifies whether a shadow is applied to the text on a text path. Default is false.
text-decoration	Specifies the style of text decoration. Default is none. The values are the same as those of the CSS text-decoration property. Allowed values are: • none • underline • overline • line-through • blink
v-rotate-letters	Specifies whether the letters of the text are rotated counterclockwise by 90 degrees. Default is false.
v-same-letter-heights	Specifies whether all letters will be the same height regardless of initial case. If true, the lowercase letters are stretched to the height of the uppercase letters. Default is false.
v-text-align	Specifies the alignment of text. Default is left. Allowed values are: • left • right • center • justify • letter-justify - Distributes the extra space between the letters. • stretch-justify - Stretches the letters to fill in the space.
v-text-kern	Specifies whether kerning is turned on. Default is false.
v-text-reverse	Specifies whether the layout order of rows is reversed. Default is false. This is used for vertical text layout.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>v-text-spacing-mode</td><td>Specifies the mode for letter spacing. Default is tightening. This property determines whether space will be removed between each letter (tightening) or added between each letter (tracking). The amount of letter spacing change is defined by the v-text-spacing property. Allowed values are:

• tightening
• tracking</td></tr>
<tr><td>v-text-spacing</td><td>Specifies the amount of spacing for text in 100ths of single line spacing. Default is 100.</td></tr>
</table>

The line (§6.1.2.12), polyline (§6.1.2.15) and curve (§6.1.2.3) elements ignore the following properties:

• top
• left
• width
• height

The following properties are not inherited by an element that references a shapetype element (§6.1.2.20) via the id attribute:

• flip
• height
• left
• margin-left
• margin-top
• position
• rotation
• top
• visibility
• width
• z-index

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>target (Hyperlink Display Target)</td><td>Specifies a frame or window that a URL is displayed in. Default is no value. Allowed values are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td><targetname></td><td>String containing the name of the frame or window in which to load the document.</td></tr>
<tr><td>_blank</td><td>Specifies that the linked document is loaded into a new blank window. This window is not named.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>_media</td><td>Specifies that the linked document is loaded into the Media Bar. Available in Microsoft® Internet Explorer 6 or later.</td></tr>
<tr><td>_parent</td><td>Specifies that the linked document is loaded into the immediate parent of the document containing the link.</td></tr>
<tr><td>_search</td><td>Specifies that the linked document is loaded into the browser's search pane. Available in Microsoft Internet Explorer 5 or greater.</td></tr>
<tr><td>_self</td><td>Specifies that the linked document is loaded into the window in which the link was clicked (the active window).</td></tr>
<tr><td>_top</td><td>Specifies that the linked document is loaded into the topmost window.</td></tr>
</table>
[Example:
<pre>
<v:shape ...
  href="http://www.openxmlformats.org"
  target="_self" ... >
</v:shape>
</pre>
end example]
The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>title (Shape Title)</td><td>
Specifies the text displayed when the mouse pointer moves over the shape. Default is no value.
[Example:
<pre>
<v:shape ... title="tooltip" ... >
</v:shape>
</pre>
end example]
The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>userdrawn (Exists In Master Slide)

Namespace: urn:schemas-microsoft-com:office:office</td><td>
Specifies whether the user has added the shape to a master slide. Default is false. Used by PresentationML.
[Example:
<pre>
<v:shape ... o:userdrawn="true" ... >
</v:shape>
</pre>
end example]
The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
</td></tr>
</table>

Attributes	Description
userhidden (Hide Script Anchors) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a script anchor is hidden. Default is `false`. If `true`, script anchors stay hidden even if the shape is otherwise visible. A script anchor is the visual representation of a script that when displayed in an application. [*Example:* `<v:shape ... o:userhidden="true" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
wrapcoords (Shape Bounding Polygon)	Specifies the bounding polygon that surrounds a shape. This is specified using a comma-delimited list of x and y coordinates: "x1,y1,x2,y2,x3,y3,..." This is used when text is tightly wrapped around a shape. Default is no value until the mso-wrap-mode style attribute is set to `tight` or `through`. [*Example:* `<v:shape ... wrapcoords="0,0 0,200, 200,200, 200,0" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Oval">
   <choice maxOccurs="unbounded">
      <group ref="EG_ShapeElements" minOccurs="0" maxOccurs="unbounded"/>
   </choice>
   <attributeGroup ref="AG_AllCoreAttributes"/>
   <attributeGroup ref="AG_AllShapeAttributes"/>
</complexType>
```

6.1.2.14 path (Shape Path)

This element defines the path that makes up the shape. This is done through a string that contains a rich set of pen movement commands. This element also describes the limo-stretch point, inscribed textbox rectangle locations and connection site locations. The limo-stretch definition and the formulas element (§6.1.2.6) allow greater designer control of how the path scales. They allow, for example, definition of a true rounded corner rectangle where the corners remain circular even though the rectangle is scaled anisotropically.

Parent Elements

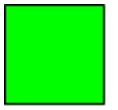
Parent Elements
arc (§6.1.2.1); background (§2.2.1); curve (§6.1.2.3); group (§6.1.2.7); image (§6.1.2.10); line (§6.1.2.12); object (§2.3.3.19); oval (§6.1.2.13); pict (§2.3.3.21); pict (§2.9.23); polyline (§6.1.2.15); rect (§6.1.2.16); roundrect (§6.1.2.17); shape (§6.1.2.19); shapetype (§6.1.2.20)

Attributes	Description

arrowok (Arrowhead Display Toggle)	Specifies whether arrowheads are allowed to be displayed. This attribute overrides all other arrowhead attributes in the parent or the stroke element (§6.1.2.21). Default is `false`.
	
	[*Example:*
	
	`<v:shape style="width:50;height:50">`
	`  <v:stroke endarrow="block"/>`
	`  <v:path arrowok="true"`
	`  v="m 0,0 l 1000,0 1000,1000 e"/>`
	`</v:shape>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).

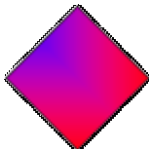
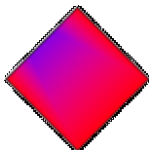
connectangles (Connection Point Connect Angles)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies the angle at which curves connect to a shape's connection points. The connection angles are defined by a string consisting of angle values delimited by commas. Default is no value.
	
	[*Example:* Connections are made along the horizontal and vertical axes:
	
	`<v:path ... o:connectangles="0,90,180,270" ... >`
	`</v:path>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

connectlocs (Connection Points) Namespace: urn:schemas-microsoft-com:office:office	Specifies the location of connection points on a path. The connection points are defined by a string consisting of pairs of x and y values, delimited by commas. This is used if connecttype is `custom`. Default is no value. [*Example:* Connection points exist at the midpoints of the sides of the square: `<v:path ... v="m 0,0 l 100,0 100,100 0,100 x e"`

Attributes	Description
	o:connectlocs="50,0;100,50;50,100;0,50" ... > </v:path> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
connecttype (Connection Point Type) Namespace: urn:schemas-microsoft-com:office:office	Specifies the type of connection points used for attaching shapes to other shapes. Default is none. If set to custom, connectlocs is used. Allowed values are: [*Example:* <v:path ... o:connecttype="custom" o:connectlocs="50,0;100,50;50,100;0,50" ... > </v:path> *end example*] The possible values for this attribute are defined by the ST_ConnectType simple type (§6.2.3.7).
extrusionok (Extrusion Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether an extrusion is allowed to be displayed. This attribute overrides all other extrusion attributes in the parent or the extrusion element (§6.2.2.10). Default is true. [*Example:* <v:rect fillcolor="lime" style="width:50;height:50"> <v:extrusion on="true"/> <v:path o:extrusionok="false"/> </v:rect>  <v:path o:extrusionok="false"/>  <v:path o:extrusionok="true"/> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
fillok (Shape Fill Toggle)	Specifies whether a fill is allowed to be displayed. This attribute overrides all other fill attributes in the parent or fill element (§6.1.2.5). Default is true.

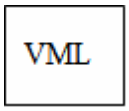
Attributes	Description

	[*Example:*
	
	`<v:shape style="width:50;height:50"`
	`  fillcolor="red">`
	`  <v:path fillok="false"`
	`  v="m 0,0 l 0,1000, 1000,1000, 1000,0 x e"/>`
	`</v:shape>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).

gradientshapeok (Gradient Shape Toggle)	Specifies whether a gradient path will be made up of repeated concentric paths. Default is `false`.
	
	If `true`, a gradient fill can be produced by repeated drawing of scaled versions of the path - this must only be set if it is possible to scale the path in such a way that a fill is always contained in the original path. This controls the interpretation of the type="gradientradial" attribute of the fill element (§6.1.2.5).
	
	[*Example:* In the first case, the radial gradient is aligned irrespective of the shape's path:
	
	`<v:shape style="width:50;height:50;rotation:45"`
	`  path="m 0,0 l 0,1000, 1000,1000, 1000,0 x e">`
	`  <v:path gradientshapeok="false"/>`
	`  <v:fill type="gradientradial"`
	`  color="red" color2="blue"/>`
	`</v:shape>`
	
	 
	`gradientshapeok="false"`
	 
	`gradientshapeok="true"`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type

Attributes	Description
	(§6.1.3.14).
id (Unique Identifier)	Specifies a unique identifier that can be used to reference a VML object. Default is no value. [*Example:* `<v:shape ... id="myShape" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
insetpenok (Inset Stroke From Path Flag)	Specifies whether the stroke may be inset from the path. If this is `false`, it overrides the insetpen attribute and prevents the stroke from being inset. [*Example:* The stroke is not inset: `<v:shape ... insetpen="true"> <v:path ... insetpenok="false"/> </v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
limo (Limo Stretch Point)	Specifies a stretch point on the shape's edge that defines where and how a shape is allowed to be stretched by a user in a graphical editor. Default is "0,0". [*Example:* `<v:line from="20pt,20pt" to="100pt,20pt"> <v:path limo="60pt,20pt"/> </v:line>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
shadowok (Shadow Toggle)	Specifies whether a shadow is allowed to be displayed. This attribute overrides all other shadow attributes in the parent or the shadow element (§6.1.2.18). Default is `true`. [*Example:* The shape has no shadow: `<v:shape style="width:50;height:50"> <v:path v="m 0,0 l 0,1000, 1000,1000, 1000,0 x e"`

Attributes	Description
	shadowok="false"/> <v:shadow on="true"/> </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
strokeok (Stroke Toggle)	Specifies whether a stroke will be displayed. This attribute overrides all other stroke attributes in the parent or the stroke element (§6.1.2.21). Default is true. [*Example:* The shape's red stroke is not shown: <v:shape style="width:50;height:50" fillcolor="blue" strokecolor="red"> <v:path v="m 0,0 l 0,1000, 1000,1000, 1000,0 x e" strokeok="false"/> </v:shape>  *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
textboxrect (Text Box Bounding Box)	Specifies one or more text boxes inside a shape. Default is the same as the geometry's bounding box. A textbox is defined by one or more sets of numbers specifying (in order) the left, top, right, and bottom points of the rectangle. Multiple sets are delimited by a semicolon. The default value is the same dimension value as the containing rectangle. If more than one textbox is defined, the comma-delimited quadruple sets that define each textbox are separated by semicolons. Normally textboxes come in sets of 1, 2, 3, or 6 rectangles on a shape. The textboxrect dimensions clip any text that extends beyond its region. [*Example:* The textbox is 25% down from the top and the exclamation point is clipped: <v:shape style="width:60;height:50"> <v:path v="m 0,0 l 0,1000, 1000,1000, 1000,0 x e" textboxrect="0,250,850,1000"/> <v:textbox>VML!</v:textbox> </v:shape>

Attributes	Description
	 *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
textpathok (Text Path Toggle)	Specifies whether a text path will be displayed. Default is `false`. If `true`, this indicates that the path is an appropriate warping path for the textpath element (§6.1.2.23). Otherwise, the textpath element must be ignored. [*Example:* The defined textpath is ignored: `<v:curve from="50,100" to="400,100" control1="200,200" control2="300,200"><v:path textpathok="false"/><v:textpath on="false" style="font:normal normal normal 36pt Arial" string="textpath"/></v:curve>`  *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
v (Path Definition)	Specifies a string containing the commands that define the shape's path. This value consists of commands followed by zero or more parameters. Default is no value. The following rules apply to path strings: • Commas or spaces delimit parameters for each command. Both "m 0,0" and "m0 0" are acceptable. • A parameter that is omitted using commas is treated as having a value of zero. Thus, "c 10,10,0,0,25,13" and "c 10,10,,,25,13" are equivalent. • Parameterized paths are also allowed. In this case, the shape must also have a formulas element (§6.1.2.6) with a list of formulas that are substituted into the path using the @ symbol followed by the number of the formula. The adj property of the shape contains the input parameters for these formulas. For example, "moveto @1@4". The evaluations of the formulas are substituted into the appropriate positions. Note that @ also serves as a delimiter.

Attributes	Description

The allowed commands are given below. An asterisk (*) indicates that the command is allowed to be repeated. For the qb command, the controlpoint parameter is also allowed to be repeated.

Command	Name	Parameters	Description
m	moveto	2	Start a new sub-path at the given (x,y) coordinate.
l	lineto	2*	Draw a line from the current point to the given (x,y) coordinate which becomes the new current point. Specifying a number of coordinate pairs forms a polyline.
c	curveto	6*	Draw a cubic bézier curve from the current point to the coordinate given by the final two parameters. The control points are given by the first four parameters.
x	close	0	Close the current sub-path by drawing a straight line from the current point to the original moveto point.
e	end	0	End the current set of sub-paths. A given set of sub-paths (as delimited by end) is filled. Subsequent sets of sub-paths are filled independently and superimposed on existing ones.
t	rmoveto	2*	Start a new sub-path at a coordinate relative to the current point, cp (cpx+x, cpy+y).
r	rlineto	2*	Draw a line from the current point to the given relative coordinate (cpx+x, cpy+y).
v	rcurveto	6*	Cubic bézier curve using the given coordinate relative to the current point.
nf	nofill	0	The current set of sub-paths (delimited by e) will not be filled.
ns	nostroke	0	The current set of sub-paths (delimited by e) will not be stroked.

Attributes	Description			
	ae	angleellipseto	6*	Draws a segment of an ellipse as described using these parameters. A straight line is drawn from the current point to the start point of the segment. The parameters are: center (x,y), size(w,h), start angle, end angle.
	al	angleellipse	6*	Same as angleellipseto except that there is an implied moveto the starting point of the segment.
	at	arcto	8*	A segment of the ellipse is drawn which starts at the angle defined by the start radius vector and ends at the angle defined by the end vector. A straight line is drawn from the current point to the start of the arc. The arc is always drawn in a counterclockwise direction. The parameters are: left, top, right, bottom, start(x,y), end(x,y). The first four values define the bounding box of an ellipse. The last four define two radial vectors.
	ar	arc	8*	Same as arcto except there is an implied moveto the start point of the arc.
	wa	clockwisearcto	8*	Same as arcto but the arc is drawn in a clockwise direction.
	wr	clockwisearc	8*	Same as arc but the arc is drawn in a clockwise direction
	qx	ellipticalqaudrantx	2*	A quarter ellipse is drawn from the current point to the given end point. The elliptical segment is initially tangential to a line parallel to the x-axis. (i.e. the segment starts out horizontal). The parameters are: end(x,y).
	qy	ellipticalquadranty	2*	Same as ellipticalquadrantx except that the elliptical segment is initially tangential to a line parallel to the y-axis (i.e. the segment starts out vertical).

Attributes	Description
	qb | quadraticbezier | 2+2* | Defines one or more quadratic bézier curves by means of control points and an end point. Intermediate (on-curve) points are obtained by interpolation between successive control points as in the OpenType font specification. The sub-path need not be started in which case the sub-path will be closed. In this case the last point of the sub-path defines the start point of the quadratic bézier. The parameters are: controlpoint(x,y)*, end(x,y). The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

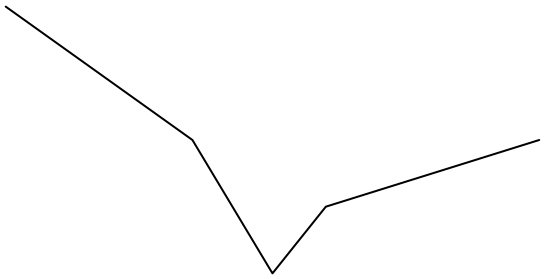
```
<complexType name="CT_Path">
   <attributeGroup ref="AG_Id"/>
   <attribute name="v" type="xsd:string" use="optional"/>
   <attribute name="limo" type="xsd:string" use="optional"/>
   <attribute name="textboxrect" type="xsd:string" use="optional"/>
   <attribute name="fillok" type="ST_TrueFalse" use="optional"/>
   <attribute name="strokeok" type="ST_TrueFalse" use="optional"/>
   <attribute name="shadowok" type="ST_TrueFalse" use="optional"/>
   <attribute name="arrowok" type="ST_TrueFalse" use="optional"/>
   <attribute name="gradientshapeok" type="ST_TrueFalse" use="optional"/>
   <attribute name="textpathok" type="ST_TrueFalse" use="optional"/>
   <attribute name="insetpenok" type="ST_TrueFalse" use="optional"/>
   <attribute ref="o:connecttype"/>
   <attribute ref="o:connectlocs"/>
   <attribute ref="o:connectangles"/>
   <attribute ref="o:extrusionok"/>
</complexType>
```

6.1.2.15 polyline (Multiple Path Line)

This element defines shapes made up of connected line segments.

[*Example:*

```
<v:polyline
  points="50pt,0pt 120pt,50pt 150pt,100pt 170pt,75pt 250pt,50pt">
</v:polyline>
```



end example]

Parent Elements
background (§2.2.1); group (§6.1.2.7); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23)

Child Elements	Subclause
anchorlock (Anchor Location Is Locked)	§6.3.2.1
borderbottom (Bottom Border)	§6.3.2.2
borderleft (Left Border)	§6.3.2.3
borderright (Right Border)	§6.3.2.4
bordertop (Top Border)	§6.3.2.5
callout (Callout)	§6.2.2.2
ClientData (Attached Object Data)	§6.4.2.12
clippath (Shape Clipping Path)	§6.2.2.3
extrusion (3D Extrusion)	§6.2.2.10
fill (Shape Fill Properties)	§6.1.2.5
formulas (Set of Formulas)	§6.1.2.6
handles (Set of Handles)	§6.1.2.9
imagedata (Image Data)	§6.1.2.11
lock (Shape Protections)	§6.2.2.17
path (Shape Path)	§6.1.2.14
shadow (Shadow Effect)	§6.1.2.18
signatureline (Digital Signature Line)	§6.2.2.29
skew (Skew Transform)	§6.2.2.30
stroke (Line Stroke Settings)	§6.1.2.21
textbox (Text Box)	§6.1.2.22
textdata (VML Diagram Text)	§6.5.2.2
textpath (Text Layout Path)	§6.1.2.23
wrap (Text Wrapping)	§6.3.2.6

Attributes	Description
allowincell (Allow in Table Cell) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape can be placed in a table. Default is `false`. [*Example:* `<v:shape ... o:allowincell="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
allowoverlap (Allow Shape Overlap) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape can overlap another shape. Default is `true`. If `false`, the shape will shift left or right so as not to overlap another shape, similar to the behavior of the HTML float attribute. [*Example:* `<v:shape ... o:allowoverlap="false" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

alt (Alternate Text)	Specifies alternative text describing the graphical object. This text should provide a brief description of the shape for use by accessibility tools. Default is no value.
	
	[*Example:* The alt text describes the basic shape:
	
	`<v:shape ... fillcolor="red" alt="Red rectangle"></v:shape>`
	
	The alt text describes the contents of a shape displaying an image:
	
	`<v:shape ... alt="Picture of a sunset"></v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

borderbottomcolor (Bottom Border	Specifies the bottom border color of an inline shape. Default is no value.

Attributes	Description
Color) Namespace: urn:schemas-microsoft-com:office:office	[*Example:* `<v:shape ... o:borderbottomcolor="red" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
borderleftcolor (Border Left Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the left border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:borderleftcolor="red" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
borderrightcolor (Border Right Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the right border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:borderrightcolor="red" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
bordertopcolor (Border Top Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the top border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:bordertopcolor="red" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
bullet (Graphical Bullet) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the shape is a graphical bullet. Default is `false`. [*Example:* `<v:shape ... o:bullet="true" ... ></v:shape>`

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
button (Button Behavior Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape will exhibit button press behavior on click. Default is `false`. [*Example:* `<v:shape ... o:button="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
bwmode (Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies how a shape will render for black-and-white output devices. When a shape is printed on a black-and-white printer or displayed in a black-and-white view in an application, several options are possible. Default is `auto`, which will use o:bwnormal for normal black-and-white rendering and o:bwpure for pure black-and-white rendering. bwnormal and bwpure are subordinate to bwmode. If bwmode is "auto" then the value for bwnormal or bwpure is used depending on what the output format is. An application may define for itself what, if any, difference there is between normal B&W and pure B&W. For example, normal B&W might allow greyscale and pure B&W might not. [*Example:* This shape renders in grayscale in a black-and-white environment: `<v:shape ... o:bwmode="grayscale" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
bwnormal (Normal Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies the black-and-white mode for normal black-and-white output devices. Default is `auto`. [*Example:* This shape renders in a pale grayscale in a normal black-and-white environment: `<v:shape ... o:bwmode="auto" o:bwnormal="lightgrayscale" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type

Attributes	Description
	(§6.2.3.2).
bwpure (Pure Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies the black-and-white mode for pure black-and-white output devices. Default is auto. [*Example:* This shape renders in high contrast when in a pure black-and-white environment: `<v:shape ... o:bwmode="auto" o:bwpure="highcontrast" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
chromakey (Image Transparency Color)	Specifies a color value that will be transparent and show anything behind the shape. Default is no value. [*Example:* `<v:image ... chromakey="white" ...> </v:image>` *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).
class (CSS Reference)	Specifies a reference to the definition of a CSS style. Default is no value. [*Example:* The snippets below are equivalent: `... .narrowstyle {width:50;height:100} ... <v:shape ... class="narrowstyle" style="top:1;left:1"> </v:shape>` `<v:shape ... style="top:1;left:1; width:50;height:100"> </v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

Attributes	Description

clip (Clipping Toggle)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies that the clipping region is active. This is used in conjunction with the clippath (§6.2.2.3) element to create a clipping region.
	
	[*Example:*
	
	`<v:shape ... o:clip="true">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

cliptowrap (Clip to Wrapping Polygon)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies that the clipping region for the shape aligns with the wrapping polygon that tightly surrounds the entire shape (essentially, that the shape shall not be drawn beyond its wrapping polygon's extents – if it does, the shape shall be clipped). Default is `false`.
	
	[*Example:*
	
	`<v:shape ... o:cliptowrap="true">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

connectortype (Shape Connector Type)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies the type of connector used for joining shapes. Default is `straight`.
	
	[*Example:*
	
	`<v:shape ... o:connectortype="elbow" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_ConnectorType simple type (§6.2.3.6).

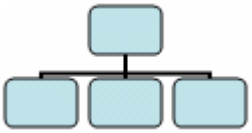
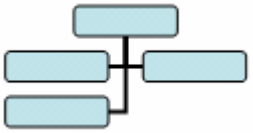
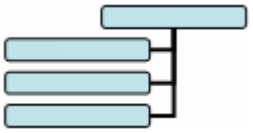
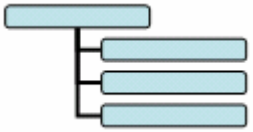
coordorigin (Coordinate Space Origin)	Specifies the coordinate of the top left corner of the shape's coordinate space. This determines the position of the (0,0) coordinate space origin within the shape's bounding box. Default is "0,0", which places the (0,0) origin at the top left corner of the bounding box. This affects shape properties that specify coordinate positions, such as the path attribute. Thus a path can be defined against a generic (0,0) origin and the coordorigin value translates the entire path within the shape's bounding space.

Attributes	Description

	[*Example:* The horizontal and vertical coordinate space ranges from -100 to +100 because the coordinate space (coordsize) is 200 by 200 and the top left coordinate is (-100,-100). The (0,0) origin lies at the center of the shape's bounding box, as evidenced by the position of the shape's path within the coordinate space:
	
	`<v:shape ... coordsize="200,200"`
	`  coordorigin="-100,-100"`
	`  path="m 0,0 l 0,50, 50,50, 50,0 x e">`
	`</v:shape>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

coordsize (Coordinate Space Size)	Specifies the size of the shape's coordinate space in coordinate units. Default is "1000,1000".
	
	The physical size of a coordinate unit length is determined by both the size of the coordinate space (coordsize) and the size of the shape (style width and height). The coordsize attribute defines the number of horizontal and vertical subdivisions into which the shape's bounding box is divided. The combination of coordsize and style width/height effective scales the shape anisotropically.
	
	[*Example:* The path is 50 units wide and tall, which is 25% of the size of the coordinate space:
	
	`<v:shape ... coordsize="200,200"`
	`  coordorigin="-100,-100"`
	`  path="m 0,0 l 0,50, 50,50, 50,0 x e">`
	`</v:shape>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

dgmlayout (Diagram Node Layout Identifier) Namespace:	Specifies the type of automatic layout to apply to the diagram node. This is only meaningful for organization charts. Valid values for organization charts are: **Value** | **Description**

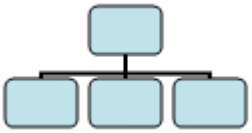
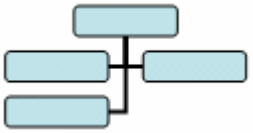
<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>urn:schemas-microsoft-com:office:office</td><td>
<table>
<tr><td>0</td><td>Top-down, centered layout.
  </td></tr>
<tr><td>1</td><td>Hanging, both sides layout.
  </td></tr>
<tr><td>2</td><td>Hanging, right side layout.
  </td></tr>
<tr><td>3</td><td>Hanging, left side layout.
  </td></tr>
</table>

[Example:

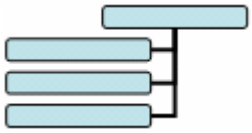
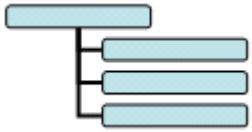
<pre><v:shape ... dgmlayout="1">
</v:shape></pre>
end example]

The possible values for this attribute are defined by the XML Schema integer datatype.</td></tr>
<tr><td>dgmlayoutmru (Diagram Node Recent Layout Identifier)

Namespace: urn:schemas-microsoft-com:office:office</td><td>Specifies the type of automatic layout most recently used on the child elements of the diagram node. This is only meaningful for organization charts. Valid values for organization charts are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Top-down, centered layout.
  </td></tr>
<tr><td>1</td><td>Hanging, both sides layout.
  </td></tr>
<tr><td>2</td><td>Hanging, right side layout.</td></tr>
</table>
</td></tr>
</table>

Attributes	Description

	<table><tr><td></td><td>  </td></tr><tr><td>3</td><td>Hanging, left side layout.  </td></tr></table>
	[*Example:*
	
	`<v:shape ... dgmlayout="1">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema integer datatype.

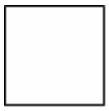
dgmnodekind (Diagram Node Identifier)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies an optional, application-specific parameter that is intended to be used by the application to tag different types of nodes in a diagram.
	
	[*Example:*
	
	`<v:shape ... dgmnodekind="1">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema integer datatype.

doubleclicknotify (Double-click Notification Toggle)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies that an event message is sent when a shape is double-clicked. Default is `false`.
	
	[*Example:*
	
	`<v:shape ... o:doubleclicknotify="true" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

fillcolor (Fill Color)	Specifies the color to use for the fill. Default is `white`. If the fill element (§6.1.2.5) is present, its color attribute takes precedence. Colors are typically specified as either a named color, such as `red`, or six hexadecimal digits representing the red, green and blue values of the color, such as `#00FF30`. Full details are specified in the simple type description.

Attributes	Description

	[*Example:* This shape is red if its fill is visible:
	
	`<v:shape ... fillcolor="red" ... ></v:shape>`
	
	This is equivalent to:
	
	`<v:shape ... fillcolor="#ff0000" ... ></v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).

filled (Shape Fill Toggle)	Specifies whether the closed path will be filled. Default is `true`. This attribute is overridden by the fill on attribute. [*Example:* `<v:shape ... filled="f" fillcolor="red" ...></v:shape>`  *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
forcedash (Force Dashed Outline) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a dashed outline is used to draw a shape when a shape has no line or fill. Default is `false`. Used by PresentationML placeholders to draw a dashed outline when there is no line and no fill for a shape. [*Example:* `<v:shape ... o:forcedash="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type

Attributes	Description
	(§6.2.3.23).
hr (Horizontal Rule Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies that a shape is a horizontal rule. Default is `false`. [*Example:* `<v:shape ... o:hr="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
hralign (Horizontal Rule Alignment) Namespace: urn:schemas-microsoft-com:office:office	Specifies the alignment of a horizontal rule. Default is `left`. [*Example:* `<v:shape ... o:hralign="center" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_HrAlign simple type (§6.2.3.14).
href (Hyperlink Target)	Specifies a hyperlink URL target for the shape. Default is no value. [*Example:* `<v:shape ... href="http://www.openxmlformats.org" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
hrnoshade (Horizontal Rule 3D Shading Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies that the horizontal rule does not have 3-D shading. Default is `false`. [*Example:* `<v:shape ... o:hrnoshade="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
hrpct (Horizontal	Specifies the length of a horizontal rule as a percentage of page width. Default is 0.

Attributes	Description
Rule Length Percentage) Namespace: urn:schemas-microsoft-com:office:office	[*Example:* `<v:shape ... o:hrpct="85" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema float datatype.
hrstd (Horizontal Rule Standard Display Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape is displayed as a standard horizontal rule. Only applies if hr is `true`. Default is `false`. [*Example:* `<v:shape ... o:hrstd="true" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
id (Unique Identifier)	Specifies a unique identifier that can be used to reference a VML object. Default is no value. [*Example:* `<v:shape ... id="myShape" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
insetmode (Text Inset Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the application calculates the internal text margin instead of using the inset attribute. Default is `custom`. This attribute is only meaningful for text boxes. [*Example:* `<v:shape ... o:insetmode="auto" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_InsetMode simple type (§6.2.3.15).

Attributes	Description
insetpen (Inset Border From Path)	Specifies that the border shall be displayed inside of the path defining the shape, rather than along the path (the default border placement), or outside of the path as might be done with an image. [*Example:* `<v:shape ... insetpen="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
ole (Embedded Object Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the shape is an embedded object. Default is `false`. [*Example:* `<v:shape ... o:ole="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalseBlank simple type (§6.2.3.24).
oleicon (Embedded Object Icon Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether an embedded object will be displayed as an icon. Default is `false`. [*Example:* `<v:shape ... o:oleicon="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
oned (Shape Handle Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the extra handles of a shape are hidden. If `true`, hides all shape handles except the top left and bottom right; that is, the same handles that are used for a straight line segment. Default is `false`. [*Example:* `<v:shape ... o:oned="true" ... ></v:shape>` *end example*]

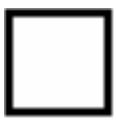
Attributes	Description
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

opacity (Fill Color Opacity)	Specifies the opacity of the primary fill color. Default is 1.0. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8.
	
	[*Example:* The red color is 25% opaque:
	
	`<v:fill type="gradient" color="red"`
	`  color2="blue" opacity=".25">`
	`</v:fill>`
	
	  `opacity="1"`
	  `opacity=".25"`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

points (Points for Compound Line)	Specifies a set of straight line segments that are composed of a series of pairs of points. Default is "0,0 10,10". Points are specified in the coordinate system of the parent element. If the parent is not a VML element, the default unit is a pixel. Allowed units are in, cm, mm, pt, pc and px. While commas are not required, they should be used for easier readability. See above for an example. The possible values for this attribute are defined by the XML Schema string datatype.

preferrelative (Relative Resize Toggle)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies whether the original size of an object is saved after reformatting. Default is `false`. If `true`, the original size of the object is stored and all resizing is based on a percentage of that original size. Otherwise, each resizing will reset the scale to 100%.
	
	[*Example:*
	
	`<v:shape ... o:preferrelative="true" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type

Attributes	Description
	(§6.2.3.23).
print (Print Toggle)	Specifies whether the shape will be printed. Default is `true`. [*Example:* `<v:shape ... print="false" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
regroupid (Regroup ID) Namespace: urn:schemas-microsoft-com:office:office	Specifies a previous group for a shape. An ID number is used to identify groups of shapes that are no longer grouped. This allows shapes to be regrouped programmatically. [*Example:* The shape was part of a group identified by the ID 040754: `<v:shape ... o:regroupid="040754" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema integer datatype.
spid (Optional String) Namespace: urn:schemas-microsoft-com:office:office	Specifies an optional string that an application may use to Identify the particular shape. Default is no value. The possible values for this attribute are defined by the XML Schema string datatype.
spt (Optional Number) Namespace: urn:schemas-microsoft-com:office:office	Specifies an optional number that an application may use to associate the particular shape with a defined shape type. Default is 0. The possible values for this attribute are defined by the XML Schema float datatype.
strokecolor (Shape Stroke Color)	Specifies the primary color of the brush to use to stroke the path of the shape. Default is `black`. The color attribute of the stroke element (§6.1.2.21) overrides this. Colors are typically specified as either a named color, such as `red`, or six hexadecimal digits representing the red, green and blue values of the color, such as `#00FF30`. Full details are specified in the simple type description. [*Example:*

Attributes	Description
	<v:shape ... strokecolor="red" ...> </v:shape>  *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).
stroked (Shape Stroke Toggle)	Specifies whether the path defining the shape is stroked with a solid line. The stroke element (§6.1.2.21) defines other strokes. The on attribute of the stroke element overrides this attribute. Default is true. [*Example:* <v:shape ... fillcolor="red" stroked="false" strokecolor="blue"...> </v:shape>  *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
strokeweight (Shape Stroke Weight)	Specifies the width of the brush to use to stroke the path. Default is 1 point. If a number is given without units, the emu is used. The weight attribute of the stroke element (§6.1.2.21) overrides this attribute. [*Example:* <v:shape ... strokeweight="3pt" ... > </v:shape>  *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
style (Shape Styling	Specifies the CSS2 styling properties of the shape. This uses the syntax described in the

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>Properties)</td><td>"Visual formatting model" of the Cascading Style Sheets, Level 2 specification, a Recommendation of the World Wide Web Consortium available here: http://www.w3.org/TR/REC-CSS2. Full descriptions of each property are not repeated here, but the VML treatment of each property is defined. Allowed properties include:
<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>flip</td><td>Specifies that the orientation of a shape is flipped. Default is no value. Allowed values are:
x - Flip along the y-axis, reversing the x-coordinates.
y - Flip along the x-axis, reversing the y-coordinates.
xy - Flip along both the y- and x-axis.
yx - Flip along both the x- and y-axis.</td></tr>
<tr><td>height</td><td>Specifies the height of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are:
auto - Default position of an element in the flow of the page.
<units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed.
<percentage>- Value expressed as a percentage of the parent object's height.</td></tr>
<tr><td>left</td><td>Specifies the position of the left of the containing block of the shape relative to the element left of it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are:
auto - Default position of an element in the flow of the page.
<units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed.
<percentage>- Value expressed as a percentage of the parent object's width.</td></tr>
<tr><td>margin-bottom</td><td>Specifies the position of the bottom of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are:
auto - Default position of an element in the flow of the page.</td></tr>
</table>
</td></tr>
</table>

Attributes	Description	
		• <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	margin-left	Specifies the position of the left of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	margin-right	Specifies the position of the right of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	margin-top	Specifies the position of the top of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	mso-position-horizontal	Specifies the horizontal positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are:

Attributes	Description	
		• absolute • left • center • right • inside • outside
	mso-position-horizontal-relative	Specifies relative horizontal position data for objects in WordprocessingML documents. This modifies the mso-position-horizontal property. Default is text. Allowed values are: • margin • page • text • char
	mso-position-vertical	Specifies the vertical positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute • top • center • bottom • inside • outside
	mso-position-vertical-relative	Specifies relative vertical position data for objects in WordprocessingML documents. This modifies the mso-position-vertical property. Default is text. Allowed values are: • margin • page • text • line
	mso-wrap-distance-bottom	Specifies the distance from the bottom of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
	mso-wrap-	Specifies the distance from the left side of the shape to the text

Attributes	Description	
	distance-left	that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
	mso-wrap-distance-right	Specifies the distance from the right side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
	mso-wrap-distance-top	Specifies the distance from the top of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
	mso-wrap-edited	Specifies whether the wrap coordinates were customized by the user. If the wrap coordinates are generated by an editor, this property is true; otherwise they were customized by a user. Default is false.
	mso-wrap-style	Specifies the wrapping mode for text in shapes in WordprocessingML documents. Default is square. Allowed values are: • square - Wraps text inside the shape in a square. • none - Text does not wrap.
	position	Specifies the type of positioning used to place an element. Default is static. When the element is contained inside a group, this property must be absolute. Allowed values are: • static - The element is positioned according to the normal flow of the page. The top and left properties are ignored. If the object is anchored inline, this value is used. • absolute - The element is positioned relative to the parent, using the top and left properties. • relative - The element is positioned according to the normal flow of the page, but the top and left properties are used. The overlap of overlapping elements is governed by the z-index property.
	rotation	Specifies the angle that a shape is rotated, in degrees. Default is 0. Positive angles are clockwise.
	top	Specifies the position of the top of the containing block of the shape relative to the element above it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group,

Attributes	Description
	in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
visibility	Specifies whether a shape is displayed. Only inherit and hidden are used; any other values are mapped to inherit. Default is inherit. Allowed values are: • hidden - The shape is not visible, but is still part of the flow of the objects in the browser. Mouse events are not processed. • inherit - The visibility state is inherited from the parent of the shape.
width	Specifies the width of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
z-index	Specifies the display order of overlapping shapes. Default is 0. This property shall not be used for shapes anchored inline. Allowed values are: • auto - Uses the order that the shapes appear in the page, bottom to top. • <order>- A number that represents the stacking precedence. Shapes with higher numbers are placed on top of those with lower numbers. Negative numbers are allowed.
The following properties are only used by the textbox element (§6.1.2.22):	

Attributes	Description

Property	Description
direction	Specifies the direction of the text in the textbox. Default is ltr. This property is superceded by the mso-direction-alt property if that is specified. Allowed values are: • ltr - Test is displayed left-to-right. • rtl - Test is displayed right-to-left.
layout-flow	Determines the flow of the text layout in a textbox. Default is horizontal. Allowed values are: • horizontal - Text is displayed horizontally. • vertical - Text is displayed vertically. • vertical-ideographic - Ideographic text is displayed vertically. • horizontal-ideographic - Ideographic text is displayed horizontally.
mso-direction-alt	Specifies an alternate direction for text in textboxes. Overrides the direction property. The only allowed value is context.
mso-fit-shape-to-text	Specifies whether the shape stretches to fit the text in the textbox. Default is false.
mso-fit-text-to-shape	Specifies whether the text stretches to fit the textbox. Default is false.
mso-layout-flow-alt	Specifies the alternate layout flow for text in textboxes. This property is used instead of layout-flow when the layout flow is from bottom to top for non-ideographic languages. Its only value is bottom-to-top.
mso-next-textbox	Specifies the ID of the next textbox in a series. Used to keep track of a set of linked textboxes. Default is no value.
mso-rotate	Specifies a specific rotation value for text in a textbox. Default is 0. Allowed values are: • 0 • 90 • 180 • -90
mso-text-scale	Specifies the scaling factor for fitting text to shapes. Default is 0. This property is only used if mso-fit-text-to-shape is true.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>v-text-anchor</td><td>Specifies the vertical anchoring of text in a textbox. Default is top. The alignment of a text anchor only becomes evident if mso-fit-text-to-shape is false. This property is different from the vertical-align CSS property, which is used for ideographic languages. Allowed values are:
top
middle
bottom
top-center
middle-center
bottom-center
top-baseline
bottom-baseline
top-center-baseline
bottom-center-baseline</td></tr>
</table>
The following properties are only used by the textpath element (§6.1.2.23):
<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>font</td><td>Specifies a compound value of font settings. Default is no value. The values are the same as those of the CSS font property. The order of definitions in the string is: font-style, font-variant, font-weight, font-size, line-height, font-family.</td></tr>
<tr><td>font-family</td><td>Specifies the family of the font. Default is no value. The values are the same as those of the CSS font-family property.</td></tr>
<tr><td>font-size</td><td>Specifies the size of the font. Default is no value. The font size is defined in points. The values are the same as those of the CSS font-size property.</td></tr>
<tr><td>font-style</td><td>Specifies the amount of slant for a font. Default is normal. The values are the same as those of the CSS font-style property. Allowed values are:
normal
italic
oblique - Treated the same as italic.</td></tr>
<tr><td>font-variant</td><td>Specifies the variant style of a font. Default is normal. The values are the same as those of the CSS font-variant property. Allowed values are:
normal
small-caps</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>font-weight</td><td>Specifies the thickness of the letters of the font. Default is normal. The values are the same as those of the CSS font-weight property. Allowed values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>normal
lighter
100
200
300
400</td><td>Treated as non-bold.</td></tr>
<tr><td>bold
bolder
500
600
700
800
900</td><td>Treated as bold.</td></tr>
</table>
</td></tr>
<tr><td>mso-text-shadow</td><td>Specifies whether a shadow is applied to the text on a text path. Default is false.</td></tr>
<tr><td>text-decoration</td><td>Specifies the style of text decoration. Default is none. The values are the same as those of the CSS text-decoration property. Allowed values are:

none
underline
overline
line-through
blink

</td></tr>
<tr><td>v-rotate-letters</td><td>Specifies whether the letters of the text are rotated counterclockwise by 90 degrees. Default is false.</td></tr>
<tr><td>v-same-letter-heights</td><td>Specifies whether all letters will be the same height regardless of initial case. If true, the lowercase letters are stretched to the</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>height of the uppercase letters. Default is <code>false</code>.</td></tr>
<tr><td><code>v-text-align</code></td><td>Specifies the alignment of text. Default is <code>left</code>. Allowed values are:
<code>left</code>
<code>right</code>
<code>center</code>
<code>justify</code>
<code>letter-justify</code> - Distributes the extra space between the letters.
<code>stretch-justify</code> - Stretches the letters to fill in the space.</td></tr>
<tr><td><code>v-text-kern</code></td><td>Specifies whether kerning is turned on. Default is <code>false</code>.</td></tr>
<tr><td><code>v-text-reverse</code></td><td>Specifies whether the layout order of rows is reversed. Default is <code>false</code>. This is used for vertical text layout.</td></tr>
<tr><td><code>v-text-spacing-mode</code></td><td>Specifies the mode for letter spacing. Default is <code>tightening</code>. This property determines whether space will be removed between each letter (<code>tightening</code>) or added between each letter (<code>tracking</code>). The amount of letter spacing change is defined by the <code>v-text-spacing</code> property. Allowed values are:
<code>tightening</code>
<code>tracking</code></td></tr>
<tr><td><code>v-text-spacing</code></td><td>Specifies the amount of spacing for text in 100ths of single line spacing. Default is 100.</td></tr>
</table>
The line (§6.1.2.12), polyline (§6.1.2.15) and curve (§6.1.2.3) elements ignore the following properties:
<code>top</code>
<code>left</code>
<code>width</code>
<code>height</code>
The following properties are not inherited by an element that references a shapetype element (§6.1.2.20) via the id attribute:
<code>flip</code>
<code>height</code>
<code>left</code>
<code>margin-left</code>
<code>margin-top</code>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>

position
rotation
top
visibility
width
z-index

The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>target (Hyperlink Display Target)</td><td>
Specifies a frame or window that a URL is displayed in. Default is no value. Allowed values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td><targetname></td><td>String containing the name of the frame or window in which to load the document.</td></tr>
<tr><td>_blank</td><td>Specifies that the linked document is loaded into a new blank window. This window is not named.</td></tr>
<tr><td>_media</td><td>Specifies that the linked document is loaded into the Media Bar. Available in Microsoft® Internet Explorer 6 or later.</td></tr>
<tr><td>_parent</td><td>Specifies that the linked document is loaded into the immediate parent of the document containing the link.</td></tr>
<tr><td>_search</td><td>Specifies that the linked document is loaded into the browser's search pane. Available in Microsoft Internet Explorer 5 or greater.</td></tr>
<tr><td>_self</td><td>Specifies that the linked document is loaded into the window in which the link was clicked (the active window).</td></tr>
<tr><td>_top</td><td>Specifies that the linked document is loaded into the topmost window.</td></tr>
</table>
[Example:
<pre>
  <v:shape ...
    href="http://www.openxmlformats.org"
    target="_self" ... >
  </v:shape>
</pre>
end example]
The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>title (Shape Title)</td><td>
Specifies the text displayed when the mouse pointer moves over the shape. Default is no value.
[Example:
</td></tr>
</table>

Attributes	Description
	<v:shape ... title="tooltip" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
userdrawn (Exists In Master Slide) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the user has added the shape to a master slide. Default is `false`. Used by PresentationML. [*Example:* <v:shape ... o:userdrawn="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
userhidden (Hide Script Anchors) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a script anchor is hidden. Default is `false`. If `true`, script anchors stay hidden even if the shape is otherwise visible. A script anchor is the visual representation of a script that when displayed in an application. [*Example:* <v:shape ... o:userhidden="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
wrapcoords (Shape Bounding Polygon)	Specifies the bounding polygon that surrounds a shape. This is specified using a comma-delimited list of x and y coordinates: "x1,y1,x2,y2,x3,y3,..." This is used when text is tightly wrapped around a shape. Default is no value until the mso-wrap-mode style attribute is set to `tight` or `through`. [*Example:* <v:shape ... wrapcoords="0,0 0,200, 200,200, 200,0" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

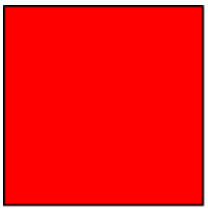
```
<complexType name="CT_PolyLine">
   <choice minOccurs="0" maxOccurs="unbounded">
      <group ref="EG_ShapeElements"/>
      <element ref="o:ink"/>
   </choice>
   <attributeGroup ref="AG_AllCoreAttributes"/>
   <attributeGroup ref="AG_AllShapeAttributes"/>
   <attribute name="points" type="xsd:string" use="optional"/>
</complexType>
```

6.1.2.16 rect (Rectangle)

This element is used to draw a simple rectangle. The CSS2 style content width and height define the width and height of the rectangle.

[*Example:*

```
<v:rect fillcolor="red"
  style="position:relative;top:0;left:0;width:100;height:100">
</v:rect>
```



end example]

Parent Elements
background (§2.2.1); group (§6.1.2.7); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23)

Child Elements	Subclause
anchorlock (Anchor Location Is Locked)	§6.3.2.1
borderbottom (Bottom Border)	§6.3.2.2
borderleft (Left Border)	§6.3.2.3
borderright (Right Border)	§6.3.2.4
bordertop (Top Border)	§6.3.2.5
callout (Callout)	§6.2.2.2
ClientData (Attached Object Data)	§6.4.2.12
clippath (Shape Clipping Path)	§6.2.2.3
extrusion (3D Extrusion)	§6.2.2.10

Child Elements	Subclause
fill (Shape Fill Properties)	§6.1.2.5
formulas (Set of Formulas)	§6.1.2.6
handles (Set of Handles)	§6.1.2.9
imagedata (Image Data)	§6.1.2.11
lock (Shape Protections)	§6.2.2.17
path (Shape Path)	§6.1.2.14
shadow (Shadow Effect)	§6.1.2.18
signatureline (Digital Signature Line)	§6.2.2.29
skew (Skew Transform)	§6.2.2.30
stroke (Line Stroke Settings)	§6.1.2.21
textbox (Text Box)	§6.1.2.22
textdata (VML Diagram Text)	§6.5.2.2
textpath (Text Layout Path)	§6.1.2.23
wrap (Text Wrapping)	§6.3.2.6

Attributes	Description
allowincell (Allow in Table Cell) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape can be placed in a table. Default is `false`. [*Example:* `<v:shape ... o:allowincell="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
allowoverlap (Allow Shape Overlap) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape can overlap another shape. Default is `true`. If `false`, the shape will shift left or right so as not to overlap another shape, similar to the behavior of the HTML float attribute. [*Example:* `<v:shape ... o:allowoverlap="false" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

Attributes	Description

alt (Alternate Text)	Specifies alternative text describing the graphical object. This text should provide a brief description of the shape for use by accessibility tools. Default is no value.
	
	[*Example:* The alt text describes the basic shape:
	
	`<v:shape ... fillcolor="red"`
	`  alt="Red rectangle">`
	`</v:shape>`
	
	The alt text describes the contents of a shape displaying an image:
	
	`<v:shape ... alt="Picture of a sunset">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

borderbottomcolor (Bottom Border Color)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies the bottom border color of an inline shape. Default is no value.
	
	[*Example:*
	
	`<v:shape ... o:borderbottomcolor="red" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

borderleftcolor (Border Left Color)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies the left border color of an inline shape. Default is no value.
	
	[*Example:*
	
	`<v:shape ... o:borderleftcolor="red" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

borderrightcolor (Border Right Color)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies the right border color of an inline shape. Default is no value.
	
	[*Example:*
	
	`<v:shape ... o:borderrightcolor="red" ... >`
	`</v:shape>`
	
	end example]

Attributes	Description
	The possible values for this attribute are defined by the XML Schema string datatype.
bordertopcolor (Border Top Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the top border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:bordertopcolor="red" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
bullet (Graphical Bullet) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the shape is a graphical bullet. Default is `false`. [*Example:* `<v:shape ... o:bullet="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
button (Button Behavior Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape will exhibit button press behavior on click. Default is `false`. [*Example:* `<v:shape ... o:button="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
bwmode (Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies how a shape will render for black-and-white output devices. When a shape is printed on a black-and-white printer or displayed in a black-and-white view in an application, several options are possible. Default is `auto`, which will use o:bwnormal for normal black-and-white rendering and o:bwpure for pure black-and-white rendering. bwnormal and bwpure are subordinate to bwmode. If bwmode is "auto" then the value for bwnormal or bwpure is used depending on what the output format is. An application may define for itself what, if any, difference there is between normal B&W and pure B&W. For example, normal B&W might allow greyscale and pure B&W might not. [*Example:* This shape renders in grayscale in a black-and-white environment:

Attributes	Description
	<v:shape ... o:bwmode="grayscale" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
bwnormal (Normal Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies the black-and-white mode for normal black-and-white output devices. Default is auto. [*Example:* This shape renders in a pale grayscale in a normal black-and-white environment: <v:shape ... o:bwmode="auto" o:bwnormal="lightgrayscale" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
bwpure (Pure Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies the black-and-white mode for pure black-and-white output devices. Default is auto. [*Example:* This shape renders in high contrast when in a pure black-and-white environment: <v:shape ... o:bwmode="auto" o:bwpure="highcontrast" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
chromakey (Image Transparency Color)	Specifies a color value that will be transparent and show anything behind the shape. Default is no value. [*Example:* <v:image ... chromakey="white" ...> </v:image> *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).

Attributes	Description

class (CSS Reference)	Specifies a reference to the definition of a CSS style. Default is no value.
	
	[*Example:* The snippets below are equivalent:
	
	`...`
	`.narrowstyle {width:50;height:100}`
	`...`
	`<v:shape ... class="narrowstyle"`
	`  style="top:1;left:1">`
	`</v:shape>`
	
	`<v:shape ... style="top:1;left:1;`
	`  width:50;height:100">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

clip (Clipping Toggle)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies that the clipping region is active. This is used in conjunction with the clippath (§6.2.2.3) element to create a clipping region.
	
	[*Example:*
	
	`<v:shape ... o:clip="true">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

cliptowrap (Clip to Wrapping Polygon)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies that the clipping region for the shape aligns with the wrapping polygon that tightly surrounds the entire shape (essentially, that the shape shall not be drawn beyond its wrapping polygon's extents – if it does, the shape shall be clipped). Default is `false`.
	
	[*Example:*
	
	`<v:shape ... o:cliptowrap="true">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

connectortype (Shape Connector	Specifies the type of connector used for joining shapes. Default is `straight`.

Attributes	Description
Type) Namespace: urn:schemas-microsoft-com:office:office	[*Example:* `<v:shape ... o:connectortype="elbow" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_ConnectorType simple type (§6.2.3.6).

coordorigin (Coordinate Space Origin)	Specifies the coordinate of the top left corner of the shape's coordinate space. This determines the position of the (0,0) coordinate space origin within the shape's bounding box. Default is "0,0", which places the (0,0) origin at the top left corner of the bounding box.
	
	This affects shape properties that specify coordinate positions, such as the path attribute. Thus a path can be defined against a generic (0,0) origin and the coordorigin value translates the entire path within the shape's bounding space.
	
	[*Example:* The horizontal and vertical coordinate space ranges from -100 to +100 because the coordinate space (coordsize) is 200 by 200 and the top left coordinate is (-100,-100). The (0,0) origin lies at the center of the shape's bounding box, as evidenced by the position of the shape's path within the coordinate space:
	
	`<v:shape ... coordsize="200,200"`
	`coordorigin="-100,-100"`
	`path="m 0,0 l 0,50, 50,50, 50,0 x e">`
	`</v:shape>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

coordsize (Coordinate Space Size)	Specifies the size of the shape's coordinate space in coordinate units. Default is "1000,1000". The physical size of a coordinate unit length is determined by both the size of the coordinate space (coordsize) and the size of the shape (style width and height). The coordsize attribute defines the number of horizontal and vertical subdivisions into which the shape's bounding box is divided. The combination of coordsize and style width/height effective scales the shape anisotropically. [*Example:* The path is 50 units wide and tall, which is 25% of the size of the coordinate

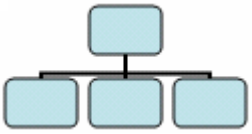
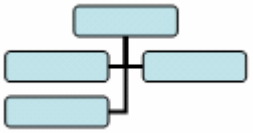
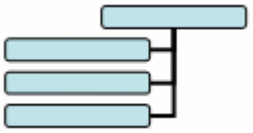
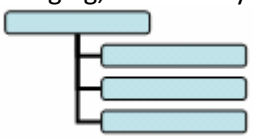
<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>space:

<pre><v:shape ... coordsize="200,200"
  coordorigin="-100,-100"
  path="m 0,0 l 0,50, 50,50, 50,0 x e">
</v:shape></pre>
 

end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>dgmlayout
(Diagram Node Layout Identifier)

Namespace:
urn:schemas-microsoft-com:office:office</td><td>Specifies the type of automatic layout to apply to the diagram node. This is only meaningful for organization charts. Valid values for organization charts are:

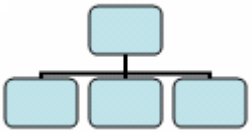
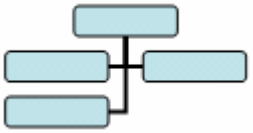
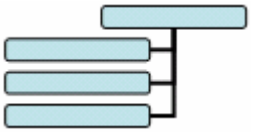
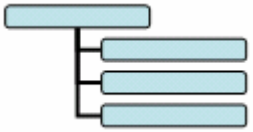
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Top-down, centered layout.
  </td></tr>
<tr><td>1</td><td>Hanging, both sides layout.
  </td></tr>
<tr><td>2</td><td>Hanging, right side layout.
  </td></tr>
<tr><td>3</td><td>Hanging, left side layout.
  </td></tr>
</table>

[Example:

<pre><v:shape ... dgmlayout="1">
</v:shape></pre>
end example]</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>The possible values for this attribute are defined by the XML Schema integer datatype.</td></tr>
<tr><td>dgmlayoutmru
(Diagram Node Recent Layout Identifier)

Namespace:
urn:schemas-microsoft-com:office:office</td><td>Specifies the type of automatic layout most recently used on the child elements of the diagram node. This is only meaningful for organization charts. Valid values for organization charts are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Top-down, centered layout.
  </td></tr>
<tr><td>1</td><td>Hanging, both sides layout.
  </td></tr>
<tr><td>2</td><td>Hanging, right side layout.
  </td></tr>
<tr><td>3</td><td>Hanging, left side layout.
  </td></tr>
</table>

[Example:

<pre><v:shape ... dgmlayout="1">
</v:shape></pre>
end example]

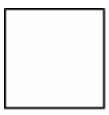
The possible values for this attribute are defined by the XML Schema integer datatype.</td></tr>
<tr><td>dgmnodekind
(Diagram Node Identifier)

Namespace:
urn:schemas-microsoft-com:office:office</td><td>Specifies an optional, application-specific parameter that is intended to be used by the application to tag different types of nodes in a diagram.

[Example:

<pre><v:shape ... dgmnodekind="1">
</v:shape></pre>
end example]</td></tr>
</table>

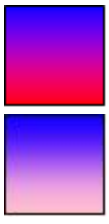
Attributes	Description
	The possible values for this attribute are defined by the XML Schema integer datatype.
doubleclicknotify (Double-click Notification Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies that an event message is sent when a shape is double-clicked. Default is `false`. [*Example:* `<v:shape ... o:doubleclicknotify="true" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
fillcolor (Fill Color)	Specifies the color to use for the fill. Default is `white`. If the fill element (§6.1.2.5) is present, its color attribute takes precedence. Colors are typically specified as either a named color, such as `red`, or six hexadecimal digits representing the red, green and blue values of the color, such as `#00FF30`. Full details are specified in the simple type description. [*Example:* This shape is red if its fill is visible: `<v:shape ... fillcolor="red" ... > </v:shape>` This is equivalent to: `<v:shape ... fillcolor="#ff0000" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).

filled (Shape Fill Toggle)	Specifies whether the closed path will be filled. Default is `true`. This attribute is overridden by the fill on attribute.
	
	[*Example:*
	
	`<v:shape ... filled="f"`
	`  fillcolor="red" ...>`
	`</v:shape>`
	
	 

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
forcedash (Force Dashed Outline) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a dashed outline is used to draw a shape when a shape has no line or fill. Default is `false`. Used by PresentationML placeholders to draw a dashed outline when there is no line and no fill for a shape. [*Example:* `<v:shape ... o:forcedash="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
hr (Horizontal Rule Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies that a shape is a horizontal rule. Default is `false`. [*Example:* `<v:shape ... o:hr="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
hralign (Horizontal Rule Alignment) Namespace: urn:schemas-microsoft-com:office:office	Specifies the alignment of a horizontal rule. Default is `left`. [*Example:* `<v:shape ... o:hralign="center" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_HrAlign simple type (§6.2.3.14).
href (Hyperlink Target)	Specifies a hyperlink URL target for the shape. Default is no value. [*Example:*

Attributes	Description
	<v:shape ... href="http://www.openxmlformats.org" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
hrnoshade (Horizontal Rule 3D Shading Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies that the horizontal rule does not have 3-D shading. Default is `false`. [*Example:* <v:shape ... o:hrnoshade="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
hrpct (Horizontal Rule Length Percentage) Namespace: urn:schemas-microsoft-com:office:office	Specifies the length of a horizontal rule as a percentage of page width. Default is 0. [*Example:* <v:shape ... o:hrpct="85" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema float datatype.
hrstd (Horizontal Rule Standard Display Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape is displayed as a standard horizontal rule. Only applies if hr is `true`. Default is `false`. [*Example:* <v:shape ... o:hrstd="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
id (Unique Identifier)	Specifies a unique identifier that can be used to reference a VML object. Default is no value. [*Example:*

Attributes	Description
	<v:shape ... id="myShape" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
insetmode (Text Inset Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the application calculates the internal text margin instead of using the inset attribute. Default is `custom`. This attribute is only meaningful for text boxes. [*Example:* <v:shape ... o:insetmode="auto" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_InsetMode simple type (§6.2.3.15).
insetpen (Inset Border From Path)	Specifies that the border shall be displayed inside of the path defining the shape, rather than along the path (the default border placement), or outside of the path as might be done with an image. [*Example:* <v:shape ... insetpen="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
ole (Embedded Object Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the shape is an embedded object. Default is `false`. [*Example:* <v:shape ... o:ole="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalseBlank simple type (§6.2.3.24).
oleicon (Embedded Object Icon Toggle)	Specifies whether an embedded object will be displayed as an icon. Default is `false`. [*Example:*

Attributes	Description
Namespace: urn:schemas-microsoft-com:office:office	<v:shape ... o:oleicon="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
oned (Shape Handle Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the extra handles of a shape are hidden. If `true`, hides all shape handles except the top left and bottom right; that is, the same handles that are used for a straight line segment. Default is `false`. [*Example:* <v:shape ... o:oned="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
opacity (Fill Color Opacity)	Specifies the opacity of the primary fill color. Default is 1.0. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8. [*Example:* The red color is 25% opaque: <v:fill type="gradient" color="red" color2="blue" opacity=".25"> </v:fill>  opacity="1" opacity=".25" *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
preferrelative (Relative Resize Toggle)	Specifies whether the original size of an object is saved after reformatting. Default is `false`. If `true`, the original size of the object is stored and all resizing is based on a percentage of that original size. Otherwise, each resizing will reset the scale to 100%.

Attributes	Description
Namespace: urn:schemas-microsoft-com:office:office	[*Example:* `<v:shape ... o:preferrelative="true" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
print (Print Toggle)	Specifies whether the shape will be printed. Default is `true`. [*Example:* `<v:shape ... print="false" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
regroupid (Regroup ID) Namespace: urn:schemas-microsoft-com:office:office	Specifies a previous group for a shape. An ID number is used to identify groups of shapes that are no longer grouped. This allows shapes to be regrouped programmatically. [*Example:* The shape was part of a group identified by the ID 040754: `<v:shape ... o:regroupid="040754" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema integer datatype.
spid (Optional String) Namespace: urn:schemas-microsoft-com:office:office	Specifies an optional string that an application may use to Identify the particular shape. Default is no value. The possible values for this attribute are defined by the XML Schema string datatype.
spt (Optional Number) Namespace: urn:schemas-microsoft-com:office:office	Specifies an optional number that an application may use to associate the particular shape with a defined shape type. Default is 0. The possible values for this attribute are defined by the XML Schema float datatype.

Attributes	Description

strokecolor (Shape Stroke Color)	Specifies the primary color of the brush to use to stroke the path of the shape. Default is `black`. The color attribute of the stroke element (§6.1.2.21) overrides this. Colors are typically specified as either a named color, such as `red`, or six hexadecimal digits representing the red, green and blue values of the color, such as `#00FF30`. Full details are specified in the simple type description.
	
	[*Example:*
	
	`<v:shape ... strokecolor="red" ...>`
	`</v:shape>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).

stroked (Shape Stroke Toggle)	Specifies whether the path defining the shape is stroked with a solid line. The stroke element (§6.1.2.21) defines other strokes. The on attribute of the stroke element overrides this attribute. Default is `true`.
	
	[*Example:*
	
	`<v:shape ... fillcolor="red"`
	`  stroked="false" strokecolor="blue"...>`
	`</v:shape>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).

strokeweight (Shape Stroke Weight)	Specifies the width of the brush to use to stroke the path. Default is 1 point. If a number is given without units, the emu is used. The weight attribute of the stroke element (§6.1.2.21) overrides this attribute.
	
	[*Example:*
	
	`<v:shape ... strokeweight="3pt" ... >`
	`</v:shape>`

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td> 

end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>style (Shape Styling Properties)</td><td>Specifies the CSS2 styling properties of the shape. This uses the syntax described in the "Visual formatting model" of the Cascading Style Sheets, Level 2 specification, a Recommendation of the World Wide Web Consortium available here: http://www.w3.org/TR/REC-CSS2. Full descriptions of each property are not repeated here, but the VML treatment of each property is defined. Allowed properties include:

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>flip</td><td>Specifies that the orientation of a shape is flipped. Default is no value. Allowed values are:

• x - Flip along the y-axis, reversing the x-coordinates.
• y - Flip along the x-axis, reversing the y-coordinates.
• xy - Flip along both the y- and x-axis.
• yx - Flip along both the x- and y-axis.</td></tr>
<tr><td>height</td><td>Specifies the height of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are:

• auto - Default position of an element in the flow of the page.
• <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed.
• <percentage>- Value expressed as a percentage of the parent object's height.</td></tr>
<tr><td>left</td><td>Specifies the position of the left of the containing block of the shape relative to the element left of it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are:

• auto - Default position of an element in the flow of the page.
• <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed.
• <percentage>- Value expressed as a percentage of the</td></tr>
</table>
</td></tr>
</table>

Attributes	Description
	parent object's width.
margin-bottom	Specifies the position of the bottom of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
margin-left	Specifies the position of the left of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
margin-right	Specifies the position of the right of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
margin-top	Specifies the position of the top of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page.

Attributes	Description	
		• <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	mso-position-horizontal	Specifies the horizontal positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute • left • center • right • inside • outside
	mso-position-horizontal-relative	Specifies relative horizontal position data for objects in WordprocessingML documents. This modifies the mso-position-horizontal property. Default is text. Allowed values are: • margin • page • text • char
	mso-position-vertical	Specifies the vertical positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute • top • center • bottom • inside • outside
	mso-position-vertical-relative	Specifies relative vertical position data for objects in WordprocessingML documents. This modifies the mso-position-vertical property. Default is text. Allowed values are: • margin • page

Attributes	Description
	• text • line
mso-wrap-distance-bottom	Specifies the distance from the bottom of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-distance-left	Specifies the distance from the left side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-distance-right	Specifies the distance from the right side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-distance-top	Specifies the distance from the top of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-edited	Specifies whether the wrap coordinates were customized by the user. If the wrap coordinates are generated by an editor, this property is true; otherwise they were customized by a user. Default is false.
mso-wrap-style	Specifies the wrapping mode for text in shapes in WordprocessingML documents. Default is square. Allowed values are: • square - Wraps text inside the shape in a square. • none - Text does not wrap.
position	Specifies the type of positioning used to place an element. Default is static. When the element is contained inside a group, this property must be absolute. Allowed values are: • static - The element is positioned according to the normal flow of the page. The top and left properties are ignored. If the object is anchored inline, this value is used. • absolute - The element is positioned relative to the parent, using the top and left properties. • relative - The element is positioned according to the

Attributes	Description	
		normal flow of the page, but the `top` and `left` properties are used. The overlap of overlapping elements is governed by the `z-index` property.
	`rotation`	Specifies the angle that a shape is rotated, in degrees. Default is 0. Positive angles are clockwise.
	`top`	Specifies the position of the top of the containing block of the shape relative to the element above it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	`visibility`	Specifies whether a shape is displayed. Only `inherit` and `hidden` are used; any other values are mapped to `inherit`. Default is `inherit`. Allowed values are: • `hidden` - The shape is not visible, but is still part of the flow of the objects in the browser. Mouse events are not processed. • `inherit` - The visibility state is inherited from the parent of the shape.
	`width`	Specifies the width of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	`z-index`	Specifies the display order of overlapping shapes. Default is 0. This property shall not be used for shapes anchored inline. Allowed values are: • `auto` - Uses the order that the shapes appear in the page,

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>bottom to top.
• <order>- A number that represents the stacking precedence. Shapes with higher numbers are placed on top of those with lower numbers. Negative numbers are allowed.</td></tr>
</table>
The following properties are only used by the textbox element (§6.1.2.22):
<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>direction</td><td>Specifies the direction of the text in the textbox. Default is ltr. This property is superceded by the mso-direction-alt property if that is specified. Allowed values are:
• ltr - Test is displayed left-to-right.
• rtl - Test is displayed right-to-left.</td></tr>
<tr><td>layout-flow</td><td>Determines the flow of the text layout in a textbox. Default is horizontal. Allowed values are:
• horizontal - Text is displayed horizontally.
• vertical - Text is displayed vertically.
• vertical-ideographic - Ideographic text is displayed vertically.
• horizontal-ideographic - Ideographic text is displayed horizontally.</td></tr>
<tr><td>mso-direction-alt</td><td>Specifies an alternate direction for text in textboxes. Overrides the direction property. The only allowed value is context.</td></tr>
<tr><td>mso-fit-shape-to-text</td><td>Specifies whether the shape stretches to fit the text in the textbox. Default is false.</td></tr>
<tr><td>mso-fit-text-to-shape</td><td>Specifies whether the text stretches to fit the textbox. Default is false.</td></tr>
<tr><td>mso-layout-flow-alt</td><td>Specifies the alternate layout flow for text in textboxes. This property is used instead of layout-flow when the layout flow is from bottom to top for non-ideographic languages. Its only value is bottom-to-top.</td></tr>
<tr><td>mso-next-textbox</td><td>Specifies the ID of the next textbox in a series. Used to keep track of a set of linked textboxes. Default is no value.</td></tr>
</table>
</td></tr>
</table>

Attributes	Description
	mso-rotate: Specifies a specific rotation value for text in a textbox. Default is 0. Allowed values are: • 0 • 90 • 180 • -90
	mso-text-scale: Specifies the scaling factor for fitting text to shapes. Default is 0. This property is only used if mso-fit-text-to-shape is true.
	v-text-anchor: Specifies the vertical anchoring of text in a textbox. Default is top. The alignment of a text anchor only becomes evident if mso-fit-text-to-shape is false. This property is different from the vertical-align CSS property, which is used for ideographic languages. Allowed values are: • top • middle • bottom • top-center • middle-center • bottom-center • top-baseline • bottom-baseline • top-center-baseline • bottom-center-baseline

The following properties are only used by the textpath element (§6.1.2.23):

Property	Description
font	Specifies a compound value of font settings. Default is no value. The values are the same as those of the CSS font property. The order of definitions in the string is: font-style, font-variant, font-weight, font-size, line-height, font-family.
font-family	Specifies the family of the font. Default is no value. The values are the same as those of the CSS font-family property.
font-size	Specifies the size of the font. Default is no value. The font size is defined in points. The values are the same as those of the CSS font-size property.
font-style	Specifies the amount of slant for a font. Default is normal. The values are the same as those of the CSS font-style property. Allowed values are:

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>

normal
italic
oblique - Treated the same as italic.

</td></tr>
<tr><td>font-variant</td><td>Specifies the variant style of a font. Default is normal. The values are the same as those of the CSS font-variant property. Allowed values are:

normal
small-caps

</td></tr>
<tr><td>font-weight</td><td>Specifies the thickness of the letters of the font. Default is normal. The values are the same as those of the CSS font-weight property. Allowed values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>normal
lighter
100
200
300
400</td><td>Treated as non-bold.</td></tr>
<tr><td>bold
bolder
500
600
700
800
900</td><td>Treated as bold.</td></tr>
</table>
</td></tr>
<tr><td>mso-text-shadow</td><td>Specifies whether a shadow is applied to the text on a text path. Default is false.</td></tr>
<tr><td>text-decoration</td><td>Specifies the style of text decoration. Default is none. The values are the same as those of the CSS text-decoration property. Allowed values are:

none

</td></tr>
</table>
</td></tr>
</table>

Attributes	Description
	<table><tr><td></td><td> - underline - overline - line-through - blink </td></tr><tr><td>v-rotate-letters</td><td>Specifies whether the letters of the text are rotated counterclockwise by 90 degrees. Default is false.</td></tr><tr><td>v-same-letter-heights</td><td>Specifies whether all letters will be the same height regardless of initial case. If true, the lowercase letters are stretched to the height of the uppercase letters. Default is false.</td></tr><tr><td>v-text-align</td><td>Specifies the alignment of text. Default is left. Allowed values are: - left - right - center - justify - letter-justify - Distributes the extra space between the letters. - stretch-justify - Stretches the letters to fill in the space. </td></tr><tr><td>v-text-kern</td><td>Specifies whether kerning is turned on. Default is false.</td></tr><tr><td>v-text-reverse</td><td>Specifies whether the layout order of rows is reversed. Default is false. This is used for vertical text layout.</td></tr><tr><td>v-text-spacing-mode</td><td>Specifies the mode for letter spacing. Default is tightening. This property determines whether space will be removed between each letter (tightening) or added between each letter (tracking). The amount of letter spacing change is defined by the v-text-spacing property. Allowed values are: - tightening - tracking </td></tr><tr><td>v-text-spacing</td><td>Specifies the amount of spacing for text in 100ths of single line spacing. Default is 100.</td></tr></table> The line (§6.1.2.12), polyline (§6.1.2.15) and curve (§6.1.2.3) elements ignore the following properties: - top - left - width - height

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>The following properties are not inherited by an element that references a shapetype element (§6.1.2.20) via the id attribute:

• flip
• height
• left
• margin-left
• margin-top
• position
• rotation
• top
• visibility
• width
• z-index

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>target (Hyperlink Display Target)</td><td>Specifies a frame or window that a URL is displayed in. Default is no value. Allowed values are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td><targetname></td><td>String containing the name of the frame or window in which to load the document.</td></tr>
<tr><td>_blank</td><td>Specifies that the linked document is loaded into a new blank window. This window is not named.</td></tr>
<tr><td>_media</td><td>Specifies that the linked document is loaded into the Media Bar. Available in Microsoft® Internet Explorer 6 or later.</td></tr>
<tr><td>_parent</td><td>Specifies that the linked document is loaded into the immediate parent of the document containing the link.</td></tr>
<tr><td>_search</td><td>Specifies that the linked document is loaded into the browser's search pane. Available in Microsoft Internet Explorer 5 or greater.</td></tr>
<tr><td>_self</td><td>Specifies that the linked document is loaded into the window in which the link was clicked (the active window).</td></tr>
<tr><td>_top</td><td>Specifies that the linked document is loaded into the topmost window.</td></tr>
</table>

[Example:

<pre><v:shape ...
  href="http://www.openxmlformats.org"
  target="_self" ... >
</v:shape></pre></td></tr>
</table>

Attributes	Description
	end example] The possible values for this attribute are defined by the XML Schema string datatype.
title (Shape Title)	Specifies the text displayed when the mouse pointer moves over the shape. Default is no value. [*Example:* `<v:shape ... title="tooltip" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
userdrawn (Exists In Master Slide) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the user has added the shape to a master slide. Default is `false`. Used by PresentationML. [*Example:* `<v:shape ... o:userdrawn="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
userhidden (Hide Script Anchors) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a script anchor is hidden. Default is `false`. If `true`, script anchors stay hidden even if the shape is otherwise visible. A script anchor is the visual representation of a script that when displayed in an application. [*Example:* `<v:shape ... o:userhidden="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
wrapcoords (Shape Bounding Polygon)	Specifies the bounding polygon that surrounds a shape. This is specified using a comma-delimited list of x and y coordinates: "x1,y1,x2,y2,x3,y3,..." This is used when text is tightly wrapped around a shape. Default is no value until the mso-wrap-mode style attribute is set to `tight` or `through`.

Attributes	Description

	[*Example:*
	
	`<v:shape ...`
	`  wrapcoords="0,0 0,200, 200,200, 200,0" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

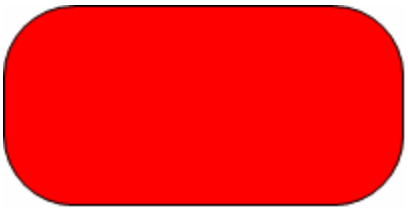
```
<complexType name="CT_Rect">
   <choice maxOccurs="unbounded">
      <group ref="EG_ShapeElements" minOccurs="0" maxOccurs="unbounded"/>
   </choice>
   <attributeGroup ref="AG_AllCoreAttributes"/>
   <attributeGroup ref="AG_AllShapeAttributes"/>
</complexType>
```

6.1.2.17 roundrect (Rounded Rectangle)

This element is used to draw a rectangle with rounded corners. The CSS2 style content width and height define the width and height of the rectangle.

[*Example:*

```
<v:roundrect fillcolor="red" arcsize="35%"
  style="position:relative;top:0;left:0;width:200;height:100">
</v:roundrect>
```



end example]

Parent Elements
background (§2.2.1); group (§6.1.2.7); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23)

Child Elements	Subclause
anchorlock (Anchor Location Is Locked)	§6.3.2.1
borderbottom (Bottom Border)	§6.3.2.2

Child Elements	Subclause
borderleft (Left Border)	§6.3.2.3
borderright (Right Border)	§6.3.2.4
bordertop (Top Border)	§6.3.2.5
callout (Callout)	§6.2.2.2
ClientData (Attached Object Data)	§6.4.2.12
clippath (Shape Clipping Path)	§6.2.2.3
extrusion (3D Extrusion)	§6.2.2.10
fill (Shape Fill Properties)	§6.1.2.5
formulas (Set of Formulas)	§6.1.2.6
handles (Set of Handles)	§6.1.2.9
imagedata (Image Data)	§6.1.2.11
lock (Shape Protections)	§6.2.2.17
path (Shape Path)	§6.1.2.14
shadow (Shadow Effect)	§6.1.2.18
signatureline (Digital Signature Line)	§6.2.2.29
skew (Skew Transform)	§6.2.2.30
stroke (Line Stroke Settings)	§6.1.2.21
textbox (Text Box)	§6.1.2.22
textdata (VML Diagram Text)	§6.5.2.2
textpath (Text Layout Path)	§6.1.2.23
wrap (Text Wrapping)	§6.3.2.6

Attributes	Description
allowincell (Allow in Table Cell) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape can be placed in a table. Default is `false`. [*Example:* `<v:shape ... o:allowincell="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
allowoverlap (Allow Shape Overlap)	Specifies whether a shape can overlap another shape. Default is `true`. If `false`, the shape will shift left or right so as not to overlap another shape, similar to the behavior of the HTML float attribute.

Attributes	Description
Namespace: urn:schemas-microsoft-com:office:office	[*Example:* `<v:shape ... o:allowoverlap="false" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

alt (Alternate Text)	Specifies alternative text describing the graphical object. This text should provide a brief description of the shape for use by accessibility tools. Default is no value.
	
	[*Example:* The alt text describes the basic shape:
	
	`<v:shape ... fillcolor="red" alt="Red rectangle"></v:shape>`
	
	The alt text describes the contents of a shape displaying an image:
	
	`<v:shape ... alt="Picture of a sunset"></v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

arcsize (Rounded Corner Arc Size)	Specifies the amount of roundness for a rounded rectangle as a percentage of half the smaller dimension of the length and width of the rectangle. Default is 20%. An arc size of 0% yields square corners and 100% forms circular corners. A square with an arc size value of 100% is a circle. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8. [*Example:* `<v:roundrect ... arcsize="35%"></v:roundrect>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
borderbottomcolor (Bottom Border Color)	Specifies the bottom border color of an inline shape. Default is no value. [*Example:*

Attributes	Description
Namespace: urn:schemas-microsoft-com:office:office	<v:shape ... o:borderbottomcolor="red" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
borderleftcolor (Border Left Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the left border color of an inline shape. Default is no value. [*Example:* <v:shape ... o:borderleftcolor="red" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
borderrightcolor (Border Right Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the right border color of an inline shape. Default is no value. [*Example:* <v:shape ... o:borderrightcolor="red" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
bordertopcolor (Border Top Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the top border color of an inline shape. Default is no value. [*Example:* <v:shape ... o:bordertopcolor="red" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
bullet (Graphical Bullet) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the shape is a graphical bullet. Default is `false`. [*Example:* <v:shape ... o:bullet="true" ... > </v:shape> *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
button (Button Behavior Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape will exhibit button press behavior on click. Default is `false`. [*Example:* `<v:shape ... o:button="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
bwmode (Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies how a shape will render for black-and-white output devices. When a shape is printed on a black-and-white printer or displayed in a black-and-white view in an application, several options are possible. Default is `auto`, which will use o:bwnormal for normal black-and-white rendering and o:bwpure for pure black-and-white rendering. bwnormal and bwpure are subordinate to bwmode. If bwmode is "auto" then the value for bwnormal or bwpure is used depending on what the output format is. An application may define for itself what, if any, difference there is between normal B&W and pure B&W. For example, normal B&W might allow greyscale and pure B&W might not. [*Example:* This shape renders in grayscale in a black-and-white environment: `<v:shape ... o:bwmode="grayscale" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
bwnormal (Normal Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies the black-and-white mode for normal black-and-white output devices. Default is `auto`. [*Example:* This shape renders in a pale grayscale in a normal black-and-white environment: `<v:shape ... o:bwmode="auto" o:bwnormal="lightgrayscale" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
bwpure (Pure	Specifies the black-and-white mode for pure black-and-white output devices. Default is

Attributes	Description
Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	auto. [*Example:* This shape renders in high contrast when in a pure black-and-white environment: `<v:shape ... o:bwmode="auto" o:bwpure="highcontrast" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
chromakey (Image Transparency Color)	Specifies a color value that will be transparent and show anything behind the shape. Default is no value. [*Example:* `<v:image ... chromakey="white" ...> </v:image>` *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).
class (CSS Reference)	Specifies a reference to the definition of a CSS style. Default is no value. [*Example:* The snippets below are equivalent: `... .narrowstyle {width:50;height:100} ... <v:shape ... class="narrowstyle" style="top:1;left:1"> </v:shape>` `<v:shape ... style="top:1;left:1; width:50;height:100"> </v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
clip (Clipping Toggle)	Specifies that the clipping region is active. This is used in conjunction with the clippath (§6.2.2.3) element to create a clipping region.

Attributes	Description

Namespace: urn:schemas-microsoft-com:office:office	[*Example:*
	
	`<v:shape ... o:clip="true">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

cliptowrap (Clip to Wrapping Polygon)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies that the clipping region for the shape aligns with the wrapping polygon that tightly surrounds the entire shape (essentially, that the shape shall not be drawn beyond its wrapping polygon's extents – if it does, the shape shall be clipped). Default is `false`.
	
	[*Example:*
	
	`<v:shape ... o:cliptowrap="true">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

connectortype (Shape Connector Type)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies the type of connector used for joining shapes. Default is `straight`.
	
	[*Example:*
	
	`<v:shape ... o:connectortype="elbow" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_ConnectorType simple type (§6.2.3.6).

coordorigin (Coordinate Space Origin)	Specifies the coordinate of the top left corner of the shape's coordinate space. This determines the position of the (0,0) coordinate space origin within the shape's bounding box. Default is "0,0", which places the (0,0) origin at the top left corner of the bounding box. This affects shape properties that specify coordinate positions, such as the path attribute. Thus a path can be defined against a generic (0,0) origin and the coordorigin value translates the entire path within the shape's bounding space. [*Example:* The horizontal and vertical coordinate space ranges from -100 to +100 because the coordinate space (coordsize) is 200 by 200 and the top left coordinate is (-100,-100).

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>The (0,0) origin lies at the center of the shape's bounding box, as evidenced by the position of the shape's path within the coordinate space:

<pre><v:shape ... coordsize="200,200"
  coordorigin="-100,-100"
  path="m 0,0 l 0,50, 50,50, 50,0 x e">
</v:shape></pre>
 

end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>coordsize
(Coordinate Space Size)</td><td>Specifies the size of the shape's coordinate space in coordinate units. Default is "1000,1000".

The physical size of a coordinate unit length is determined by both the size of the coordinate space (coordsize) and the size of the shape (style width and height). The coordsize attribute defines the number of horizontal and vertical subdivisions into which the shape's bounding box is divided. The combination of coordsize and style width/height effective scales the shape anisotropically.

[Example: The path is 50 units wide and tall, which is 25% of the size of the coordinate space:

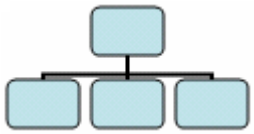
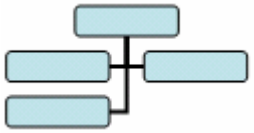
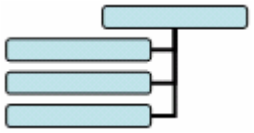
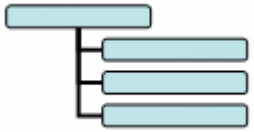
<pre><v:shape ... coordsize="200,200"
  coordorigin="-100,-100"
  path="m 0,0 l 0,50, 50,50, 50,0 x e">
</v:shape></pre>
 

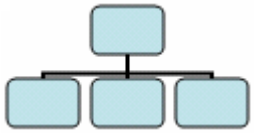
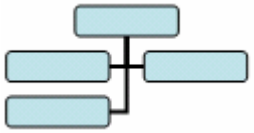
end example]

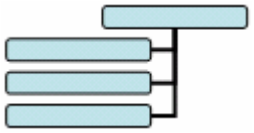
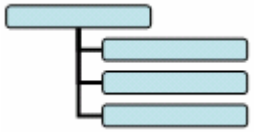
The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>dgmlayout
(Diagram Node Layout Identifier)

Namespace: urn:schemas-microsoft-</td><td>Specifies the type of automatic layout to apply to the diagram node. This is only meaningful for organization charts. Valid values for organization charts are:

<table><tr><th>Value</th><th>Description</th></tr><tr><td>0</td><td>Top-down, centered layout.</td></tr></table></td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>com:office:office</td><td>
<table>
<tr><td></td><td>  </td></tr>
<tr><td>1</td><td>Hanging, both sides layout.
  </td></tr>
<tr><td>2</td><td>Hanging, right side layout.
  </td></tr>
<tr><td>3</td><td>Hanging, left side layout.
  </td></tr>
</table>
[Example:
<pre><v:shape ... dgmlayout="1">
</v:shape></pre>
end example]
The possible values for this attribute are defined by the XML Schema integer datatype.
</td></tr>
<tr><td>dgmlayoutmru (Diagram Node Recent Layout Identifier)

Namespace: urn:schemas-microsoft-com:office:office</td><td>
Specifies the type of automatic layout most recently used on the child elements of the diagram node. This is only meaningful for organization charts. Valid values for organization charts are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Top-down, centered layout.
  </td></tr>
<tr><td>1</td><td>Hanging, both sides layout.
  </td></tr>
<tr><td>2</td><td>Hanging, right side layout.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>  </td></tr>
<tr><td>3</td><td>Hanging, left side layout.
  </td></tr>
</table>

[Example:

<pre><v:shape ... dgmlayout="1">
</v:shape></pre>
end example]

The possible values for this attribute are defined by the XML Schema integer datatype.</td></tr>
<tr><td>dgmnodekind (Diagram Node Identifier)

Namespace: urn:schemas-microsoft-com:office:office</td><td>Specifies an optional, application-specific parameter that is intended to be used by the application to tag different types of nodes in a diagram.

[Example:

<pre><v:shape ... dgmnodekind="1">
</v:shape></pre>
end example]

The possible values for this attribute are defined by the XML Schema integer datatype.</td></tr>
<tr><td>doubleclicknotify (Double-click Notification Toggle)

Namespace: urn:schemas-microsoft-com:office:office</td><td>Specifies that an event message is sent when a shape is double-clicked. Default is <code>false</code>.

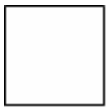
[Example:

<pre><v:shape ... o:doubleclicknotify="true" ... >
</v:shape></pre>
end example]

The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).</td></tr>
<tr><td>fillcolor (Fill Color)</td><td>Specifies the color to use for the fill. Default is <code>white</code>. If the fill element (§6.1.2.5) is present, its color attribute takes precedence. Colors are typically specified as either a named color, such as <code>red</code>, or six hexadecimal digits representing the red, green and blue values of the color, such as <code>#00FF30</code>. Full details are specified in the simple type description.</td></tr>
</table>

Attributes	Description

	[*Example:* This shape is red if its fill is visible:
	
	`<v:shape ... fillcolor="red" ... ></v:shape>`
	
	This is equivalent to:
	
	`<v:shape ... fillcolor="#ff0000" ... ></v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).

filled (Shape Fill Toggle)	Specifies whether the closed path will be filled. Default is `true`. This attribute is overridden by the fill on attribute. [*Example:* `<v:shape ... filled="f" fillcolor="red" ...></v:shape>`  *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
forcedash (Force Dashed Outline) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a dashed outline is used to draw a shape when a shape has no line or fill. Default is `false`. Used by PresentationML placeholders to draw a dashed outline when there is no line and no fill for a shape. [*Example:* `<v:shape ... o:forcedash="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type

Attributes	Description
	(§6.2.3.23).
hr (Horizontal Rule Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies that a shape is a horizontal rule. Default is `false`. [*Example:* `<v:shape ... o:hr="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
hralign (Horizontal Rule Alignment) Namespace: urn:schemas-microsoft-com:office:office	Specifies the alignment of a horizontal rule. Default is `left`. [*Example:* `<v:shape ... o:hralign="center" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_HrAlign simple type (§6.2.3.14).
href (Hyperlink Target)	Specifies a hyperlink URL target for the shape. Default is no value. [*Example:* `<v:shape ... href="http://www.openxmlformats.org" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
hrnoshade (Horizontal Rule 3D Shading Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies that the horizontal rule does not have 3-D shading. Default is `false`. [*Example:* `<v:shape ... o:hrnoshade="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
hrpct (Horizontal	Specifies the length of a horizontal rule as a percentage of page width. Default is 0.

Attributes	Description
Rule Length Percentage) Namespace: urn:schemas-microsoft-com:office:office	[*Example:* `<v:shape ... o:hrpct="85" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema float datatype.
hrstd (Horizontal Rule Standard Display Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape is displayed as a standard horizontal rule. Only applies if hr is `true`. Default is `false`. [*Example:* `<v:shape ... o:hrstd="true" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
id (Unique Identifier)	Specifies a unique identifier that can be used to reference a VML object. Default is no value. [*Example:* `<v:shape ... id="myShape" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
insetmode (Text Inset Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the application calculates the internal text margin instead of using the inset attribute. Default is `custom`. This attribute is only meaningful for text boxes. [*Example:* `<v:shape ... o:insetmode="auto" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_InsetMode simple type (§6.2.3.15).

Attributes	Description
insetpen (Inset Border From Path)	Specifies that the border shall be displayed inside of the path defining the shape, rather than along the path (the default border placement), or outside of the path as might be done with an image. [*Example:* `<v:shape ... insetpen="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
ole (Embedded Object Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the shape is an embedded object. Default is `false`. [*Example:* `<v:shape ... o:ole="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalseBlank simple type (§6.2.3.24).
oleicon (Embedded Object Icon Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether an embedded object will be displayed as an icon. Default is `false`. [*Example:* `<v:shape ... o:oleicon="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
oned (Shape Handle Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the extra handles of a shape are hidden. If `true`, hides all shape handles except the top left and bottom right; that is, the same handles that are used for a straight line segment. Default is `false`. [*Example:* `<v:shape ... o:oned="true" ... ></v:shape>` *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

opacity (Fill Color Opacity)	Specifies the opacity of the primary fill color. Default is 1.0. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8.
	
	[*Example:* The red color is 25% opaque:
	
	`<v:fill type="gradient" color="red"`
	`  color2="blue" opacity=".25">`
	`</v:fill>`
	
	  `opacity="1"`
	  `opacity=".25"`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

preferrelative (Relative Resize Toggle)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies whether the original size of an object is saved after reformatting. Default is `false`. If `true`, the original size of the object is stored and all resizing is based on a percentage of that original size. Otherwise, each resizing will reset the scale to 100%.
	
	[*Example:*
	
	`<v:shape ... o:preferrelative="true" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

print (Print Toggle)	Specifies whether the shape will be printed. Default is `true`.
	
	[*Example:*
	
	`<v:shape ... print="false" ... >`
	`</v:shape>`
	
	end example]

Attributes	Description
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
regroupid (Regroup ID) Namespace: urn:schemas-microsoft-com:office:office	Specifies a previous group for a shape. An ID number is used to identify groups of shapes that are no longer grouped. This allows shapes to be regrouped programmatically. [*Example:* The shape was part of a group identified by the ID 040754: `<v:shape ... o:regroupid="040754" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema integer datatype.
spid (Optional String) Namespace: urn:schemas-microsoft-com:office:office	Specifies an optional string that an application may use to Identify the particular shape. Default is no value. The possible values for this attribute are defined by the XML Schema string datatype.
spt (Optional Number) Namespace: urn:schemas-microsoft-com:office:office	Specifies an optional number that an application may use to associate the particular shape with a defined shape type. Default is 0. The possible values for this attribute are defined by the XML Schema float datatype.
strokecolor (Shape Stroke Color)	Specifies the primary color of the brush to use to stroke the path of the shape. Default is `black`. The color attribute of the stroke element (§6.1.2.21) overrides this. Colors are typically specified as either a named color, such as `red`, or six hexadecimal digits representing the red, green and blue values of the color, such as `#00FF30`. Full details are specified in the simple type description. [*Example:* `<v:shape ... strokecolor="red" ...> </v:shape>`  *end example*] The possible values for this attribute are defined by the ST_ColorType simple type

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>(§6.1.3.1).</td></tr>
<tr><td>stroked (Shape Stroke Toggle)</td><td>Specifies whether the path defining the shape is stroked with a solid line. The stroke element (§6.1.2.21) defines other strokes. The on attribute of the stroke element overrides this attribute. Default is true.

[Example:

<pre><v:shape ... fillcolor="red"
  stroked="false" strokecolor="blue"...>
</v:shape></pre>
 

end example]

The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).</td></tr>
<tr><td>strokeweight (Shape Stroke Weight)</td><td>Specifies the width of the brush to use to stroke the path. Default is 1 point. If a number is given without units, the emu is used. The weight attribute of the stroke element (§6.1.2.21) overrides this attribute.

[Example:

<pre><v:shape ... strokeweight="3pt" ... >
</v:shape></pre>
 

end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>style (Shape Styling Properties)</td><td>Specifies the CSS2 styling properties of the shape. This uses the syntax described in the "Visual formatting model" of the Cascading Style Sheets, Level 2 specification, a Recommendation of the World Wide Web Consortium available here: http://www.w3.org/TR/REC-CSS2. Full descriptions of each property are not repeated here, but the VML treatment of each property is defined. Allowed properties include:

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>flip</td><td>Specifies that the orientation of a shape is flipped. Default is no value. Allowed values are:</td></tr>
</table>
</td></tr>
</table>

Attributes	Description
	• x - Flip along the y-axis, reversing the x-coordinates. • y - Flip along the x-axis, reversing the y-coordinates. • xy - Flip along both the y- and x-axis. • yx - Flip along both the x- and y-axis.
`height`	Specifies the height of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
`left`	Specifies the position of the left of the containing block of the shape relative to the element left of it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
`margin-bottom`	Specifies the position of the bottom of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
`margin-left`	Specifies the position of the left of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are:

Attributes	Description	
		• auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	margin-right	Specifies the position of the right of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	margin-top	Specifies the position of the top of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	mso-position-horizontal	Specifies the horizontal positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute • left • center • right • inside • outside
	mso-position-	Specifies relative horizontal position data for objects in

Attributes	Description	
	horizontal-relative	WordprocessingML documents. This modifies the mso-position-horizontal property. Default is text. Allowed values are: • margin • page • text • char
	mso-position-vertical	Specifies the vertical positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute • top • center • bottom • inside • outside
	mso-position-vertical-relative	Specifies relative vertical position data for objects in WordprocessingML documents. This modifies the mso-position-vertical property. Default is text. Allowed values are: • margin • page • text • line
	mso-wrap-distance-bottom	Specifies the distance from the bottom of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
	mso-wrap-distance-left	Specifies the distance from the left side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
	mso-wrap-distance-right	Specifies the distance from the right side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.

Attributes	Description	
	mso-wrap-distance-top	Specifies the distance from the top of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
	mso-wrap-edited	Specifies whether the wrap coordinates were customized by the user. If the wrap coordinates are generated by an editor, this property is true; otherwise they were customized by a user. Default is false.
	mso-wrap-style	Specifies the wrapping mode for text in shapes in WordprocessingML documents. Default is square. Allowed values are: • square - Wraps text inside the shape in a square. • none - Text does not wrap.
	position	Specifies the type of positioning used to place an element. Default is static. When the element is contained inside a group, this property must be absolute. Allowed values are: • static - The element is positioned according to the normal flow of the page. The top and left properties are ignored. If the object is anchored inline, this value is used. • absolute - The element is positioned relative to the parent, using the top and left properties. • relative - The element is positioned according to the normal flow of the page, but the top and left properties are used. The overlap of overlapping elements is governed by the z-index property.
	rotation	Specifies the angle that a shape is rotated, in degrees. Default is 0. Positive angles are clockwise.

Attributes	Description	
	top	Specifies the position of the top of the containing block of the shape relative to the element above it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	visibility	Specifies whether a shape is displayed. Only inherit and hidden are used; any other values are mapped to inherit. Default is inherit. Allowed values are: • hidden - The shape is not visible, but is still part of the flow of the objects in the browser. Mouse events are not processed. • inherit - The visibility state is inherited from the parent of the shape.
	width	Specifies the width of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	z-index	Specifies the display order of overlapping shapes. Default is 0. This property shall not be used for shapes anchored inline. Allowed values are: • auto - Uses the order that the shapes appear in the page, bottom to top. • <order>- A number that represents the stacking precedence. Shapes with higher numbers are placed on top of those with lower numbers. Negative numbers are allowed.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
The following properties are only used by the textbox element (§6.1.2.22):
<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>direction</td><td>Specifies the direction of the text in the textbox. Default is ltr. This property is superceded by the mso-direction-alt property if that is specified. Allowed values are:
ltr - Test is displayed left-to-right.
rtl - Test is displayed right-to-left.</td></tr>
<tr><td>layout-flow</td><td>Determines the flow of the text layout in a textbox. Default is horizontal. Allowed values are:
horizontal - Text is displayed horizontally.
vertical - Text is displayed vertically.
vertical-ideographic - Ideographic text is displayed vertically.
horizontal-ideographic - Ideographic text is displayed horizontally.</td></tr>
<tr><td>mso-direction-alt</td><td>Specifies an alternate direction for text in textboxes. Overrides the direction property. The only allowed value is context.</td></tr>
<tr><td>mso-fit-shape-to-text</td><td>Specifies whether the shape stretches to fit the text in the textbox. Default is false.</td></tr>
<tr><td>mso-fit-text-to-shape</td><td>Specifies whether the text stretches to fit the textbox. Default is false.</td></tr>
<tr><td>mso-layout-flow-alt</td><td>Specifies the alternate layout flow for text in textboxes. This property is used instead of layout-flow when the layout flow is from bottom to top for non-ideographic languages. Its only value is bottom-to-top.</td></tr>
<tr><td>mso-next-textbox</td><td>Specifies the ID of the next textbox in a series. Used to keep track of a set of linked textboxes. Default is no value.</td></tr>
<tr><td>mso-rotate</td><td>Specifies a specific rotation value for text in a textbox. Default is 0. Allowed values are:
0
90
180</td></tr>
</table>
</td></tr>
</table>

Attributes	Description
	<table><tr><td></td><td>• -90</td></tr><tr><td>mso-text-scale</td><td>Specifies the scaling factor for fitting text to shapes. Default is 0. This property is only used if mso-fit-text-to-shape is true.</td></tr><tr><td>v-text-anchor</td><td>Specifies the vertical anchoring of text in a textbox. Default is top. The alignment of a text anchor only becomes evident if mso-fit-text-to-shape is false. This property is different from the vertical-align CSS property, which is used for ideographic languages. Allowed values are: • top • middle • bottom • top-center • middle-center • bottom-center • top-baseline • bottom-baseline • top-center-baseline • bottom-center-baseline</td></tr></table> The following properties are only used by the textpath element (§6.1.2.23): <table><tr><th>Property</th><th>Description</th></tr><tr><td>font</td><td>Specifies a compound value of font settings. Default is no value. The values are the same as those of the CSS font property. The order of definitions in the string is: font-style, font-variant, font-weight, font-size, line-height, font-family.</td></tr><tr><td>font-family</td><td>Specifies the family of the font. Default is no value. The values are the same as those of the CSS font-family property.</td></tr><tr><td>font-size</td><td>Specifies the size of the font. Default is no value. The font size is defined in points. The values are the same as those of the CSS font-size property.</td></tr><tr><td>font-style</td><td>Specifies the amount of slant for a font. Default is normal. The values are the same as those of the CSS font-style property. Allowed values are: • normal • italic • oblique - Treated the same as italic.</td></tr><tr><td>font-variant</td><td>Specifies the variant style of a font. Default is normal. The values are the same as those of the CSS font-variant property. Allowed values are:</td></tr></table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>

normal
small-caps

</td></tr>
<tr><td>font-weight</td><td>Specifies the thickness of the letters of the font. Default is normal. The values are the same as those of the CSS font-weight property. Allowed values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>normal
lighter
100
200
300
400</td><td>Treated as non-bold.</td></tr>
<tr><td>bold
bolder
500
600
700
800
900</td><td>Treated as bold.</td></tr>
</table>
</td></tr>
<tr><td>mso-text-shadow</td><td>Specifies whether a shadow is applied to the text on a text path. Default is false.</td></tr>
<tr><td>text-decoration</td><td>Specifies the style of text decoration. Default is none. The values are the same as those of the CSS text-decoration property. Allowed values are:

none
underline
overline
line-through
blink

</td></tr>
<tr><td>v-rotate-</td><td>Specifies whether the letters of the text are rotated counterclockwise by 90 degrees. Default is false.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>letters</td><td></td></tr>
<tr><td>v-same-letter-heights</td><td>Specifies whether all letters will be the same height regardless of initial case. If true, the lowercase letters are stretched to the height of the uppercase letters. Default is false.</td></tr>
<tr><td>v-text-align</td><td>Specifies the alignment of text. Default is left. Allowed values are:
left
right
center
justify
letter-justify - Distributes the extra space between the letters.
stretch-justify - Stretches the letters to fill in the space.</td></tr>
<tr><td>v-text-kern</td><td>Specifies whether kerning is turned on. Default is false.</td></tr>
<tr><td>v-text-reverse</td><td>Specifies whether the layout order of rows is reversed. Default is false. This is used for vertical text layout.</td></tr>
<tr><td>v-text-spacing-mode</td><td>Specifies the mode for letter spacing. Default is tightening. This property determines whether space will be removed between each letter (tightening) or added between each letter (tracking). The amount of letter spacing change is defined by the v-text-spacing property. Allowed values are:
tightening
tracking</td></tr>
<tr><td>v-text-spacing</td><td>Specifies the amount of spacing for text in 100ths of single line spacing. Default is 100.</td></tr>
</table>
The line (§6.1.2.12), polyline (§6.1.2.15) and curve (§6.1.2.3) elements ignore the following properties:
top
left
width
height
The following properties are not inherited by an element that references a shapetype element (§6.1.2.20) via the id attribute:
flip
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>

height
left
margin-left
margin-top
position
rotation
top
visibility
width
z-index

The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>target (Hyperlink Display Target)</td><td>
Specifies a frame or window that a URL is displayed in. Default is no value. Allowed values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td><targetname></td><td>String containing the name of the frame or window in which to load the document.</td></tr>
<tr><td>_blank</td><td>Specifies that the linked document is loaded into a new blank window. This window is not named.</td></tr>
<tr><td>_media</td><td>Specifies that the linked document is loaded into the Media Bar. Available in Microsoft® Internet Explorer 6 or later.</td></tr>
<tr><td>_parent</td><td>Specifies that the linked document is loaded into the immediate parent of the document containing the link.</td></tr>
<tr><td>_search</td><td>Specifies that the linked document is loaded into the browser's search pane. Available in Microsoft Internet Explorer 5 or greater.</td></tr>
<tr><td>_self</td><td>Specifies that the linked document is loaded into the window in which the link was clicked (the active window).</td></tr>
<tr><td>_top</td><td>Specifies that the linked document is loaded into the topmost window.</td></tr>
</table>
[Example:
<pre>
  <v:shape ...
    href="http://www.openxmlformats.org"
    target="_self" ... >
  </v:shape>
</pre>
end example]
The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>title (Shape Title)</td><td>Specifies the text displayed when the mouse pointer moves over the shape. Default is no</td></tr>
</table>

Attributes	Description
	value. [*Example:* `<v:shape ... title="tooltip" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
userdrawn (Exists In Master Slide) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the user has added the shape to a master slide. Default is `false`. Used by PresentationML. [*Example:* `<v:shape ... o:userdrawn="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
userhidden (Hide Script Anchors) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a script anchor is hidden. Default is `false`. If `true`, script anchors stay hidden even if the shape is otherwise visible. A script anchor is the visual representation of a script that when displayed in an application. [*Example:* `<v:shape ... o:userhidden="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
wrapcoords (Shape Bounding Polygon)	Specifies the bounding polygon that surrounds a shape. This is specified using a comma-delimited list of x and y coordinates: "x1,y1,x2,y2,x3,y3,..." This is used when text is tightly wrapped around a shape. Default is no value until the mso-wrap-mode style attribute is set to `tight` or `through`. [*Example:* `<v:shape ... wrapcoords="0,0 0,200, 200,200, 200,0" ... ></v:shape>`

Attributes	Description
	end example] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RoundRect">
   <choice maxOccurs="unbounded">
      <group ref="EG_ShapeElements" minOccurs="0" maxOccurs="unbounded"/>
   </choice>
   <attributeGroup ref="AG_AllCoreAttributes"/>
   <attributeGroup ref="AG_AllShapeAttributes"/>
   <attribute name="arcsize" type="xsd:string" use="optional"/>
</complexType>
```

6.1.2.18 shadow (Shadow Effect)

This element adds shadow effects to a shape. The on attribute must be `true` for the shadow to be displayed.

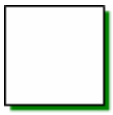
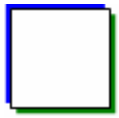
[*Example:*

```
<v:shadow on="true" type="perspective"
  matrix="1.25,-2,,1.5,,.000001"
  offset="38pt,-6pt">
</v:shadow>
```

end example]

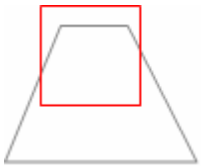
Parent Elements
arc (§6.1.2.1); background (§2.2.1); curve (§6.1.2.3); group (§6.1.2.7); image (§6.1.2.10); line (§6.1.2.12); object (§2.3.3.19); oval (§6.1.2.13); pict (§2.3.3.21); pict (§2.9.23); polyline (§6.1.2.15); rect (§6.1.2.16); roundrect (§6.1.2.17); shape (§6.1.2.19); shapedefaults (§6.2.2.27); shapetype (§6.1.2.20)

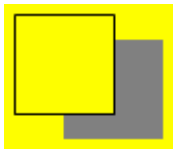
Attributes	Description
color (Shadow Primary Color)	Specifies the color of the primary shadow. Default is gray (RGB 128,128,128). [*Example:* `<v:shadow on="true" color="green">`

Attributes	Description
	</v:shadow> Applied to a simple square the shadow looks like this:  *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).
color2 (Shadow Secondary Color)	Specifies the color of the second shadow, or highlight in an embossed or engraved shadow. Default is light gray (RGB 203,203,203). [*Example:* <v:shadow on="true" type="double" color="green" color2="blue"> </v:shadow>  *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).
id (Unique Identifier)	Specifies a unique identifier that can be used to reference a VML object. Default is no value. [*Example:* <v:shape ... id="myShape" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
matrix (Shadow Perspective Matrix)	Specifies a perspective transform for a shadow. Default is no value. The matrix is given in the form "s_{xx}, s_{xy}, s_{yx}, s_{yy}, p_x, p_y" where s = scale and p = perspective. If the offset attribute is in absolute units then p_x, p_y are in 1/EMU units;

Attributes	Description
	otherwise they are an inverse fraction of the shape size. [*Example:* The following snippets explain the matrix parameters. The shadow is applied to a simple square with no fill and a red stroke color (note there is a default shadow offset):  `matrix=",,,,,"` s_{xx}, s_{yy} specify scaling factors for the x and y dimensions:  `matrix="2,,,,,"`  `matrix=",,,2,,"` s_{xy}, s_{yx} specify skews in the x and y dimensions:  `matrix=",2,,,,"`  `matrix=",,-2,,,"` p_x, p_y effectively set the perspective trapezoid skews along the x and y dimensions:  `matrix=",,,,.000001,"`

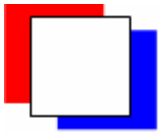
Attributes	Description

	 `matrix=",,,,,-.000002"`
	
	end example]
	
	[*Example:*
	
	`<v:shadow on="true" type="perspective"`
	`matrix="1.25,-2,,1.5,,.000001"`
	`offset="38pt,-6pt">`
	`</v:shadow>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

obscured (Shadow Transparency)	Specifies whether a shadow is transparent. Default is `false`. If `true`, the shadow is transparent if there is no fill on the shape.
	
	[*Example:*
	
	`<v:background fillcolor="yellow"/>`
	`<v:shape style="width:50;height:50"`
	`filled="false" fillcolor="red"`
	`path="m 0,0 l 0,1000 1000,1000 1000,0 x e">`
	`<v:shadow on="true" offset="50%,25%"`
	`obscured="true">`
	`</v:shadow>`
	`</v:shape>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).

Attributes	Description

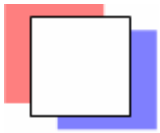
offset (Shadow Primary Offset)	Specifies the primary shadow's x,y offset from the shape's location. Default is "2pt,2pt".
	
	Values are either an absolute measurement or a fractional value of the shape dimensions, from –50% to 50%.
	
	[*Example:*
	
	`<v:shadow on="true" offset="50%,25%">`
	`</v:shadow>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

offset2 (Shadow Secondary Offset)	Specifies the secondary shadow's x,y offset from the shape's location. Default is "-2pt,-2pt".
	
	[*Example:*
	
	`<v:shadow type="double" on="true"`
	`color="blue" offset="10pt,5pt"`
	`color2="red" offset2="-10pt,-5pt">`
	`</v:shadow>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

on (Shadow Toggle)	Specifies whether to show a shadow. Default is `true`. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
opacity (Shadow Opacity)	Specifies the opacity of the shadow. Default is 1. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8. [*Example:*

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>

```
<v:shadow type="double" on="true" opacity=".5"
  color="blue" offset="10pt,5pt"
  color2="red" offset2="-10pt,-5pt">
</v:shadow>
```



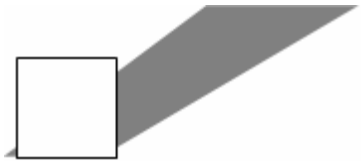
end example]

The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>origin (Shadow Origin)</td><td>

Specifies the center of the shadow relative to the shape's origin. Specified as a pair of fractional values of the shape dimensions, ranging from 50% to -50%. Default is "0,0".

[*Example:* This example is unchanged from above except for the addition of the origin attribute:

```
<v:shadow on="true" type="perspective"
  matrix="1.25,-2,,1.5,,.000001"
  offset="38pt,-6pt" origin="10%,-10%">
</v:shadow>
```



end example]

The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>type (Shadow Type)</td><td>

Specifies the type of shadow. Default is `single`. Allowed values are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td><code>single</code></td><td>Single shadow.</td></tr>
<tr><td><code>double</code></td><td>Double shadow. color2 and offset2 are used for the second shadow's color and offset.</td></tr>
<tr><td><code>perspective</code></td><td>Perspective shadow.</td></tr>
<tr><td><code>shaperelative</code></td><td>The shadow is created relative to the shape.</td></tr>
<tr><td><code>drawingrelative</code></td><td>The shadow is created relative to the drawing.</td></tr>
</table>

</td></tr>
</table>

Attributes	Description
	emboss — The shadow has an embossed look. The possible values for this attribute are defined by the ST_ShadowType simple type (§6.1.3.7).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Shadow">
   <attributeGroup ref="AG_Id"/>
   <attribute name="on" type="ST_TrueFalse" use="optional"/>
   <attribute name="type" type="ST_ShadowType" use="optional"/>
   <attribute name="obscured" type="ST_TrueFalse" use="optional"/>
   <attribute name="color" type="ST_ColorType" use="optional"/>
   <attribute name="opacity" type="xsd:string" use="optional"/>
   <attribute name="offset" type="xsd:string" use="optional"/>
   <attribute name="color2" type="ST_ColorType" use="optional"/>
   <attribute name="offset2" type="xsd:string" use="optional"/>
   <attribute name="origin" type="xsd:string" use="optional"/>
   <attribute name="matrix" type="xsd:string" use="optional"/>
</complexType>
```

6.1.2.19 shape (Shape Definition)

This element is used to describe a shape, the core object in VML. This element may appear by itself or within a group element (§6.1.2.7). If a shapetype element (§6.1.2.20) is referenced using the type attribute, any attributes specified in the shape will override those found in the shapetype.

[*Example:*

```
<v:shape style="position:absolute;top:50;left:20;width:50;height:50"
  path="m 0,0 l 0,1000 1000,1000 1000,0 x e">
  <v:shadow on="true" type="perspective"
    matrix="1.25,-2,,1.5,,.000001" offset="38pt,-6pt"/>
</v:shape>

<v:shape style="position:absolute;top:50;left:20;width:50;height:50"
  fillcolor="yellow" path="m 0,0 l 0,1000 1000,1000 1000,0 x e">
  <v:extrusion on="true" lightposition="0,-2000,10000"/>
</v:shape>
```

end example]

Parent Elements
background (§2.2.1); group (§6.1.2.7); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23)

Child Elements	Subclause
anchorlock (Anchor Location Is Locked)	§6.3.2.1
borderbottom (Bottom Border)	§6.3.2.2
borderleft (Left Border)	§6.3.2.3
borderright (Right Border)	§6.3.2.4
bordertop (Top Border)	§6.3.2.5
callout (Callout)	§6.2.2.2
ClientData (Attached Object Data)	§6.4.2.12
clippath (Shape Clipping Path)	§6.2.2.3
extrusion (3D Extrusion)	§6.2.2.10
fill (Shape Fill Properties)	§6.1.2.5
formulas (Set of Formulas)	§6.1.2.6
handles (Set of Handles)	§6.1.2.9
imagedata (Image Data)	§6.1.2.11
ink (Ink)	§6.2.2.14
iscomment (Ink Annotation Flag)	§6.5.2.1
lock (Shape Protections)	§6.2.2.17
path (Shape Path)	§6.1.2.14
shadow (Shadow Effect)	§6.1.2.18
signatureline (Digital Signature Line)	§6.2.2.29
skew (Skew Transform)	§6.2.2.30
stroke (Line Stroke Settings)	§6.1.2.21
textbox (Text Box)	§6.1.2.22
textdata (VML Diagram Text)	§6.5.2.2
textpath (Text Layout Path)	§6.1.2.23
wrap (Text Wrapping)	§6.3.2.6

Attributes	Description
adj (Adjustment Parameters)	Specifies a comma-delimited list of parameters, or adjustment values, used to define values for a parameterized formula. Values may be omitted. There can be up to 8 adjust

Attributes	Description

	values. Each value is referenced using # followed by a number corresponding to the zero-based index for that value in the list of adjustment values. For example, #2 references the second value in the adj list.
	
	[*Example:* The following shape uses formulas to define a simple rectangle. The adj values are referenced by the eqn attribute of the f element (§6.1.2.4) and in turn referenced by the path element (§6.1.2.14).
	
	`<v:shape`
	`coordorigin="0 0" coordsize="200 200"`
	`style="position:relative;top:30;left:30;width:20;height:20"`
	`adj="1, 1, 1, 200, 200, 200, 200, 1">`
	`<v:path v="m @0,@1 l @2,@3, @4,@5, @6,@7 x e"/>`
	`<v:formulas>`
	`<v:f eqn="val #0"/>`
	`<v:f eqn="val #1"/>`
	`<v:f eqn="val #2"/>`
	`<v:f eqn="val #3"/>`
	`<v:f eqn="val #4"/>`
	`<v:f eqn="val #5"/>`
	`<v:f eqn="val #6"/>`
	`<v:f eqn="val #7"/>`
	`</v:formulas>`
	`</v:shape>`
	
	This is the equivalent of:
	
	`<v:shape`
	`coordorigin="0 0" coordsize="200 200"`
	`style="position:relative;top:30;left:30;width:20;height:20"`
	`path="m 1,1 l 1,200, 200,200, 200,1 x e">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

allowincell (Allow in Table Cell)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies whether a shape can be placed in a table. Default is `false`.
	
	[*Example:*
	
	`<v:shape ... o:allowincell="true" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

Attributes	Description
allowoverlap (Allow Shape Overlap) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape can overlap another shape. Default is `true`. If `false`, the shape will shift left or right so as not to overlap another shape, similar to the behavior of the HTML float attribute. [*Example:* `<v:shape ... o:allowoverlap="false" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

alt (Alternate Text)	Specifies alternative text describing the graphical object. This text should provide a brief description of the shape for use by accessibility tools. Default is no value.
	
	[*Example:* The alt text describes the basic shape:
	
	`<v:shape ... fillcolor="red" alt="Red rectangle"></v:shape>`
	
	The alt text describes the contents of a shape displaying an image:
	
	`<v:shape ... alt="Picture of a sunset"></v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

borderbottomcolor (Bottom Border Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the bottom border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:borderbottomcolor="red" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
borderleftcolor (Border Left Color) Namespace: urn:schemas-	Specifies the left border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:borderleftcolor="red" ... >`

Attributes	Description
microsoft-com:office:office	</v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
borderrightcolor (Border Right Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the right border color of an inline shape. Default is no value. [*Example:* <v:shape ... o:borderrightcolor="red" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
bordertopcolor (Border Top Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the top border color of an inline shape. Default is no value. [*Example:* <v:shape ... o:bordertopcolor="red" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
bullet (Graphical Bullet) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the shape is a graphical bullet. Default is `false`. [*Example:* <v:shape ... o:bullet="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
button (Button Behavior Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape will exhibit button press behavior on click. Default is `false`. [*Example:* <v:shape ... o:button="true" ... > </v:shape> *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
bwmode (Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies how a shape will render for black-and-white output devices. When a shape is printed on a black-and-white printer or displayed in a black-and-white view in an application, several options are possible. Default is auto, which will use o:bwnormal for normal black-and-white rendering and o:bwpure for pure black-and-white rendering. bwnormal and bwpure are subordinate to bwmode. If bwmode is "auto" then the value for bwnormal or bwpure is used depending on what the output format is. An application may define for itself what, if any, difference there is between normal B&W and pure B&W. For example, normal B&W might allow greyscale and pure B&W might not. [*Example:* This shape renders in grayscale in a black-and-white environment: `<v:shape ... o:bwmode="grayscale" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
bwnormal (Normal Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies the black-and-white mode for normal black-and-white output devices. Default is auto. [*Example:* This shape renders in a pale grayscale in a normal black-and-white environment: `<v:shape ... o:bwmode="auto" o:bwnormal="lightgrayscale" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
bwpure (Pure Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies the black-and-white mode for pure black-and-white output devices. Default is auto. [*Example:* This shape renders in high contrast when in a pure black-and-white environment: `<v:shape ... o:bwmode="auto" o:bwpure="highcontrast" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type

Attributes	Description
	(§6.2.3.2).

chromakey (Image Transparency Color)	Specifies a color value that will be transparent and show anything behind the shape. Default is no value.
	
	[*Example:*
	
	`<v:image ... chromakey="white" ...>`
	`</v:image>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).

class (CSS Reference)	Specifies a reference to the definition of a CSS style. Default is no value.
	
	[*Example:* The snippets below are equivalent:
	
	`...`
	`.narrowstyle {width:50;height:100}`
	`...`
	`<v:shape ... class="narrowstyle"`
	`style="top:1;left:1">`
	`</v:shape>`
	
	`<v:shape ... style="top:1;left:1;`
	`width:50;height:100">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

clip (Clipping Toggle)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies that the clipping region is active. This is used in conjunction with the clippath (§6.2.2.3) element to create a clipping region.
	
	[*Example:*
	
	`<v:shape ... o:clip="true">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

cliptowrap (Clip to	Specifies that the clipping region for the shape aligns with the wrapping polygon that

Attributes	Description

Wrapping Polygon)
 	
Namespace: urn:schemas-microsoft-com:office:office
	tightly surrounds the entire shape (essentially, that the shape shall not be drawn beyond its wrapping polygon's extents – if it does, the shape shall be clipped). Default is `false`.
	
	[*Example:*
	
	`<v:shape ... o:cliptowrap="true">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

connectortype (Shape Connector Type)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies the type of connector used for joining shapes. Default is `straight`.
	
	[*Example:*
	
	`<v:shape ... o:connectortype="elbow" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_ConnectorType simple type (§6.2.3.6).

coordorigin (Coordinate Space Origin)	Specifies the coordinate of the top left corner of the shape's coordinate space. This determines the position of the (0,0) coordinate space origin within the shape's bounding box. Default is "0,0", which places the (0,0) origin at the top left corner of the bounding box.
	
	This affects shape properties that specify coordinate positions, such as the path attribute. Thus a path can be defined against a generic (0,0) origin and the coordorigin value translates the entire path within the shape's bounding space.
	
	[*Example:* The horizontal and vertical coordinate space ranges from -100 to +100 because the coordinate space (coordsize) is 200 by 200 and the top left coordinate is (-100,-100). The (0,0) origin lies at the center of the shape's bounding box, as evidenced by the position of the shape's path within the coordinate space:
	
	`<v:shape ... coordsize="200,200"`
	`coordorigin="-100,-100"`
	`path="m 0,0 l 0,50, 50,50, 50,0 x e">`
	`</v:shape>`
	
	 

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>coordsize
(Coordinate Space Size)</td><td>Specifies the size of the shape's coordinate space in coordinate units. Default is "1000,1000".

The physical size of a coordinate unit length is determined by both the size of the coordinate space (coordsize) and the size of the shape (style width and height). The coordsize attribute defines the number of horizontal and vertical subdivisions into which the shape's bounding box is divided. The combination of coordsize and style width/height effective scales the shape anisotropically.

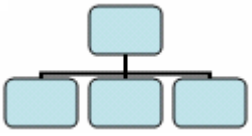
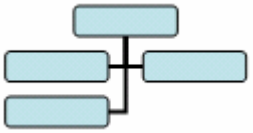
[Example: The path is 50 units wide and tall, which is 25% of the size of the coordinate space:

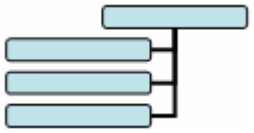
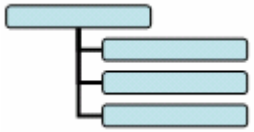
<pre><v:shape ... coordsize="200,200"
  coordorigin="-100,-100"
  path="m 0,0 l 0,50, 50,50, 50,0 x e">
</v:shape></pre>
 

end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>dgmlayout
(Diagram Node Layout Identifier)

Namespace:
urn:schemas-microsoft-com:office:office</td><td>Specifies the type of automatic layout to apply to the diagram node. This is only meaningful for organization charts. Valid values for organization charts are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Top-down, centered layout.
  </td></tr>
<tr><td>1</td><td>Hanging, both sides layout.
  </td></tr>
<tr><td>2</td><td>Hanging, right side layout.</td></tr>
</table>
</td></tr>
</table>

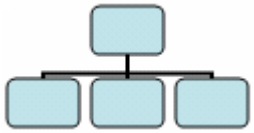
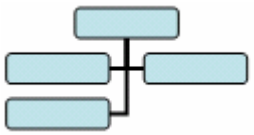
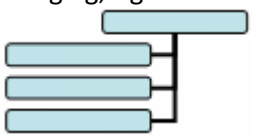
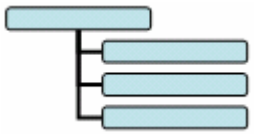
<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>  </td></tr>
<tr><td>3</td><td>Hanging, left side layout.
  </td></tr>
</table>

[Example:

<pre><v:shape ... dgmlayout="1">
</v:shape></pre>
end example]

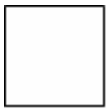
The possible values for this attribute are defined by the XML Schema integer datatype.</td></tr>
<tr><td>dgmlayoutmru (Diagram Node Recent Layout Identifier)

Namespace: urn:schemas-microsoft-com:office:office</td><td>Specifies the type of automatic layout most recently used on the child elements of the diagram node. This is only meaningful for organization charts. Valid values for organization charts are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Top-down, centered layout.
  </td></tr>
<tr><td>1</td><td>Hanging, both sides layout.
  </td></tr>
<tr><td>2</td><td>Hanging, right side layout.
  </td></tr>
<tr><td>3</td><td>Hanging, left side layout.
  </td></tr>
</table>

[Example:</td></tr>
</table>

Attributes	Description
	<v:shape ... dgmlayout="1"> </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema integer datatype.
dgmnodekind (Diagram Node Identifier) Namespace: urn:schemas-microsoft-com:office:office	Specifies an optional, application-specific parameter that is intended to be used by the application to tag different types of nodes in a diagram. [*Example:* <v:shape ... dgmnodekind="1"> </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema integer datatype.
doubleclicknotify (Double-click Notification Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies that an event message is sent when a shape is double-clicked. Default is `false`. [*Example:* <v:shape ... o:doubleclicknotify="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
equationxml (Storage for Alternate Math Content)	Specifies alternate XML markup which may be used to rehydrate an equation using the Office Open XML Math syntax. The actual format of the contents of this attribute are application-defined, but shall contain Office Open XML Math as well as any application-specific content. The XML markup stored in this attribute shall be escaped as needed to contain only those characters legal in an attribute value. The possible values for this attribute are defined by the XML Schema string datatype.
fillcolor (Fill Color)	Specifies the color to use for the fill. Default is `white`. If the fill element (§6.1.2.5) is present, its color attribute takes precedence. Colors are typically specified as either a named color, such as `red`, or six hexadecimal digits representing the red, green and blue values of the color, such as `#00FF30`. Full details are specified in the simple type description. [*Example:* This shape is red if its fill is visible:

Attributes	Description
	<v:shape ... fillcolor="red" ... > </v:shape> This is equivalent to: <v:shape ... fillcolor="#ff0000" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).
filled (Shape Fill Toggle)	Specifies whether the closed path will be filled. Default is `true`. This attribute is overridden by the fill on attribute. [*Example:* <v:shape ... filled="f" fillcolor="red" ...> </v:shape>  *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
forcedash (Force Dashed Outline) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a dashed outline is used to draw a shape when a shape has no line or fill. Default is `false`. Used by PresentationML placeholders to draw a dashed outline when there is no line and no fill for a shape. [*Example:* <v:shape ... o:forcedash="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

Attributes	Description

gfxdata (Encoded Package)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies a base-64 encoded package as deifined in Part 2 of this Standard that contains DrawingML content. The contents of this package are application-defined, but the contents of the package shall use the Parts defined by this Standard whenever possible. [*Rationale:* This attribute allows an application to use VML to represent graphical content while still persisting DrawingML for consuming applications that support DrawingML. For example, a diagram stored within this attribute would have the four parts defined for a DrawingML diagram, as well as any number of application-defined parts and relationships. *end ationale*]
	
	[*Example:* A DrawingML object is encoded in the gfxdata attribute, leaving VML to handle the visual display:
	
	`<v:shape id="Diagram 1" o:spid="_x0000_i1025"`
	`type="#_x0000_t75" style="width:446.25pt;height:252pt;`
	`visibility:visible" o:gfxdata="UEsDBBQABgAIAAAAIQDIu8KcTQE…">`
	`<v:imagedata r:id="rId4" o:title=""/>`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema base64Binary datatype.

hr (Horizontal Rule Toggle)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies that a shape is a horizontal rule. Default is `false`.
	
	[*Example:*
	
	`<v:shape ... o:hr="true" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

hralign (Horizontal Rule Alignment)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies the alignment of a horizontal rule. Default is `left`.
	
	[*Example:*
	
	`<v:shape ... o:hralign="center" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_HrAlign simple type (§6.2.3.14).

href (Hyperlink	Specifies a hyperlink URL target for the shape. Default is no value.

Attributes	Description
Target)	[*Example:* `<v:shape ... href="http://www.openxmlformats.org" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
hrnoshade (Horizontal Rule 3D Shading Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies that the horizontal rule does not have 3-D shading. Default is `false`. [*Example:* `<v:shape ... o:hrnoshade="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
hrpct (Horizontal Rule Length Percentage) Namespace: urn:schemas-microsoft-com:office:office	Specifies the length of a horizontal rule as a percentage of page width. Default is 0. [*Example:* `<v:shape ... o:hrpct="85" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema float datatype.
hrstd (Horizontal Rule Standard Display Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape is displayed as a standard horizontal rule. Only applies if hr is `true`. Default is `false`. [*Example:* `<v:shape ... o:hrstd="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
id (Unique Identifier)	Specifies a unique identifier that can be used to reference a VML object. Default is no value.

Attributes	Description
	[*Example:* `<v:shape ... id="myShape" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
insetmode (Text Inset Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the application calculates the internal text margin instead of using the inset attribute. Default is `custom`. This attribute is only meaningful for text boxes. [*Example:* `<v:shape ... o:insetmode="auto" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_InsetMode simple type (§6.2.3.15).
insetpen (Inset Border From Path)	Specifies that the border shall be displayed inside of the path defining the shape, rather than along the path (the default border placement), or outside of the path as might be done with an image. [*Example:* `<v:shape ... insetpen="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
ole (Embedded Object Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the shape is an embedded object. Default is `false`. [*Example:* `<v:shape ... o:ole="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalseBlank simple type (§6.2.3.24).

Attributes	Description
oleicon (Embedded Object Icon Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether an embedded object will be displayed as an icon. Default is `false`. [*Example:* `<v:shape ... o:oleicon="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
oned (Shape Handle Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the extra handles of a shape are hidden. If `true`, hides all shape handles except the top left and bottom right; that is, the same handles that are used for a straight line segment. Default is `false`. [*Example:* `<v:shape ... o:oned="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
opacity (Fill Color Opacity)	Specifies the opacity of the primary fill color. Default is 1.0. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8. [*Example:* The red color is 25% opaque: `<v:fill type="gradient" color="red" color2="blue" opacity=".25"></v:fill>`  `opacity="1"`  `opacity=".25"` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
path (Edge Path)	Specifies the line that makes up the edges of a shape. See the v attribute of the path

Attributes	Description
	element (§6.1.2.14) for a full description. The possible values for this attribute are defined by the XML Schema string datatype.
preferrelative (Relative Resize Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the original size of an object is saved after reformatting. Default is `false`. If `true`, the original size of the object is stored and all resizing is based on a percentage of that original size. Otherwise, each resizing will reset the scale to 100%. [*Example:* `<v:shape ... o:preferrelative="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
print (Print Toggle)	Specifies whether the shape will be printed. Default is `true`. [*Example:* `<v:shape ... print="false" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
regroupid (Regroup ID) Namespace: urn:schemas-microsoft-com:office:office	Specifies a previous group for a shape. An ID number is used to identify groups of shapes that are no longer grouped. This allows shapes to be regrouped programmatically. [*Example:* The shape was part of a group identified by the ID 040754: `<v:shape ... o:regroupid="040754" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema integer datatype.
spid (Optional String) Namespace: urn:schemas-microsoft-com:office:office	Specifies an optional string that an application may use to Identify the particular shape. Default is no value. The possible values for this attribute are defined by the XML Schema string datatype.

Attributes	Description
spt (Optional Number) Namespace: urn:schemas-microsoft-com:office:office	Specifies an optional number that an application may use to associate the particular shape with a defined shape type. Default is 0. The possible values for this attribute are defined by the XML Schema float datatype.

strokecolor (Shape Stroke Color)	Specifies the primary color of the brush to use to stroke the path of the shape. Default is `black`. The color attribute of the stroke element (§6.1.2.21) overrides this. Colors are typically specified as either a named color, such as `red`, or six hexadecimal digits representing the red, green and blue values of the color, such as `#00FF30`. Full details are specified in the simple type description.
	
	[*Example:*
	
	`<v:shape ... strokecolor="red" ...>`
	`</v:shape>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).

stroked (Shape Stroke Toggle)	Specifies whether the path defining the shape is stroked with a solid line. The stroke element (§6.1.2.21) defines other strokes. The on attribute of the stroke element overrides this attribute. Default is `true`.
	
	[*Example:*
	
	`<v:shape ... fillcolor="red"`
	`  stroked="false" strokecolor="blue"...>`
	`</v:shape>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).

strokeweight (Shape Stroke	Specifies the width of the brush to use to stroke the path. Default is 1 point. If a number is given without units, the emu is used. The weight attribute of the stroke element

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>Weight)</td><td>(§6.1.2.21) overrides this attribute.

[Example:

<pre><v:shape ... strokeweight="3pt" ... >
</v:shape></pre>
 

end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>style (Shape Styling Properties)</td><td>Specifies the CSS2 styling properties of the shape. This uses the syntax described in the "Visual formatting model" of the Cascading Style Sheets, Level 2 specification, a Recommendation of the World Wide Web Consortium available here: http://www.w3.org/TR/REC-CSS2. Full descriptions of each property are not repeated here, but the VML treatment of each property is defined. Allowed properties include:

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>flip</td><td>Specifies that the orientation of a shape is flipped. Default is no value. Allowed values are:

• x - Flip along the y-axis, reversing the x-coordinates.
• y - Flip along the x-axis, reversing the y-coordinates.
• xy - Flip along both the y- and x-axis.
• yx - Flip along both the x- and y-axis.</td></tr>
<tr><td>height</td><td>Specifies the height of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are:

• auto - Default position of an element in the flow of the page.
• <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed.
• <percentage>- Value expressed as a percentage of the parent object's height.</td></tr>
<tr><td>left</td><td>Specifies the position of the left of the containing block of the shape relative to the element left of it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are:</td></tr>
</table>
</td></tr>
</table>

Attributes	Description
	• auto - Default position of an element in the flow of the page. • <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
margin-bottom	Specifies the position of the bottom of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
margin-left	Specifies the position of the left of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
margin-right	Specifies the position of the right of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.

Attributes	Description	
	margin-top	Specifies the position of the top of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	mso-position-horizontal	Specifies the horizontal positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute • left • center • right • inside • outside
	mso-position-horizontal-relative	Specifies relative horizontal position data for objects in WordprocessingML documents. This modifies the mso-position-horizontal property. Default is text. Allowed values are: • margin • page • text • char
	mso-position-vertical	Specifies the vertical positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute • top • center • bottom • inside • outside

Attributes	Description	
	mso-position-vertical-relative	Specifies relative vertical position data for objects in WordprocessingML documents. This modifies the mso-position-vertical property. Default is text. Allowed values are: • margin • page • text • line
	mso-wrap-distance-bottom	Specifies the distance from the bottom of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
	mso-wrap-distance-left	Specifies the distance from the left side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
	mso-wrap-distance-right	Specifies the distance from the right side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
	mso-wrap-distance-top	Specifies the distance from the top of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
	mso-wrap-edited	Specifies whether the wrap coordinates were customized by the user. If the wrap coordinates are generated by an editor, this property is true; otherwise they were customized by a user. Default is false.
	mso-wrap-style	Specifies the wrapping mode for text in shapes in WordprocessingML documents. Default is square. Allowed values are: • square - Wraps text inside the shape in a square. • none - Text does not wrap.
	position	Specifies the type of positioning used to place an element. Default is static. When the element is contained inside a group, this property must be absolute. Allowed values are:

Attributes	Description	
		• `static` - The element is positioned according to the normal flow of the page. The `top` and `left` properties are ignored. If the object is anchored inline, this value is used. • `absolute` - The element is positioned relative to the parent, using the `top` and `left` properties. • `relative` - The element is positioned according to the normal flow of the page, but the `top` and `left` properties are used. The overlap of overlapping elements is governed by the `z-index` property.
	`rotation`	Specifies the angle that a shape is rotated, in degrees. Default is 0. Positive angles are clockwise.
	`top`	Specifies the position of the top of the containing block of the shape relative to the element above it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	`visibility`	Specifies whether a shape is displayed. Only `inherit` and `hidden` are used; any other values are mapped to `inherit`. Default is `inherit`. Allowed values are: • `hidden` - The shape is not visible, but is still part of the flow of the objects in the browser. Mouse events are not processed. • `inherit` - The visibility state is inherited from the parent of the shape.
	`width`	Specifies the width of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed.

Attributes	Description
	<table><tr><td></td><td>• <percentage>- Value expressed as a percentage of the parent object's width.</td></tr><tr><td>z-index</td><td>Specifies the display order of overlapping shapes. Default is 0. This property shall not be used for shapes anchored inline. Allowed values are: • auto - Uses the order that the shapes appear in the page, bottom to top. • <order>- A number that represents the stacking precedence. Shapes with higher numbers are placed on top of those with lower numbers. Negative numbers are allowed.</td></tr></table> The following properties are only used by the textbox element (§6.1.2.22): <table><tr><th>Property</th><th>Description</th></tr><tr><td>direction</td><td>Specifies the direction of the text in the textbox. Default is ltr. This property is superceded by the mso-direction-alt property if that is specified. Allowed values are: • ltr - Test is displayed left-to-right. • rtl - Test is displayed right-to-left.</td></tr><tr><td>layout-flow</td><td>Determines the flow of the text layout in a textbox. Default is horizontal. Allowed values are: • horizontal - Text is displayed horizontally. • vertical - Text is displayed vertically. • vertical-ideographic - Ideographic text is displayed vertically. • horizontal-ideographic - Ideographic text is displayed horizontally.</td></tr><tr><td>mso-direction-alt</td><td>Specifies an alternate direction for text in textboxes. Overrides the direction property. The only allowed value is context.</td></tr><tr><td>mso-fit-shape-to-text</td><td>Specifies whether the shape stretches to fit the text in the textbox. Default is false.</td></tr><tr><td>mso-fit-text-to-shape</td><td>Specifies whether the text stretches to fit the textbox. Default is false.</td></tr><tr><td>mso-layout-</td><td>Specifies the alternate layout flow for text in textboxes. This</td></tr></table>

Attributes	Description
	flow-alt: property is used instead of layout-flow when the layout flow is from bottom to top for non-ideographic languages. Its only value is bottom-to-top.
	mso-next-textbox: Specifies the ID of the next textbox in a series. Used to keep track of a set of linked textboxes. Default is no value.
	mso-rotate: Specifies a specific rotation value for text in a textbox. Default is 0. Allowed values are: • 0 • 90 • 180 • -90
	mso-text-scale: Specifies the scaling factor for fitting text to shapes. Default is 0. This property is only used if mso-fit-text-to-shape is true.
	v-text-anchor: Specifies the vertical anchoring of text in a textbox. Default is top. The alignment of a text anchor only becomes evident if mso-fit-text-to-shape is false. This property is different from the vertical-align CSS property, which is used for ideographic languages. Allowed values are: • top • middle • bottom • top-center • middle-center • bottom-center • top-baseline • bottom-baseline • top-center-baseline • bottom-center-baseline

The following properties are only used by the textpath element (§6.1.2.23):

Property	Description
font	Specifies a compound value of font settings. Default is no value. The values are the same as those of the CSS font property. The order of definitions in the string is: font-style, font-variant, font-weight, font-size, line-height, font-family.
font-family	Specifies the family of the font. Default is no value. The values are the same as those of the CSS font-family property.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>font-size</td><td>Specifies the size of the font. Default is no value. The font size is defined in points. The values are the same as those of the CSS font-size property.</td></tr>
<tr><td>font-style</td><td>Specifies the amount of slant for a font. Default is normal. The values are the same as those of the CSS font-style property. Allowed values are:
normal
italic
oblique - Treated the same as italic.</td></tr>
<tr><td>font-variant</td><td>Specifies the variant style of a font. Default is normal. The values are the same as those of the CSS font-variant property. Allowed values are:
normal
small-caps</td></tr>
<tr><td>font-weight</td><td>Specifies the thickness of the letters of the font. Default is normal. The values are the same as those of the CSS font-weight property. Allowed values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>normal
lighter
100
200
300
400</td><td>Treated as non-bold.</td></tr>
<tr><td>bold
bolder
500
600
700
800
900</td><td>Treated as bold.</td></tr>
</table>
</td></tr>
<tr><td>mso-text-</td><td>Specifies whether a shadow is applied to the text on a text path.</td></tr>
</table>
</td></tr>
</table>

Attributes	Description
	shadow: Default is false.
	text-decoration: Specifies the style of text decoration. Default is none. The values are the same as those of the CSS text-decoration property. Allowed values are: • none • underline • overline • line-through • blink
	v-rotate-letters: Specifies whether the letters of the text are rotated counterclockwise by 90 degrees. Default is false.
	v-same-letter-heights: Specifies whether all letters will be the same height regardless of initial case. If true, the lowercase letters are stretched to the height of the uppercase letters. Default is false.
	v-text-align: Specifies the alignment of text. Default is left. Allowed values are: • left • right • center • justify • letter-justify - Distributes the extra space between the letters. • stretch-justify - Stretches the letters to fill in the space.
	v-text-kern: Specifies whether kerning is turned on. Default is false.
	v-text-reverse: Specifies whether the layout order of rows is reversed. Default is false. This is used for vertical text layout.
	v-text-spacing-mode: Specifies the mode for letter spacing. Default is tightening. This property determines whether space will be removed between each letter (tightening) or added between each letter (tracking). The amount of letter spacing change is defined by the v-text-spacing property. Allowed values are: • tightening • tracking
	v-text-spacing: Specifies the amount of spacing for text in 100ths of single line spacing. Default is 100.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>The line (§6.1.2.12), polyline (§6.1.2.15) and curve (§6.1.2.3) elements ignore the following properties:

• top
• left
• width
• height

The following properties are not inherited by an element that references a shapetype element (§6.1.2.20) via the id attribute:

• flip
• height
• left
• margin-left
• margin-top
• position
• rotation
• top
• visibility
• width
• z-index

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>target (Hyperlink Display Target)</td><td>Specifies a frame or window that a URL is displayed in. Default is no value. Allowed values are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td><targetname></td><td>String containing the name of the frame or window in which to load the document.</td></tr>
<tr><td>_blank</td><td>Specifies that the linked document is loaded into a new blank window. This window is not named.</td></tr>
<tr><td>_media</td><td>Specifies that the linked document is loaded into the Media Bar. Available in Microsoft® Internet Explorer 6 or later.</td></tr>
<tr><td>_parent</td><td>Specifies that the linked document is loaded into the immediate parent of the document containing the link.</td></tr>
<tr><td>_search</td><td>Specifies that the linked document is loaded into the browser's search pane. Available in Microsoft Internet Explorer 5 or greater.</td></tr>
<tr><td>_self</td><td>Specifies that the linked document is loaded into the window in which the link was clicked (the active window).</td></tr>
<tr><td>_top</td><td>Specifies that the linked document is loaded into the topmost window.</td></tr>
</table>
</td></tr>
</table>

Attributes	Description

	[*Example:*
	
	`<v:shape ...`
	`  href="http://www.openxmlformats.org"`
	`  target="_self" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

title (Shape Title)	Specifies the text displayed when the mouse pointer moves over the shape. Default is no value.
	
	[*Example:*
	
	`<v:shape ... title="tooltip" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

type (Shape Type Reference)	Specifies a reference to a shapetype ID that describes the standard path, fill and stroke properties of a shape. Properties specified in the shape override the shapetype properties. Default is no value.
	
	[*Example:* The following example defines a shapetype that is a simple rectangle and an actual shape instance that uses it and overrides the fill color.
	
	`<v:shapetype id="mytype"`
	`fillcolor="red" strokecolor="blue"`
	`coordorigin="0 0" coordsize="200 200"`
	`path="m 0,0 l 0,200, 200,200, 200,0 x e"/>`
	`</v:shapetype>`
	
	`<v:shape id="shape02" type="#mytype"`
	`fillcolor="green"`
	`style="position:relative;top:1;left:1;width:20;height:20">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

userdrawn (Exists In Master Slide)	Specifies whether the user has added the shape to a master slide. Default is `false`. Used by PresentationML.

Attributes	Description
Namespace: urn:schemas-microsoft-com:office:office	[*Example:* `<v:shape ... o:userdrawn="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
userhidden (Hide Script Anchors) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a script anchor is hidden. Default is `false`. If `true`, script anchors stay hidden even if the shape is otherwise visible. A script anchor is the visual representation of a script that when displayed in an application. [*Example:* `<v:shape ... o:userhidden="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
wrapcoords (Shape Bounding Polygon)	Specifies the bounding polygon that surrounds a shape. This is specified using a comma-delimited list of x and y coordinates: "x1,y1,x2,y2,x3,y3,..." This is used when text is tightly wrapped around a shape. Default is no value until the mso-wrap-mode style attribute is set to `tight` or `through`. [*Example:* `<v:shape ... wrapcoords="0,0 0,200, 200,200, 200,0" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Shape">
   <choice maxOccurs="unbounded">
      <group ref="EG_ShapeElements"/>
      <element ref="o:ink"/>
      <element ref="p:iscomment"/>
   </choice>
   <attributeGroup ref="AG_AllCoreAttributes"/>
   <attributeGroup ref="AG_AllShapeAttributes"/>
   <attributeGroup ref="AG_Type"/>
   <attributeGroup ref="AG_Adj"/>
   <attributeGroup ref="AG_Path"/>
   <attribute ref="o:gfxdata"/>
   <attribute name="equationxml" type="xsd:string" use="optional"/>
</complexType>
```

6.1.2.20 shapetype (Shape Template)

This element defines a shape template that can be used to create other shapes. Shapetype is identical to the shape element (§6.1.2.19) except it cannot reference another shapetype element. The type attribute may not be used with shapetype. Attributes defined in the shape override any that appear in the shapetype. CSS positioning attributes (such as `top`, `width`, `z-index`, `rotation`, `flip`) are not passed to a shape from a shapetype. To use this element, create a shapetype with a specific id attribute. Then create a shape and reference the shapetype's id using the type attribute.

[*Example:*

```
<v:shapetype id="mytype" fillcolor="silver" strokecolor="blue">
  <v:path v="m 0,0 l 0,1000, 1000,1000, 1000,0 x e"/>
  <v:fill on="true" type="gradient" color2="navy" angle="-45"/>
</v:shapetype>
<v:shape type="#mytype"
  style="position:absolute;top:10;left:10;width:50;height:50"/>
<v:shape type="#mytype" fillcolor="teal"
  style="position:absolute;top:10;left:75;width:75;height:50"/>
<v:shape type="#mytype"
  style="position:absolute;top:10;left:165;width:50;height:50">
  <v:fill type="solid"/>
</v:shape>
<v:shape type="#mytype" path="m 500,0 l 1000,1000 0,1000 x e"
style="position:absolute;top:10;left:230;width:50;height:50"/>
```

end example]

Parent Elements
background (§2.2.1); group (§6.1.2.7); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23)

Child Elements	Subclause
anchorlock (Anchor Location Is Locked)	§6.3.2.1
borderbottom (Bottom Border)	§6.3.2.2
borderleft (Left Border)	§6.3.2.3
borderright (Right Border)	§6.3.2.4
bordertop (Top Border)	§6.3.2.5
callout (Callout)	§6.2.2.2
ClientData (Attached Object Data)	§6.4.2.12
clippath (Shape Clipping Path)	§6.2.2.3
complex (Complex)	§6.2.2.7
extrusion (3D Extrusion)	§6.2.2.10
fill (Shape Fill Properties)	§6.1.2.5
formulas (Set of Formulas)	§6.1.2.6
handles (Set of Handles)	§6.1.2.9
imagedata (Image Data)	§6.1.2.11
lock (Shape Protections)	§6.2.2.17
path (Shape Path)	§6.1.2.14
shadow (Shadow Effect)	§6.1.2.18
signatureline (Digital Signature Line)	§6.2.2.29
skew (Skew Transform)	§6.2.2.30
stroke (Line Stroke Settings)	§6.1.2.21
textbox (Text Box)	§6.1.2.22
textdata (VML Diagram Text)	§6.5.2.2
textpath (Text Layout Path)	§6.1.2.23
wrap (Text Wrapping)	§6.3.2.6

Attributes	Description
adj (Adjustment Parameters)	Specifies a comma-delimited list of parameters, or adjustment values, used to define values for a parameterized formula. Values may be omitted. There can be up to 8 adjust values. Each value is referenced using # followed by a number corresponding to the zero-based index for that value in the list of adjustment values. For example, #2 references the second value in the adj list.

Attributes	Description

	[*Example:* The following shape uses formulas to define a simple rectangle. The adj values are referenced by the eqn attribute of the f element (§6.1.2.4) and in turn referenced by the path element (§6.1.2.14).
	
	`<v:shape`
	`coordorigin="0 0" coordsize="200 200"`
	`style="position:relative;top:30;left:30;width:20;height:20"`
	`adj="1, 1, 1, 200, 200, 200, 200, 1">`
	`<v:path v="m @0,@1 l @2,@3, @4,@5, @6,@7 x e"/>`
	`<v:formulas>`
	`<v:f eqn="val #0"/>`
	`<v:f eqn="val #1"/>`
	`<v:f eqn="val #2"/>`
	`<v:f eqn="val #3"/>`
	`<v:f eqn="val #4"/>`
	`<v:f eqn="val #5"/>`
	`<v:f eqn="val #6"/>`
	`<v:f eqn="val #7"/>`
	`</v:formulas>`
	`</v:shape>`
	
	This is the equivalent of:
	
	`<v:shape`
	`coordorigin="0 0" coordsize="200 200"`
	`style="position:relative;top:30;left:30;width:20;height:20"`
	`path="m 1,1 l 1,200, 200,200, 200,1 x e">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

allowincell (Allow in Table Cell)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies whether a shape can be placed in a table. Default is `false`.
	
	[*Example:*
	
	`<v:shape ... o:allowincell="true" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

allowoverlap (Allow Shape	Specifies whether a shape can overlap another shape. Default is `true`. If `false`, the shape will shift left or right so as not to overlap another shape, similar to the behavior of

Attributes	Description
Overlap) Namespace: urn:schemas-microsoft-com:office:office	the HTML float attribute. [*Example:* `<v:shape ... o:allowoverlap="false" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

alt (Alternate Text)	Specifies alternative text describing the graphical object. This text should provide a brief description of the shape for use by accessibility tools. Default is no value.
	
	[*Example:* The alt text describes the basic shape:
	
	`<v:shape ... fillcolor="red" alt="Red rectangle"></v:shape>`
	
	The alt text describes the contents of a shape displaying an image:
	
	`<v:shape ... alt="Picture of a sunset"></v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

borderbottomcolor (Bottom Border Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the bottom border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:borderbottomcolor="red" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
borderleftcolor (Border Left Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the left border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:borderleftcolor="red" ... ></v:shape>`

Attributes	Description
	end example] The possible values for this attribute are defined by the XML Schema string datatype.
borderrightcolor (Border Right Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the right border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:borderrightcolor="red" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
bordertopcolor (Border Top Color) Namespace: urn:schemas-microsoft-com:office:office	Specifies the top border color of an inline shape. Default is no value. [*Example:* `<v:shape ... o:bordertopcolor="red" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
bullet (Graphical Bullet) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the shape is a graphical bullet. Default is `false`. [*Example:* `<v:shape ... o:bullet="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
button (Button Behavior Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a shape will exhibit button press behavior on click. Default is `false`. [*Example:* `<v:shape ... o:button="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

Attributes	Description
bwmode (Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies how a shape will render for black-and-white output devices. When a shape is printed on a black-and-white printer or displayed in a black-and-white view in an application, several options are possible. Default is auto, which will use o:bwnormal for normal black-and-white rendering and o:bwpure for pure black-and-white rendering. bwnormal and bwpure are subordinate to bwmode. If bwmode is "auto" then the value for bwnormal or bwpure is used depending on what the output format is. An application may define for itself what, if any, difference there is between normal B&W and pure B&W. For example, normal B&W might allow greyscale and pure B&W might not. [*Example:* This shape renders in grayscale in a black-and-white environment: `<v:shape ... o:bwmode="grayscale" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
bwnormal (Normal Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies the black-and-white mode for normal black-and-white output devices. Default is auto. [*Example:* This shape renders in a pale grayscale in a normal black-and-white environment: `<v:shape ... o:bwmode="auto" o:bwnormal="lightgrayscale" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
bwpure (Pure Black-and-White Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies the black-and-white mode for pure black-and-white output devices. Default is auto. [*Example:* This shape renders in high contrast when in a pure black-and-white environment: `<v:shape ... o:bwmode="auto" o:bwpure="highcontrast" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_BWMode simple type (§6.2.3.2).
chromakey (Image	Specifies a color value that will be transparent and show anything behind the shape.

Attributes	Description

Transparency Color)	Default is no value.
	
	[*Example:*
	
	`<v:image ... chromakey="white" ...>`
	`</v:image>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).

class (CSS Reference)	Specifies a reference to the definition of a CSS style. Default is no value.
	
	[*Example:* The snippets below are equivalent:
	
	`...`
	`.narrowstyle {width:50;height:100}`
	`...`
	`<v:shape ... class="narrowstyle"`
	`style="top:1;left:1">`
	`</v:shape>`
	
	`<v:shape ... style="top:1;left:1;`
	`width:50;height:100">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

clip (Clipping Toggle)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies that the clipping region is active. This is used in conjunction with the clippath (§6.2.2.3) element to create a clipping region.
	
	[*Example:*
	
	`<v:shape ... o:clip="true">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

cliptowrap (Clip to Wrapping Polygon)	Specifies that the clipping region for the shape aligns with the wrapping polygon that tightly surrounds the entire shape (essentially, that the shape shall not be drawn beyond its wrapping polygon's extents – if it does, the shape shall be clipped). Default is `false`.

Attributes	Description

Namespace: urn:schemas-microsoft-com:office:office	[*Example:*
	
	`<v:shape ... o:cliptowrap="true">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

connectortype (Shape Connector Type)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies the type of connector used for joining shapes. Default is `straight`.
	
	[*Example:*
	
	`<v:shape ... o:connectortype="elbow" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_ConnectorType simple type (§6.2.3.6).

coordorigin (Coordinate Space Origin)	Specifies the coordinate of the top left corner of the shape's coordinate space. This determines the position of the (0,0) coordinate space origin within the shape's bounding box. Default is "0,0", which places the (0,0) origin at the top left corner of the bounding box.
	
	This affects shape properties that specify coordinate positions, such as the path attribute. Thus a path can be defined against a generic (0,0) origin and the coordorigin value translates the entire path within the shape's bounding space.
	
	[*Example:* The horizontal and vertical coordinate space ranges from -100 to +100 because the coordinate space (coordsize) is 200 by 200 and the top left coordinate is (-100,-100). The (0,0) origin lies at the center of the shape's bounding box, as evidenced by the position of the shape's path within the coordinate space:
	
	`<v:shape ... coordsize="200,200"`
	`coordorigin="-100,-100"`
	`path="m 0,0 l 0,50, 50,50, 50,0 x e">`
	`</v:shape>`
	
	 
	
	end example]

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>coordsize
(Coordinate Space Size)</td><td>Specifies the size of the shape's coordinate space in coordinate units. Default is "1000,1000".

The physical size of a coordinate unit length is determined by both the size of the coordinate space (coordsize) and the size of the shape (style width and height). The coordsize attribute defines the number of horizontal and vertical subdivisions into which the shape's bounding box is divided. The combination of coordsize and style width/height effective scales the shape anisotropically.

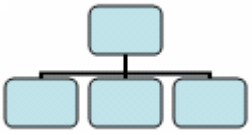
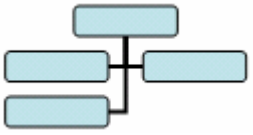
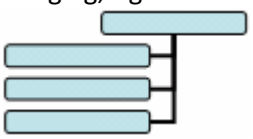
[Example: The path is 50 units wide and tall, which is 25% of the size of the coordinate space:

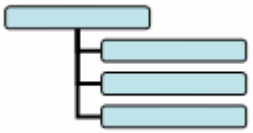
<pre><v:shape ... coordsize="200,200"
  coordorigin="-100,-100"
  path="m 0,0 l 0,50, 50,50, 50,0 x e">
</v:shape></pre>
 

end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>dgmlayout
(Diagram Node Layout Identifier)

Namespace: urn:schemas-microsoft-com:office:office</td><td>Specifies the type of automatic layout to apply to the diagram node. This is only meaningful for organization charts. Valid values for organization charts are:

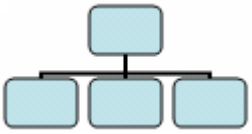
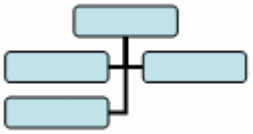
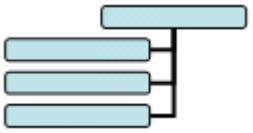
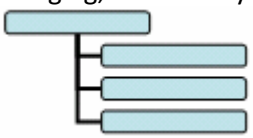
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Top-down, centered layout.
  </td></tr>
<tr><td>1</td><td>Hanging, both sides layout.
  </td></tr>
<tr><td>2</td><td>Hanging, right side layout.
  </td></tr>
<tr><td>3</td><td>Hanging, left side layout.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>  </td></tr>
</table>

[Example:

<pre><v:shape ... dgmlayout="1">
</v:shape></pre>
end example]

The possible values for this attribute are defined by the XML Schema integer datatype.
</td></tr>
<tr><td>dgmlayoutmru (Diagram Node Recent Layout Identifier)

Namespace: urn:schemas-microsoft-com:office:office</td><td>
Specifies the type of automatic layout most recently used on the child elements of the diagram node. This is only meaningful for organization charts. Valid values for organization charts are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Top-down, centered layout.
  </td></tr>
<tr><td>1</td><td>Hanging, both sides layout.
  </td></tr>
<tr><td>2</td><td>Hanging, right side layout.
  </td></tr>
<tr><td>3</td><td>Hanging, left side layout.
  </td></tr>
</table>
[Example:

<pre><v:shape ... dgmlayout="1">
</v:shape></pre>
end example]
</td></tr>
</table>

Attributes	Description
	The possible values for this attribute are defined by the XML Schema integer datatype.

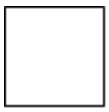
dgmnodekind (Diagram Node Identifier)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies an optional, application-specific parameter that is intended to be used by the application to tag different types of nodes in a diagram.
	
	[*Example:*
	
	`<v:shape ... dgmnodekind="1">`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema integer datatype.

doubleclicknotify (Double-click Notification Toggle)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies that an event message is sent when a shape is double-clicked. Default is `false`.
	
	[*Example:*
	
	`<v:shape ... o:doubleclicknotify="true" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

fillcolor (Fill Color)	Specifies the color to use for the fill. Default is `white`. If the fill element (§6.1.2.5) is present, its color attribute takes precedence. Colors are typically specified as either a named color, such as `red`, or six hexadecimal digits representing the red, green and blue values of the color, such as `#00FF30`. Full details are specified in the simple type description.
	
	[*Example:* This shape is red if its fill is visible:
	
	`<v:shape ... fillcolor="red" ... >`
	`</v:shape>`
	
	This is equivalent to:
	
	`<v:shape ... fillcolor="#ff0000" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).

filled (Shape Fill	Specifies whether the closed path will be filled. Default is `true`. This attribute is

Attributes	Description

Toggle)	overridden by the fill on attribute.
	
	[*Example:*
	
	`<v:shape ... filled="f"`
	`  fillcolor="red" ...>`
	`</v:shape>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).

forcedash (Force Dashed Outline)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies whether a dashed outline is used to draw a shape when a shape has no line or fill. Default is `false`.
	
	Used by PresentationML placeholders to draw a dashed outline when there is no line and no fill for a shape.
	
	[*Example:*
	
	`<v:shape ... o:forcedash="true" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

hr (Horizontal Rule Toggle)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies that a shape is a horizontal rule. Default is `false`.
	
	[*Example:*
	
	`<v:shape ... o:hr="true" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

hralign (Horizontal Rule Alignment)	Specifies the alignment of a horizontal rule. Default is `left`. [*Example:*

Attributes	Description
Namespace: urn:schemas-microsoft-com:office:office	<v:shape ... o:hralign="center" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_HrAlign simple type (§6.2.3.14).
href (Hyperlink Target)	Specifies a hyperlink URL target for the shape. Default is no value. [*Example:* <v:shape ... href="http://www.openxmlformats.org" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
hrnoshade (Horizontal Rule 3D Shading Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies that the horizontal rule does not have 3-D shading. Default is `false`. [*Example:* <v:shape ... o:hrnoshade="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
hrpct (Horizontal Rule Length Percentage) Namespace: urn:schemas-microsoft-com:office:office	Specifies the length of a horizontal rule as a percentage of page width. Default is 0. [*Example:* <v:shape ... o:hrpct="85" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema float datatype.
hrstd (Horizontal Rule Standard Display Toggle) Namespace: urn:schemas-	Specifies whether a shape is displayed as a standard horizontal rule. Only applies if hr is `true`. Default is `false`. [*Example:* <v:shape ... o:hrstd="true" ... >

Attributes	Description
microsoft-com:office:office	</v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
id (Unique Identifier)	Specifies a unique identifier that can be used to reference a VML object. Default is no value. [*Example:* <v:shape ... id="myShape" ... > </v:shape> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
insetmode (Text Inset Mode) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the application calculates the internal text margin instead of using the inset attribute. Default is custom. This attribute is only meaningful for text boxes. [*Example:* <v:shape ... o:insetmode="auto" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_InsetMode simple type (§6.2.3.15).
insetpen (Inset Border From Path)	Specifies that the border shall be displayed inside of the path defining the shape, rather than along the path (the default border placement), or outside of the path as might be done with an image. [*Example:* <v:shape ... insetpen="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
master (Master Element Toggle)	Specifies whether the shapetype is a master element. If true, it is rendered by the rendering engine. Default is false.

Attributes	Description
Namespace: urn:schemas-microsoft-com:office:office	The possible values for this attribute are defined by the XML Schema string datatype.
ole (Embedded Object Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the shape is an embedded object. Default is `false`. [*Example:* `<v:shape ... o:ole="true" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalseBlank simple type (§6.2.3.24).
oleicon (Embedded Object Icon Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether an embedded object will be displayed as an icon. Default is `false`. [*Example:* `<v:shape ... o:oleicon="true" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
oned (Shape Handle Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the extra handles of a shape are hidden. If `true`, hides all shape handles except the top left and bottom right; that is, the same handles that are used for a straight line segment. Default is `false`. [*Example:* `<v:shape ... o:oned="true" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
opacity (Fill Color Opacity)	Specifies the opacity of the primary fill color. Default is 1.0. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied. For example, a value of "52429f" represents 52429/65536 or 0.8. [*Example:* The red color is 25% opaque:

Attributes	Description
	<v:fill type="gradient" color="red" color2="blue" opacity=".25"> </v:fill>  opacity="1"  opacity=".25" *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
path (Edge Path)	Specifies the line that makes up the edges of a shape. See the v attribute of the path element (§6.1.2.14) for a full description. The possible values for this attribute are defined by the XML Schema string datatype.
preferrelative (Relative Resize Toggle) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether the original size of an object is saved after reformatting. Default is false. If true, the original size of the object is stored and all resizing is based on a percentage of that original size. Otherwise, each resizing will reset the scale to 100%. [*Example:* <v:shape ... o:preferrelative="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
print (Print Toggle)	Specifies whether the shape will be printed. Default is true. [*Example:* <v:shape ... print="false" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
regroupid (Regroup ID)	Specifies a previous group for a shape. An ID number is used to identify groups of shapes that are no longer grouped. This allows shapes to be regrouped programmatically.

Attributes	Description
Namespace: urn:schemas-microsoft-com:office:office	[*Example:* The shape was part of a group identified by the ID 040754: `<v:shape ... o:regroupid="040754" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema integer datatype.
spid (Optional String) Namespace: urn:schemas-microsoft-com:office:office	Specifies an optional string that an application may use to Identify the particular shape. Default is no value. The possible values for this attribute are defined by the XML Schema string datatype.
spt (Optional Number) Namespace: urn:schemas-microsoft-com:office:office	Specifies an optional number that an application may use to associate the particular shape with a defined shape type. Default is 0. The possible values for this attribute are defined by the XML Schema float datatype.
strokecolor (Shape Stroke Color)	Specifies the primary color of the brush to use to stroke the path of the shape. Default is `black`. The color attribute of the stroke element (§6.1.2.21) overrides this. Colors are typically specified as either a named color, such as `red`, or six hexadecimal digits representing the red, green and blue values of the color, such as `#00FF30`. Full details are specified in the simple type description. [*Example:* `<v:shape ... strokecolor="red" ...> </v:shape>`  *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).
stroked (Shape Stroke Toggle)	Specifies whether the path defining the shape is stroked with a solid line. The stroke element (§6.1.2.21) defines other strokes. The on attribute of the stroke element overrides this attribute. Default is `true`.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>[Example:

<pre><v:shape ... fillcolor="red"
  stroked="false" strokecolor="blue"...>
</v:shape></pre>
 

end example]

The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).</td></tr>
<tr><td>strokeweight (Shape Stroke Weight)</td><td>Specifies the width of the brush to use to stroke the path. Default is 1 point. If a number is given without units, the emu is used. The weight attribute of the stroke element (§6.1.2.21) overrides this attribute.

[Example:

<pre><v:shape ... strokeweight="3pt" ... >
</v:shape></pre>
 

end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>style (Shape Styling Properties)</td><td>Specifies the CSS2 styling properties of the shape. This uses the syntax described in the "Visual formatting model" of the Cascading Style Sheets, Level 2 specification, a Recommendation of the World Wide Web Consortium available here: http://www.w3.org/TR/REC-CSS2. Full descriptions of each property are not repeated here, but the VML treatment of each property is defined. Allowed properties include:

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>flip</td><td>Specifies that the orientation of a shape is flipped. Default is no value. Allowed values are:

• x - Flip along the y-axis, reversing the x-coordinates.
• y - Flip along the x-axis, reversing the y-coordinates.
• xy - Flip along both the y- and x-axis.
• yx - Flip along both the x- and y-axis.</td></tr>
</table>
</td></tr>
</table>

Attributes	Description	
	height	Specifies the height of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	left	Specifies the position of the left of the containing block of the shape relative to the element left of it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are: • auto - Default position of an element in the flow of the page. • <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	margin-bottom	Specifies the position of the bottom of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	margin-left	Specifies the position of the left of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm,

Attributes	Description
	mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	`margin-right` Specifies the position of the right of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	`margin-top` Specifies the position of the top of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	`mso-position-horizontal` Specifies the horizontal positioning data for objects in WordprocessingML documents. Default is `absolute`. Allowed values are: • `absolute` • `left` • `center` • `right` • `inside` • `outside`
	`mso-position-horizontal-relative` Specifies relative horizontal position data for objects in WordprocessingML documents. This modifies the `mso-position-horizontal` property. Default is `text`. Allowed values are:

Attributes	Description
	• margin • page • text • char
mso-position-vertical	Specifies the vertical positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute • top • center • bottom • inside • outside
mso-position-vertical-relative	Specifies relative vertical position data for objects in WordprocessingML documents. This modifies the mso-position-vertical property. Default is text. Allowed values are: • margin • page • text • line
mso-wrap-distance-bottom	Specifies the distance from the bottom of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-distance-left	Specifies the distance from the left side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-distance-right	Specifies the distance from the right side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-distance-top	Specifies the distance from the top of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not

Attributes	Description
	change the origin.
mso-wrap-edited	Specifies whether the wrap coordinates were customized by the user. If the wrap coordinates are generated by an editor, this property is true; otherwise they were customized by a user. Default is false.
mso-wrap-style	Specifies the wrapping mode for text in shapes in WordprocessingML documents. Default is square. Allowed values are: • square - Wraps text inside the shape in a square. • none - Text does not wrap.
position	Specifies the type of positioning used to place an element. Default is static. When the element is contained inside a group, this property must be absolute. Allowed values are: • static - The element is positioned according to the normal flow of the page. The top and left properties are ignored. If the object is anchored inline, this value is used. • absolute - The element is positioned relative to the parent, using the top and left properties. • relative - The element is positioned according to the normal flow of the page, but the top and left properties are used. The overlap of overlapping elements is governed by the z-index property.
rotation	Specifies the angle that a shape is rotated, in degrees. Default is 0. Positive angles are clockwise.
top	Specifies the position of the top of the containing block of the shape relative to the element above it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
visibility	Specifies whether a shape is displayed. Only inherit and hidden are used; any other values are mapped to inherit. Default is inherit. Allowed values are:

Attributes	Description
	<table><tr><td></td><td> - hidden - The shape is not visible, but is still part of the flow of the objects in the browser. Mouse events are not processed. - inherit - The visibility state is inherited from the parent of the shape. </td></tr><tr><td>width</td><td>Specifies the width of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: - auto - Default position of an element in the flow of the page. <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. <percentage>- Value expressed as a percentage of the parent object's width. </td></tr><tr><td>z-index</td><td>Specifies the display order of overlapping shapes. Default is 0. This property shall not be used for shapes anchored inline. Allowed values are: - auto - Uses the order that the shapes appear in the page, bottom to top. <order>- A number that represents the stacking precedence. Shapes with higher numbers are placed on top of those with lower numbers. Negative numbers are allowed. </td></tr></table> The following properties are only used by the textbox element (§6.1.2.22): <table><tr><th>Property</th><th>Description</th></tr><tr><td>direction</td><td>Specifies the direction of the text in the textbox. Default is ltr. This property is superceded by the mso-direction-alt property if that is specified. Allowed values are: - ltr - Test is displayed left-to-right. - rtl - Test is displayed right-to-left. </td></tr><tr><td>layout-flow</td><td>Determines the flow of the text layout in a textbox. Default is horizontal. Allowed values are: - horizontal - Text is displayed horizontally. - vertical - Text is displayed vertically. </td></tr></table>

Attributes	Description

	• `vertical-ideographic` - Ideographic text is displayed vertically. • `horizontal-ideographic` - Ideographic text is displayed horizontally.
`mso-direction-alt`	Specifies an alternate direction for text in textboxes. Overrides the `direction` property. The only allowed value is `context`.
`mso-fit-shape-to-text`	Specifies whether the shape stretches to fit the text in the textbox. Default is `false`.
`mso-fit-text-to-shape`	Specifies whether the text stretches to fit the textbox. Default is `false`.
`mso-layout-flow-alt`	Specifies the alternate layout flow for text in textboxes. This property is used instead of `layout-flow` when the layout flow is from bottom to top for non-ideographic languages. Its only value is `bottom-to-top`.
`mso-next-textbox`	Specifies the ID of the next textbox in a series. Used to keep track of a set of linked textboxes. Default is no value.
`mso-rotate`	Specifies a specific rotation value for text in a textbox. Default is 0. Allowed values are: • 0 • 90 • 180 • -90
`mso-text-scale`	Specifies the scaling factor for fitting text to shapes. Default is 0. This property is only used if `mso-fit-text-to-shape` is `true`.
`v-text-anchor`	Specifies the vertical anchoring of text in a textbox. Default is `top`. The alignment of a text anchor only becomes evident if `mso-fit-text-to-shape` is `false`. This property is different from the `vertical-align` CSS property, which is used for ideographic languages. Allowed values are: • `top` • `middle` • `bottom` • `top-center` • `middle-center` • `bottom-center` • `top-baseline`

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>

bottom-baseline
top-center-baseline
bottom-center-baseline

</td></tr>
</table>
The following properties are only used by the textpath element (§6.1.2.23):
<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>font</td><td>Specifies a compound value of font settings. Default is no value. The values are the same as those of the CSS font property. The order of definitions in the string is: font-style, font-variant, font-weight, font-size, line-height, font-family.</td></tr>
<tr><td>font-family</td><td>Specifies the family of the font. Default is no value. The values are the same as those of the CSS font-family property.</td></tr>
<tr><td>font-size</td><td>Specifies the size of the font. Default is no value. The font size is defined in points. The values are the same as those of the CSS font-size property.</td></tr>
<tr><td>font-style</td><td>Specifies the amount of slant for a font. Default is normal. The values are the same as those of the CSS font-style property. Allowed values are:
normal
italic
oblique - Treated the same as italic.</td></tr>
<tr><td>font-variant</td><td>Specifies the variant style of a font. Default is normal. The values are the same as those of the CSS font-variant property. Allowed values are:
normal
small-caps</td></tr>
<tr><td>font-weight</td><td>Specifies the thickness of the letters of the font. Default is normal. The values are the same as those of the CSS font-weight property. Allowed values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>normal
lighter
100
200</td><td>Treated as non-bold.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>
<table>
<tr><td>300

400</td><td></td></tr>
<tr><td>bold

bolder

500

600

700

800

900</td><td>Treated as bold.</td></tr>
</table>
</td></tr>
<tr><td>mso-text-shadow</td><td>Specifies whether a shadow is applied to the text on a text path. Default is false.</td></tr>
<tr><td>text-decoration</td><td>Specifies the style of text decoration. Default is none. The values are the same as those of the CSS text-decoration property. Allowed values are:

• none
• underline
• overline
• line-through
• blink</td></tr>
<tr><td>v-rotate-letters</td><td>Specifies whether the letters of the text are rotated counterclockwise by 90 degrees. Default is false.</td></tr>
<tr><td>v-same-letter-heights</td><td>Specifies whether all letters will be the same height regardless of initial case. If true, the lowercase letters are stretched to the height of the uppercase letters. Default is false.</td></tr>
<tr><td>v-text-align</td><td>Specifies the alignment of text. Default is left. Allowed values are:

• left
• right
• center
• justify
• letter-justify - Distributes the extra space between the letters.
• stretch-justify - Stretches the letters to fill in the space.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>v-text-kern</td><td>Specifies whether kerning is turned on. Default is false.</td></tr>
<tr><td>v-text-reverse</td><td>Specifies whether the layout order of rows is reversed. Default is false. This is used for vertical text layout.</td></tr>
<tr><td>v-text-spacing-mode</td><td>Specifies the mode for letter spacing. Default is tightening. This property determines whether space will be removed between each letter (tightening) or added between each letter (tracking). The amount of letter spacing change is defined by the v-text-spacing property. Allowed values are:

• tightening
• tracking</td></tr>
<tr><td>v-text-spacing</td><td>Specifies the amount of spacing for text in 100ths of single line spacing. Default is 100.</td></tr>
</table>

The line (§6.1.2.12), polyline (§6.1.2.15) and curve (§6.1.2.3) elements ignore the following properties:

• top
• left
• width
• height

The following properties are not inherited by an element that references a shapetype element (§6.1.2.20) via the id attribute:

• flip
• height
• left
• margin-left
• margin-top
• position
• rotation
• top
• visibility
• width
• z-index

The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>target (Hyperlink Display Target)</td><td>Specifies a frame or window that a URL is displayed in. Default is no value. Allowed values are:

<table>
<tr><th>Value</th><th>Description</th></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td><targetname></td><td>String containing the name of the frame or window in which to load the document.</td></tr>
<tr><td>_blank</td><td>Specifies that the linked document is loaded into a new blank window. This window is not named.</td></tr>
<tr><td>_media</td><td>Specifies that the linked document is loaded into the Media Bar. Available in Microsoft® Internet Explorer 6 or later.</td></tr>
<tr><td>_parent</td><td>Specifies that the linked document is loaded into the immediate parent of the document containing the link.</td></tr>
<tr><td>_search</td><td>Specifies that the linked document is loaded into the browser's search pane. Available in Microsoft Internet Explorer 5 or greater.</td></tr>
<tr><td>_self</td><td>Specifies that the linked document is loaded into the window in which the link was clicked (the active window).</td></tr>
<tr><td>_top</td><td>Specifies that the linked document is loaded into the topmost window.</td></tr>
</table>
[Example:
<pre>
  <v:shape ...
    href="http://www.openxmlformats.org"
    target="_self" ... >
  </v:shape>
</pre>
end example]
The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>title (Shape Title)</td><td>
Specifies the text displayed when the mouse pointer moves over the shape. Default is no value.
[Example:
<pre>
  <v:shape ... title="tooltip" ... >
  </v:shape>
</pre>
end example]
The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>userdrawn (Exists In Master Slide)

Namespace: urn:schemas-microsoft-com:office:office</td><td>
Specifies whether the user has added the shape to a master slide. Default is <code>false</code>. Used by PresentationML.
[Example:
<pre>
  <v:shape ... o:userdrawn="true" ... >
  </v:shape>
</pre>
</td></tr>
</table>

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
userhidden (Hide Script Anchors) Namespace: urn:schemas-microsoft-com:office:office	Specifies whether a script anchor is hidden. Default is `false`. If `true`, script anchors stay hidden even if the shape is otherwise visible. A script anchor is the visual representation of a script that when displayed in an application. [*Example:* `<v:shape ... o:userhidden="true" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

wrapcoords (Shape Bounding Polygon)	Specifies the bounding polygon that surrounds a shape. This is specified using a comma-delimited list of x and y coordinates: "x1,y1,x2,y2,x3,y3,..." This is used when text is tightly wrapped around a shape. Default is no value until the mso-wrap-mode style attribute is set to `tight` or `through`.
	
	[*Example:*
	
	`<v:shape ...`
	`wrapcoords="0,0 0,200, 200,200, 200,0" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Shapetype">
   <sequence>
      <group ref="EG_ShapeElements" minOccurs="0" maxOccurs="unbounded"/>
      <element ref="o:complex" minOccurs="0"/>
   </sequence>
   <attributeGroup ref="AG_AllCoreAttributes"/>
   <attributeGroup ref="AG_AllShapeAttributes"/>
   <attributeGroup ref="AG_Adj"/>
   <attributeGroup ref="AG_Path"/>
   <attribute ref="o:master"/>
</complexType>
```

6.1.2.21 stroke (Line Stroke Settings)

This element describes how to draw the path if something beyond solid line with a solid color is desired. The attributes of the stroke element can be used to describe a powerful set of stroke properties. Extensions to the VML stroke definition are encoded as sub-elements of stroke.

[*Example:*

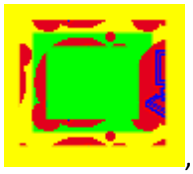
```
<v:polyline points="0pt,0pt,50pt,0pt,50pt,35pt,15pt,35pt,
  15pt,15pt,75pt,15pt">
  <v:stroke startarrow="classic" endarrow="classic"
    startarrowwidth="wide" endarrowwidth="wide" dashstyle="dashdot"
    weight="2pt" color="teal" linestyle="thinThin"/>
</v:polyline>
```

end example]

Parent Elements
arc (§6.1.2.1); background (§2.2.1); curve (§6.1.2.3); group (§6.1.2.7); image (§6.1.2.10); line (§6.1.2.12); object (§2.3.3.19); oval (§6.1.2.13); pict (§2.3.3.21); pict (§2.9.23); polyline (§6.1.2.15); rect (§6.1.2.16); roundrect (§6.1.2.17); shape (§6.1.2.19); shapedefaults (§6.2.2.27); shapetype (§6.1.2.20)

Child Elements	Subclause
bottom (Text Box Bottom Stroke)	§6.2.2.1
column (Text Box Interior Stroke)	§6.2.2.6
left (Text Box Left Stroke)	§6.2.2.15
right (Text Box Right Stroke)	§6.2.2.25
top (Text Box Top Stroke)	§6.2.2.31

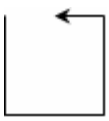
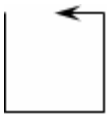
Attributes	Description
althref (Alternate Image Reference) Namespace: urn:schemas-microsoft-com:office:office	Specifies an alternate reference for an image in Macintosh PICT format. [*Example:* `<v:stroke ... althref="myimage.pcz" ... ></v:stroke>` *end example*]

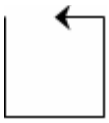
Attributes	Description
	The possible values for this attribute are defined by the XML Schema string datatype.
color (Stroke Color)	Specifies the stroke color. Overrides the strokecolor attribute of a shape. Default is `black`. See the fillcolor attribute for a list of supported named colors. [*Example:* The shape stroke is blue: `<v:shape ... strokecolor="red" ... >   <v:stroke color="blue"/> </v:shape>` *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).
color2 (Stroke Alternate Pattern Color)	Specifies a second color for strokes, used when filltype is `pattern`. Default is no value. When a pattern fill is used for the stroke, the stroke color is used in colored parts of the source image. The color2 defines an alternate color to use in place of black in the source image. [*Example:* This unusual example is intended to demonstrate how the image and colors interact to create a patterned stroke. The yellow background shows transparency. The non-square shape and square image create an effective offset. The heavy stroke weight shows more of the image. The green shape fill shows how the stroke is overlaid on the shape. `<v:background fillcolor="yellow"/> <v:shape style="width:60;height:50"   strokecolor="red" fillcolor="lime"   path="m 0,0 l 0,1000 1000,1000 1000,0 x e">   <v:stroke filltype="pattern" weight="10pt"   src="myimage.gif" color2="blue"/> </v:shape>`  , where myimage.gif is:  *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.1.3.1).
dashstyle (Stroke Dash Pattern)	Specifies the dot and dash pattern for a stroke. Default is `solid`. Pre-defined values are:

Attributes	Description

	• solid
	• shortdash
	• shortdot
	• shortdashdot
	• shortdashdotdot
	• dot
	• dash
	• longdash
	• dashdot
	• longdashdot
	• longdashdotdot
	
	A custom-defined dash pattern may also be specified using a series of numbers. These define the length of the dash (the drawn part of the stroke) and the length of the space between the dashes. The lengths are relative to the line width: a length of 1 is equal to the line width. The endcap style is applied to each dash but the arrow style is not. The string defines the length of the dash then the length of the space. This may be repeated to form complex dash styles. The string should always contain a pair of numbers; if it contains an odd number of numbers the last is disregarded. 0 implies a dot that is fourfold symmetrical (with round end caps, this is a circle).
	
	[*Example:*
	
	`<v:stroke dashstyle="0 2" weight="3pt" endcap="round"></v:stroke>`
	
	 
	
	`<v:stroke dashstyle="longdashdotdot" weight="2pt"></v:stroke>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

endarrow (Line End Arrowhead)	Specifies an arrowhead for the end of a line. Default is none. Note that the path must not be closed with the x command for the arrowhead to show. Allowed values are:

Attributes	Description
	• none • block • classic • diamond • oval • open [*Example:* `<v:stroke endarrow="classic"/>`  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowType simple type (§6.1.3.9).
endarrowlength (Line End Arrowhead Length)	Specifies the length of the arrowhead at the end of a line. Default is `medium`. Allowed values are: • short • medium • long [*Example:* `<v:stroke ... endarrowlength="long" ... />`  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowLength simple type (§6.1.3.8).
endarrowwidth (Line End Arrowhead Width)	Specifies the width of the arrowhead at the end of a line. Default is `medium`. Allowed values are: • narrow

Attributes	Description
	• medium • wide [*Example:* `<v:stroke ... endarrowwidth="wide" ... />`  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowWidth simple type (§6.1.3.10).
endcap (Line End Cap)	Specifies the cap style for the end of a stroke. Default is `flat`. Allowed values are: • flat • square • round [*Example:* `<v:stroke ... endcap="round" weight="10pt" ... />`  endcap="flat" endcap="square" endcap="round" *end example*] The possible values for this attribute are defined by the ST_StrokeEndCap simple type (§6.1.3.11).
filltype (Stroke Image Style)	Specifies the type of fill used for the background of a stroke. Default is `solid`. Allowed values are:

Attributes	Description

	• `solid` - The fill pattern is solid.
	• `tile` - The fill image is tiled.
	• `pattern` - The fill image is stretched to form a pattern.
	• `frame` - The fill image becomes a border for the shape.
	
	[*Example:*
	
	`<v:shape style="width:50;height:50"`
	`  strokecolor="red"`
	`  path="m 0,0 l 0,1000 1000,1000 1000,0 x e">`
	`  <v:stroke filltype="frame" weight="10pt"`
	`  src="border.gif"/>`
	`</v:shape>`
	
	  , where border.gif is: 
	
	end example]
	
	The possible values for this attribute are defined by the ST_FillType simple type (§6.1.3.5).

forcedash (Force Dashed Outline)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies whether a dashed outline is used to draw a shape when a shape has no line or fill. Default is `false`.
	
	Used by PresentationML placeholders to draw a dashed outline when there is no line and no fill for a shape.
	
	[*Example:*
	
	`<v:shape ... o:forcedash="true" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

href (Original Image Reference)
 	
Namespace: urn:schemas-microsoft-com:office:office
	Specifies the URL to the original image file. Used only if the picture has been linked and embedded. Default is no value.
	
	[*Example:*
	
	`<v:fill ... o:href="myimage.gif" ... >`
	`</v:fill>`

Attributes	Description
	end example] The possible values for this attribute are defined by the XML Schema string datatype.
id (Unique Identifier)	Specifies a unique identifier that can be used to reference a VML object. Default is no value. [*Example:* `<v:shape ... id="myShape" ... > </v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
id (Relationship) Namespace: .../officeDocument/2006/relationships	Specifies the relationship ID of the relationship to the image used for the stroke. The specified relationship shall be of type http://schemas.openxmlformats.org/officeDocument/2006/relationships/image or the document shall be considered non-conformant. [*Example*: The markup specifies the associated relationship part with relationship ID `rId10` contains the corresponding relationship information: `< ... r:id="rId10" />` *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

imagealignshape (Stoke Image Alignment)	Specifies the alignment of the stroke image. If `true`, the image is aligned with the shape. Otherwise, it is aligned with the containing scope. Default is `true`.
	
	[*Example:* The top position offset shifts the image alignment relative to the containing window:
	
	`<v:shape fillcolor="silver"`
	`style="top:20;width:50;height:50"`
	`path="m 0,0 l 0,1000 1000,1000 1000,0 x e">`
	`<v:stroke imagealignshape="false" weight="20pt"`
	`filltype="tile" src="myimage.gif"/>`
	`</v:shape>`

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
 imagealignshape="false"

 imagealignshape="false"

end example]

The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
</td></tr>
<tr><td>imageaspect (Stroke Image Aspect Ratio)</td><td>
Specifies how the stroke image aspect ratio is preserved. Default is ignore. Allowed values are:

<table>
<tr><td>Value</td><td>Description</td></tr>
<tr><td>ignore</td><td>Ignore aspect issues.</td></tr>
<tr><td>atleast</td><td>Image is at least as big as imagesize.</td></tr>
<tr><td>atmost</td><td>Image is no bigger than imagesize.</td></tr>
</table>

[Example:

<pre>
<v:stroke filltype="frame" weight="10pt"
  src="border.gif" imagealignshape="true"
  imageaspect="atleast">
</v:stroke>
</pre>
 imagealignshape="ignore"

 imagealignshape="atleast"

 imagealignshape="atmost"

end example]
</td></tr>
</table>

Attributes	Description
	The possible values for this attribute are defined by the ST_ImageAspect simple type (§6.1.3.6).
imagesize (Stroke Image Size)	Specifies the size of the image for the stroke. Default is the size of the image. [*Example:* `<v:stroke ... imagesize="10pt,10pt" ... />` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

insetpen (Inset Border From Path)	Specifies that the border shall be displayed inside of the path defining the shape, rather than along the path (the default border placement), or outside of the path as might be done with an image.
	
	[*Example:*
	
	`<v:shape ... insetpen="true" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).

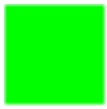
joinstyle (Line End Join Style)	Specifies the join style for line ends. Default is round.
	
	• round
	• bevel
	• miter
	
	[*Example:*
	
	`<v:polyline strokeweight="10pt" strokecolor="navy"`
	`points="10pt,10pt,50pt,50pt,90pt,10pt">`
	`<v:stroke joinstyle="bevel"/>`
	`</v:polyline>`
	
	 
	joinstyle="round"

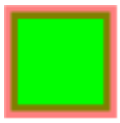
Attributes	Description
	 `joinstyle="bevel"`  `joinstyle="miter"` *end example*] The possible values for this attribute are defined by the ST_StrokeJoinStyle simple type (§6.1.3.12).

linestyle (Stroke Line Style)	Specifies the line style of the stroke. Default is `single`.
	
	• `single`
	• `thinThin`
	• `thinThick`
	• `thickThin`
	• `thickBetweenThin`
	
	[*Example:*
	
	`<v:stroke linestyle="thickThin" weight="5pt">`
	`</v:stroke>`
	 
	end example]
	
	The possible values for this attribute are defined by the ST_StrokeLineStyle simple type (§6.1.3.13).

miterlimit (Miter Joint Limit)	Specifies the smoothness of the miter joint, or the maximum distance between the inner point and outer point of a joint. This number is a multiple of the thickness of the line. Default is 8.
	
	[*Example:*
	
	`<v:stroke joinstyle="miter" weight="10pt"`
	`miterlimit="2">`
	`</v:stroke>`

Attributes	Description
	 *end example*] The possible values for this attribute are defined by the XML Schema decimal datatype.

on (Stroke Toggle)	Specifies whether the stroke is displayed. Default is `true`. This attribute overrides the shape's stroke attribute.
	
	[*Example:*
	
	`<v:rect style="width:50;height:50" stroked="true"`
	`  fillcolor="lime" strokecolor="red">`
	`  <v:stroke on="false" weight="5pt"/>`
	`</v:rect>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).

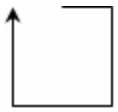
opacity (Stroke Opacity)	Specifies the amount of transparency of a stroke. Default is 1.0.
	
	[*Example:*
	
	`<v:rect style="width:50;height:50"`
	`  fillcolor="lime" strokecolor="red">`
	`  <v:stroke weight="5pt" opacity="50%"/>`
	`</v:rect>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

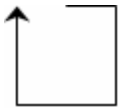
relid (Relationship to Part) Namespace: urn:schemas-	Specifies the relationship ID of the relationship to the image. The specified relationship shall be of type http://schemas.openxmlformats.org/officeDocument/2006/relationships/image or the document shall be considered non-conformant.

Attributes	Description

microsoft-com:office:office	[*Example*: The markup specifies the associated relationship part with relationship ID `rId10` contains the corresponding relationship information:
	
	`<v:stroke ... o:relid="rId10" ...>`
	`</v:stroke>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_RelationshipId simple type (§6.2.3.20).

src (Stroke Image Location)	Specifies the source image to load for a stroke fill. Default is no value.
	
	[*Example:*
	
	`<v:stroke ... src="myimage.gif" ... >`
	`</v:stroke>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

startarrow (Line Start Arrowhead)	Specifies an arrowhead for the start of a line. Default is none. Note that the path must not be closed with the x command for the arrowhead to show. Allowed values are: • `none` • `block` • `classic` • `diamond` • `oval` • `open` [*Example:* `<v:stroke startarrow="classic"/>`  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowType simple type (§6.1.3.9).
startarrowlength (Line Start	Specifies the length of the arrowhead at the start of a line. Default is `medium`. Allowed values are:

Attributes	Description
Arrowhead Length)	• short • medium • long [*Example:* `<v:stroke ... startarrowlength="long" ... />`  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowLength simple type (§6.1.3.8).
startarrowwidth (Line Start Arrowhead Width)	Specifies the width of the arrowhead at the start of a line. Default is `medium`. Allowed values are: • narrow • medium • wide [*Example:* `<v:stroke ... startarrowwidth="wide" ... />`  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowWidth simple type (§6.1.3.10).
title (Stroke Title) Namespace: urn:schemas-microsoft-com:office:office	Specifies the title of an embedded stroke image. This is typically set to the comment property of the image, which is often blank. [*Example:* `<v:fill ... o:title="alt text" ... ></v:fill>`

Attributes	Description
	end example] The possible values for this attribute are defined by the XML Schema string datatype.
weight (Stroke Weight)	Specifies the thickness of a stroke. Default is 1. This attribute overrides the shape's strokeweight attribute. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Stroke">
   <sequence>
      <element ref="o:left" minOccurs="0"/>
      <element ref="o:top" minOccurs="0"/>
      <element ref="o:right" minOccurs="0"/>
      <element ref="o:bottom" minOccurs="0"/>
      <element ref="o:column" minOccurs="0"/>
   </sequence>
   <attributeGroup ref="AG_Id"/>
   <attributeGroup ref="AG_StrokeAttributes"/>
</complexType>
```

6.1.2.22 textbox (Text Box)

This element is used to define text that appears inside the shape. This text may contain rich formatting and is rendered to fit inside the textboxrect defined by the path element (§6.1.2.14).

[*Example:*

```
<v:shape style="width=200;height=200" coordsize="400,400"
  fillcolor="yellow" strokecolor="maroon"
  path="m 119,0 l 148,86 238,86 166,140 192,226 119,175 46,226
  72,140 0,86 90,86 x e">
  <v:textbox inset="32pt,35pt,,">VML</v:textbox>
</v:shape>
```

end example]

Parent Elements
arc (§6.1.2.1); background (§2.2.1); curve (§6.1.2.3); group (§6.1.2.7); image (§6.1.2.10); line (§6.1.2.12);

Parent Elements
object (§2.3.3.19); oval (§6.1.2.13); pict (§2.3.3.21); pict (§2.9.23); polyline (§6.1.2.15); rect (§6.1.2.16); roundrect (§6.1.2.17); shape (§6.1.2.19); shapedefaults (§6.2.2.27); shapetype (§6.1.2.20)

Child Elements	Subclause
Any element from the empty namespace	n/a
Any element from the empty namespace	n/a
txbxContent (Rich Text Box Content Container)	§2.17.1.1

Attributes	Description
id (Unique Identifier)	Specifies a unique identifier that can be used to reference a VML object. Default is no value. [*Example:* `<v:shape ... id="myShape" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
inset (Text Box Inset)	Specifies inner margin values for textbox text. Default is "0.1in, 0.05in, 0.1in, 0.05in". Missing values are set to the default. This is used if insetmode is `custom`. The internal text margin value is specified as a string containing four values, each separated by commas or spaces. The values measure inset from the left, top, right, and bottom edges of the box specified by the textboxrect attribute of the path element (§6.1.2.14). [*Example:* The text is set toward the lower right of a small square: `<v:textbox inset="20pt,30pt,10pt,10pt">VML</v:textbox>`  *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>insetmode (Text Inset Mode)

Namespace: urn:schemas-microsoft-com:office:office</td><td>Specifies whether the application calculates the internal text margin instead of using the inset attribute. Default is <code>custom</code>.

[Example:

<pre><v:textbox ... o:insetmode="auto" ... >
</v:textbox></pre>
end example]

The possible values for this attribute are defined by the ST_InsetMode simple type (§6.2.3.15).</td></tr>
<tr><td>singleclick (Text Box Single-Click Selection Toggle)

Namespace: urn:schemas-microsoft-com:office:office</td><td>Specifies whether text is selectable with a single click. Default is <code>false</code>.

The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).</td></tr>
<tr><td>style (Shape Styling Properties)</td><td>Specifies the CSS2 styling properties of the shape. This uses the syntax described in the "Visual formatting model" of the Cascading Style Sheets, Level 2 specification, a Recommendation of the World Wide Web Consortium available here: http://www.w3.org/TR/REC-CSS2. Full descriptions of each property are not repeated here, but the VML treatment of each property is defined. Allowed properties include:

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td><code>flip</code></td><td>Specifies that the orientation of a shape is flipped. Default is no value. Allowed values are:

• x - Flip along the y-axis, reversing the x-coordinates.
• y - Flip along the x-axis, reversing the y-coordinates.
• xy - Flip along both the y- and x-axis.
• yx - Flip along both the x- and y-axis.</td></tr>
<tr><td><code>height</code></td><td>Specifies the height of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are:

• <code>auto</code> - Default position of an element in the flow of the page.
• <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed.
• <percentage>- Value expressed as a percentage of the</td></tr>
</table>
</td></tr>
</table>

Attributes	Description
	parent object's height.
left	Specifies the position of the left of the containing block of the shape relative to the element left of it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are: • auto - Default position of an element in the flow of the page. • <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
margin-bottom	Specifies the position of the bottom of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
margin-left	Specifies the position of the left of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
margin-right	Specifies the position of the right of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the

Attributes	Description	
		page. • <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	margin-top	Specifies the position of the top of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	mso-position-horizontal	Specifies the horizontal positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute • left • center • right • inside • outside
	mso-position-horizontal-relative	Specifies relative horizontal position data for objects in WordprocessingML documents. This modifies the mso-position-horizontal property. Default is text. Allowed values are: • margin • page • text • char
	mso-position-vertical	Specifies the vertical positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute

Attributes	Description
	• top • center • bottom • inside • outside
mso-position-vertical-relative	Specifies relative vertical position data for objects in WordprocessingML documents. This modifies the mso-position-vertical property. Default is text. Allowed values are: • margin • page • text • line
mso-wrap-distance-bottom	Specifies the distance from the bottom of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-distance-left	Specifies the distance from the left side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-distance-right	Specifies the distance from the right side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-distance-top	Specifies the distance from the top of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-edited	Specifies whether the wrap coordinates were customized by the user. If the wrap coordinates are generated by an editor, this property is true; otherwise they were customized by a user. Default is false.
mso-wrap-style	Specifies the wrapping mode for text in shapes in WordprocessingML documents. Default is square. Allowed values are:

Attributes	Description	
		• square - Wraps text inside the shape in a square. • none - Text does not wrap.
	position	Specifies the type of positioning used to place an element. Default is static. When the element is contained inside a group, this property must be absolute. Allowed values are: • static - The element is positioned according to the normal flow of the page. The top and left properties are ignored. If the object is anchored inline, this value is used. • absolute - The element is positioned relative to the parent, using the top and left properties. • relative - The element is positioned according to the normal flow of the page, but the top and left properties are used. The overlap of overlapping elements is governed by the z-index property.
	rotation	Specifies the angle that a shape is rotated, in degrees. Default is 0. Positive angles are clockwise.
	top	Specifies the position of the top of the containing block of the shape relative to the element above it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	visibility	Specifies whether a shape is displayed. Only inherit and hidden are used; any other values are mapped to inherit. Default is inherit. Allowed values are: • hidden - The shape is not visible, but is still part of the flow of the objects in the browser. Mouse events are not processed. • inherit - The visibility state is inherited from the parent of the shape.
	width	Specifies the width of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are:

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>

auto - Default position of an element in the flow of the page.
<units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed.
<percentage>- Value expressed as a percentage of the parent object's width.

</td></tr>
<tr><td>z-index</td><td>Specifies the display order of overlapping shapes. Default is 0. This property shall not be used for shapes anchored inline. Allowed values are:
auto - Uses the order that the shapes appear in the page, bottom to top.
<order>- A number that represents the stacking precedence. Shapes with higher numbers are placed on top of those with lower numbers. Negative numbers are allowed.
</td></tr>
</table>
The following properties are only used by the textbox element (§6.1.2.22):
<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>direction</td><td>Specifies the direction of the text in the textbox. Default is ltr. This property is superceded by the mso-direction-alt property if that is specified. Allowed values are:
ltr - Test is displayed left-to-right.
rtl - Test is displayed right-to-left.
</td></tr>
<tr><td>layout-flow</td><td>Determines the flow of the text layout in a textbox. Default is horizontal. Allowed values are:
horizontal - Text is displayed horizontally.
vertical - Text is displayed vertically.
vertical-ideographic - Ideographic text is displayed vertically.
horizontal-ideographic - Ideographic text is displayed horizontally.
</td></tr>
<tr><td>mso-direction-alt</td><td>Specifies an alternate direction for text in textboxes. Overrides the direction property. The only allowed value is context.</td></tr>
<tr><td>mso-fit-shape-</td><td>Specifies whether the shape stretches to fit the text in the</td></tr>
</table>
</td></tr>
</table>

Attributes	Description
	to-text: textbox. Default is false. mso-fit-text-to-shape: Specifies whether the text stretches to fit the textbox. Default is false. mso-layout-flow-alt: Specifies the alternate layout flow for text in textboxes. This property is used instead of layout-flow when the layout flow is from bottom to top for non-ideographic languages. Its only value is bottom-to-top. mso-next-textbox: Specifies the ID of the next textbox in a series. Used to keep track of a set of linked textboxes. Default is no value. mso-rotate: Specifies a specific rotation value for text in a textbox. Default is 0. Allowed values are: • 0 • 90 • 180 • -90 mso-text-scale: Specifies the scaling factor for fitting text to shapes. Default is 0. This property is only used if mso-fit-text-to-shape is true. v-text-anchor: Specifies the vertical anchoring of text in a textbox. Default is top. The alignment of a text anchor only becomes evident if mso-fit-text-to-shape is false. This property is different from the vertical-align CSS property, which is used for ideographic languages. Allowed values are: • top • middle • bottom • top-center • middle-center • bottom-center • top-baseline • bottom-baseline • top-center-baseline • bottom-center-baseline The following properties are only used by the textpath element (§6.1.2.23): font: Specifies a compound value of font settings. Default is no value. The values are the same as those of the CSS font property. The

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>order of definitions in the string is: font-style, font-variant, font-weight, font-size, line-height, font-family.</td></tr>
<tr><td>font-family</td><td>Specifies the family of the font. Default is no value. The values are the same as those of the CSS font-family property.</td></tr>
<tr><td>font-size</td><td>Specifies the size of the font. Default is no value. The font size is defined in points. The values are the same as those of the CSS font-size property.</td></tr>
<tr><td>font-style</td><td>Specifies the amount of slant for a font. Default is normal. The values are the same as those of the CSS font-style property. Allowed values are:
normal
italic
oblique - Treated the same as italic.</td></tr>
<tr><td>font-variant</td><td>Specifies the variant style of a font. Default is normal. The values are the same as those of the CSS font-variant property. Allowed values are:
normal
small-caps</td></tr>
<tr><td>font-weight</td><td>Specifies the thickness of the letters of the font. Default is normal. The values are the same as those of the CSS font-weight property. Allowed values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>normal
lighter
100
200
300
400</td><td>Treated as non-bold.</td></tr>
<tr><td>bold
bolder
500
600</td><td>Treated as bold.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Attributes	Description
	700 800 900
	mso-text-shadow: Specifies whether a shadow is applied to the text on a text path. Default is false.
	text-decoration: Specifies the style of text decoration. Default is none. The values are the same as those of the CSS text-decoration property. Allowed values are: • none • underline • overline • line-through • blink
	v-rotate-letters: Specifies whether the letters of the text are rotated counterclockwise by 90 degrees. Default is false.
	v-same-letter-heights: Specifies whether all letters will be the same height regardless of initial case. If true, the lowercase letters are stretched to the height of the uppercase letters. Default is false.
	v-text-align: Specifies the alignment of text. Default is left. Allowed values are: • left • right • center • justify • letter-justify - Distributes the extra space between the letters. • stretch-justify - Stretches the letters to fill in the space.
	v-text-kern: Specifies whether kerning is turned on. Default is false.
	v-text-reverse: Specifies whether the layout order of rows is reversed. Default is false. This is used for vertical text layout.
	v-text-spacing-mode: Specifies the mode for letter spacing. Default is tightening. This property determines whether space will be removed between each letter (tightening) or added between each letter (tracking). The amount of letter spacing change is defined by the v-text-spacing property. Allowed values are:

Attributes	Description
	<table><tr><td></td><td> - tightening - tracking </td></tr><tr><td>v-text-spacing</td><td>Specifies the amount of spacing for text in 100ths of single line spacing. Default is 100.</td></tr></table> The line (§6.1.2.12), polyline (§6.1.2.15) and curve (§6.1.2.3) elements ignore the following properties: - top - left - width - height The following properties are not inherited by an element that references a shapetype element (§6.1.2.20) via the id attribute: - flip - height - left - margin-left - margin-top - position - rotation - top - visibility - width - z-index The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Textbox">
   <choice>
      <element ref="w:txbxContent" minOccurs="0"/>
      <any namespace="##local" processContents="skip"/>
   </choice>
   <attributeGroup ref="AG_Id"/>
   <attributeGroup ref="AG_Style"/>
   <attribute name="inset" type="xsd:string" use="optional"/>
   <attribute ref="o:singleclick"/>
   <attribute ref="o:insetmode"/>
</complexType>
```

6.1.2.23 textpath (Text Layout Path)

This element is used to define a vector path based on the text data, font and font styles supplied. The path which results is then mapped into the region defined by the v attribute of the shape's path (§6.1.2.14).

[*Example:*

```
<v:curve from="50,100" to="400,100"
  control1="200,200" control2="300,200">
  <v:stroke color="blue"/>
  <v:fill color="yellow" color2="green" type="gradient"/>
  <v:path textpathok="true"/>
  <v:textpath on="true" style="font:normal normal normal 36pt Arial"
    fitpath="true" string="Hello, VML!"/>
</v:curve>
```

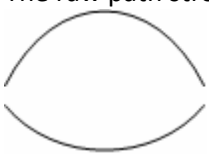
end example]

Parent Elements
arc (§6.1.2.1); background (§2.2.1); curve (§6.1.2.3); group (§6.1.2.7); image (§6.1.2.10); line (§6.1.2.12); object (§2.3.3.19); oval (§6.1.2.13); pict (§2.3.3.21); pict (§2.9.23); polyline (§6.1.2.15); rect (§6.1.2.16); roundrect (§6.1.2.17); shape (§6.1.2.19); shapetype (§6.1.2.20)

Attributes	Description

fitpath (Path Fit Toggle)	Specifies whether the text fits the path of a shape. If `true`, sizes the text to fill the path it lies out on. Default is `false`.
	
	[*Example:*
	
	`<v:textpath on="true" fitpath="true"`
	`  string="VML">`
	`</v:textpath>`
	
	 
	
	end example]

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).

fitshape (Shape Fit Toggle)	Specifies whether the text fits the bounding box of a shape. If `true`, the text is stretched out to the edges of the box that defines the entire shape. Default is `false`.
	
	[*Example:* When fitshape is `false`, the text is drawn along the first part of the path. When `true`, the text is stretched to fit the entire enclosed area of the shape.
	
	`<v:shape style="width:100;height:100"`
	`path="m 0,500 c 250,0 750,0 1000,500 e`
	`m 0,600 c 250,900 750,900 1000,600 e"`
	`fillcolor="yellow" strokecolor="maroon">`
	`<v:path textpathok="t"/>`
	`<v:textpath on="t" fitshape="t" string="VML"/>`
	`</v:shape>`
	
	 
	
	The raw path stroke is:
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).

id (Unique Identifier)	Specifies a unique identifier that can be used to reference a VML object.
	
	Default is no value.
	
	[*Example:*
	
	`<v:shape ... id="myShape" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

on (Text Path	Specifies whether the text is displayed on the textpath. Default is `false`. The

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>Toggle)</td><td>textpathok attribute of the path element (§6.1.2.14) overrides this.

[Example:

<pre><v:line from="50,100" to="100,100">
  <v:path textpathok="false"/>
  <v:textpath on="true" string="VML"/>
</v:line></pre>
———

end example]

The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).</td></tr>
<tr><td>string (Text Path Text)</td><td>Specifies the text of the text path. Default is no value.

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>style (Shape Styling Properties)</td><td>Specifies the CSS2 styling properties of the shape. This uses the syntax described in the "Visual formatting model" of the Cascading Style Sheets, Level 2 specification, a Recommendation of the World Wide Web Consortium available here: http://www.w3.org/TR/REC-CSS2. Full descriptions of each property are not repeated here, but the VML treatment of each property is defined. Allowed properties include:

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>flip</td><td>Specifies that the orientation of a shape is flipped. Default is no value. Allowed values are:

• x - Flip along the y-axis, reversing the x-coordinates.
• y - Flip along the x-axis, reversing the y-coordinates.
• xy - Flip along both the y- and x-axis.
• yx - Flip along both the x- and y-axis.</td></tr>
<tr><td>height</td><td>Specifies the height of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are:

• auto - Default position of an element in the flow of the page.
• <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed.
• <percentage>- Value expressed as a percentage of the parent object's height.</td></tr>
</table>
</td></tr>
</table>

Attributes	Description	
	`left`	Specifies the position of the left of the containing block of the shape relative to the element left of it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	`margin-bottom`	Specifies the position of the bottom of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	`margin-left`	Specifies the position of the left of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	`margin-right`	Specifies the position of the right of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • `auto` - Default position of an element in the flow of the page.

Attributes	Description	
		• <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	margin-top	Specifies the position of the top of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	mso-position-horizontal	Specifies the horizontal positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute • left • center • right • inside • outside
	mso-position-horizontal-relative	Specifies relative horizontal position data for objects in WordprocessingML documents. This modifies the mso-position-horizontal property. Default is text. Allowed values are: • margin • page • text • char
	mso-position-vertical	Specifies the vertical positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute • top

Attributes	Description
	• center • bottom • inside • outside
mso-position-vertical-relative	Specifies relative vertical position data for objects in WordprocessingML documents. This modifies the mso-position-vertical property. Default is text. Allowed values are: • margin • page • text • line
mso-wrap-distance-bottom	Specifies the distance from the bottom of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-distance-left	Specifies the distance from the left side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-distance-right	Specifies the distance from the right side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-distance-top	Specifies the distance from the top of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.
mso-wrap-edited	Specifies whether the wrap coordinates were customized by the user. If the wrap coordinates are generated by an editor, this property is true; otherwise they were customized by a user. Default is false.
mso-wrap-style	Specifies the wrapping mode for text in shapes in WordprocessingML documents. Default is square. Allowed values are: • square - Wraps text inside the shape in a square.

Attributes	Description
	• none - Text does not wrap.
position	Specifies the type of positioning used to place an element. Default is static. When the element is contained inside a group, this property must be absolute. Allowed values are: • static - The element is positioned according to the normal flow of the page. The top and left properties are ignored. If the object is anchored inline, this value is used. • absolute - The element is positioned relative to the parent, using the top and left properties. • relative - The element is positioned according to the normal flow of the page, but the top and left properties are used. The overlap of overlapping elements is governed by the z-index property.
rotation	Specifies the angle that a shape is rotated, in degrees. Default is 0. Positive angles are clockwise.
top	Specifies the position of the top of the containing block of the shape relative to the element above it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
visibility	Specifies whether a shape is displayed. Only inherit and hidden are used; any other values are mapped to inherit. Default is inherit. Allowed values are: • hidden - The shape is not visible, but is still part of the flow of the objects in the browser. Mouse events are not processed. • inherit - The visibility state is inherited from the parent of the shape.
width	Specifies the width of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are:

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>

auto - Default position of an element in the flow of the page.
<units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed.
<percentage>- Value expressed as a percentage of the parent object's width.

</td></tr>
<tr><td>z-index</td><td>Specifies the display order of overlapping shapes. Default is 0. This property shall not be used for shapes anchored inline. Allowed values are:

auto - Uses the order that the shapes appear in the page, bottom to top.
<order>- A number that represents the stacking precedence. Shapes with higher numbers are placed on top of those with lower numbers. Negative numbers are allowed.

</td></tr>
</table>
The following properties are only used by the textbox element (§6.1.2.22):
<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>direction</td><td>Specifies the direction of the text in the textbox. Default is ltr. This property is superceded by the mso-direction-alt property if that is specified. Allowed values are:

ltr - Test is displayed left-to-right.
rtl - Test is displayed right-to-left.

</td></tr>
<tr><td>layout-flow</td><td>Determines the flow of the text layout in a textbox. Default is horizontal. Allowed values are:

horizontal - Text is displayed horizontally.
vertical - Text is displayed vertically.
vertical-ideographic - Ideographic text is displayed vertically.
horizontal-ideographic - Ideographic text is displayed horizontally.

</td></tr>
<tr><td>mso-direction-alt</td><td>Specifies an alternate direction for text in textboxes. Overrides the direction property. The only allowed value is context.</td></tr>
<tr><td>mso-fit-shape-</td><td>Specifies whether the shape stretches to fit the text in the textbox. Default is false.</td></tr>
</table>
</td></tr>
</table>

Attributes	Description

to-text	
mso-fit-text-to-shape	Specifies whether the text stretches to fit the textbox. Default is false.
mso-layout-flow-alt	Specifies the alternate layout flow for text in textboxes. This property is used instead of layout-flow when the layout flow is from bottom to top for non-ideographic languages. Its only value is bottom-to-top.
mso-next-textbox	Specifies the ID of the next textbox in a series. Used to keep track of a set of linked textboxes. Default is no value.
mso-rotate	Specifies a specific rotation value for text in a textbox. Default is 0. Allowed values are: • 0 • 90 • 180 • -90
mso-text-scale	Specifies the scaling factor for fitting text to shapes. Default is 0. This property is only used if mso-fit-text-to-shape is true.
v-text-anchor	Specifies the vertical anchoring of text in a textbox. Default is top. The alignment of a text anchor only becomes evident if mso-fit-text-to-shape is false. This property is different from the vertical-align CSS property, which is used for ideographic languages. Allowed values are: • top • middle • bottom • top-center • middle-center • bottom-center • top-baseline • bottom-baseline • top-center-baseline • bottom-center-baseline

The following properties are only used by the textpath element (§6.1.2.23):

Property	Description
font	Specifies a compound value of font settings. Default is no value. The values are the same as those of the CSS font property. The

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>order of definitions in the string is: font-style, font-variant, font-weight, font-size, line-height, font-family.</td></tr>
<tr><td>font-family</td><td>Specifies the family of the font. Default is no value. The values are the same as those of the CSS font-family property.</td></tr>
<tr><td>font-size</td><td>Specifies the size of the font. Default is no value. The font size is defined in points. The values are the same as those of the CSS font-size property.</td></tr>
<tr><td>font-style</td><td>Specifies the amount of slant for a font. Default is normal. The values are the same as those of the CSS font-style property. Allowed values are:
normal
italic
oblique - Treated the same as italic.</td></tr>
<tr><td>font-variant</td><td>Specifies the variant style of a font. Default is normal. The values are the same as those of the CSS font-variant property. Allowed values are:
normal
small-caps</td></tr>
<tr><td>font-weight</td><td>Specifies the thickness of the letters of the font. Default is normal. The values are the same as those of the CSS font-weight property. Allowed values are:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>normal
lighter
100
200
300
400</td><td>Treated as non-bold.</td></tr>
<tr><td>bold
bolder
500
600</td><td>Treated as bold.</td></tr>
</table>
</td></tr>
</table>
</td></tr>
</table>

Attributes	Description
	700 800 900
	mso-text-shadow: Specifies whether a shadow is applied to the text on a text path. Default is false.
	text-decoration: Specifies the style of text decoration. Default is none. The values are the same as those of the CSS text-decoration property. Allowed values are: • none • underline • overline • line-through • blink
	v-rotate-letters: Specifies whether the letters of the text are rotated counterclockwise by 90 degrees. Default is false.
	v-same-letter-heights: Specifies whether all letters will be the same height regardless of initial case. If true, the lowercase letters are stretched to the height of the uppercase letters. Default is false.
	v-text-align: Specifies the alignment of text. Default is left. Allowed values are: • left • right • center • justify • letter-justify - Distributes the extra space between the letters. • stretch-justify - Stretches the letters to fill in the space.
	v-text-kern: Specifies whether kerning is turned on. Default is false.
	v-text-reverse: Specifies whether the layout order of rows is reversed. Default is false. This is used for vertical text layout.
	v-text-spacing-mode: Specifies the mode for letter spacing. Default is tightening. This property determines whether space will be removed between each letter (tightening) or added between each letter (tracking). The amount of letter spacing change is defined by the v-text-spacing property. Allowed values are:

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>

tightening
tracking

</td></tr>
<tr><td>v-text-spacing</td><td>Specifies the amount of spacing for text in 100ths of single line spacing. Default is 100.</td></tr>
</table>
The line (§6.1.2.12), polyline (§6.1.2.15) and curve (§6.1.2.3) elements ignore the following properties:

top
left
width
height

The following properties are not inherited by an element that references a shapetype element (§6.1.2.20) via the id attribute:

flip
height
left
margin-left
margin-top
position
rotation
top
visibility
width
z-index

The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>trim (Text Path Trim Toggle)</td><td>
Specifies whether extra space is removed above and below the text. If true, space reserved for ascenders and descenders is removed. Default is false.
[Example: The shape path is duplicated as a second shape and overlaid on the textpath for illustrative purposes:
<pre>
<v:shape style=" width:100;height:100"
  path="m 0,500 c 250,0 750,0 1000,500 e
  m 0,600 c 250,900 750,900 1000,600 e"
  fillcolor="yellow" strokecolor="maroon">
  <v:path textpathok="true"/>
  <v:textpath on="true" fitshape="true"
    string="vml" trim="true"/>
</v:shape>
</pre>
</td></tr>
</table>

Attributes	Description
	 `trim="true"`  `trim="false"` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).
xscale (Text X-Scaling)	Specifies whether a straight text path will be used instead of the shape path. If `true`, the text runs along a path from left to right along the x value of the lower boundary of the shape. Default is `false`. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.1.3.14).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TextPath">
   <attributeGroup ref="AG_Id"/>
   <attributeGroup ref="AG_Style"/>
   <attribute name="on" type="ST_TrueFalse" use="optional"/>
   <attribute name="fitshape" type="ST_TrueFalse" use="optional"/>
   <attribute name="fitpath" type="ST_TrueFalse" use="optional"/>
   <attribute name="trim" type="ST_TrueFalse" use="optional"/>
   <attribute name="xscale" type="ST_TrueFalse" use="optional"/>
   <attribute name="string" type="xsd:string" use="optional"/>
</complexType>
```

6.1.3 Simple Types

This is the complete list of simple types in the urn:schemas-microsoft-com:vml namespace.

6.1.3.1 ST_ColorType (Color Type)

This simple type specifies a color. Colors are specified in one of three ways - named color, hexadecimal RGB or color palette entry. An optional index may be stored in square brackets following the color and a space. [*Rationale:* An application might store the color's index in a system color palette using this means. *end rationale*]

A named color is specified using the name of the color. The following named colors are supported:

- `Black (#000000)` ■
- `Silver (#C0C0C0)` ■

- Gray (#808080)
- White (#FFFFFF)
- Maroon (#800000)
- Red (#FF0000)
- Purple (#800080)
- Fuchsia (#FF00FF)
- Green (#008000)
- Lime (#00FF00)
- Olive (#808000)
- Yellow (#FFFF00)
- Navy (#000080)
- Blue (#0000FF)
- Teal (#008080)
- Aqua (#00FFFF)

[*Example:*

```
<... color="red" ... >
```

end example]

Hexadecimal RGB is specified using a hash symbol (#) followed by six hexadecimal characters, where each pair represents the red, green and blue component of the color.

[*Example:*

```
< ... color="#5f2726" ... >
```

end example]

A color palette entry is specified using the name of the color in the palette.

[*Example:*

```
<... color="buttonFace [67]" ... >
```

end example]

This simple type's contents are a restriction of the XML Schema string datatype.

Referenced By
arc@chromakey (§6.1.2.1); arc@fillcolor (§6.1.2.1); arc@strokecolor (§6.1.2.1); background@fillcolor (§6.1.2.2); curve@chromakey (§6.1.2.3); curve@fillcolor (§6.1.2.3); curve@strokecolor (§6.1.2.3); fill@color (§6.1.2.5); fill@color2 (§6.1.2.5); group@fillcolor (§6.1.2.7); image@chromakey (§6.1.2.10); image@fillcolor (§6.1.2.10); image@strokecolor (§6.1.2.10); imagedata@chromakey (§6.1.2.11); imagedata@embosscolor (§6.1.2.11); imagedata@recolortarget (§6.1.2.11); line@chromakey (§6.1.2.12); line@fillcolor (§6.1.2.12); line@strokecolor (§6.1.2.12); oval@chromakey (§6.1.2.13); oval@fillcolor (§6.1.2.13); oval@strokecolor (§6.1.2.13); polyline@chromakey (§6.1.2.15); polyline@fillcolor (§6.1.2.15); polyline@strokecolor

Referenced By
(§6.1.2.15); rect@chromakey (§6.1.2.16); rect@fillcolor (§6.1.2.16); rect@strokecolor (§6.1.2.16); roundrect@chromakey (§6.1.2.17); roundrect@fillcolor (§6.1.2.17); roundrect@strokecolor (§6.1.2.17); shadow@color (§6.1.2.18); shadow@color2 (§6.1.2.18); shape@chromakey (§6.1.2.19); shape@fillcolor (§6.1.2.19); shape@strokecolor (§6.1.2.19); shapetype@chromakey (§6.1.2.20); shapetype@fillcolor (§6.1.2.20); shapetype@strokecolor (§6.1.2.20); stroke@color (§6.1.2.21); stroke@color2 (§6.1.2.21)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ColorType">
   <restriction base="xsd:string"/>
</simpleType>
```

6.1.3.2 ST_EditAs (Shape Grouping Types)

This simple type specifies the different meanings of a group of shapes.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
bullseye (Bullseye Diagram)	Specifies that the group represents a bulls-eye diagram.
canvas (Shape Canvas)	Specifies that the group is a regular group and does not represent a diagram.
cycle (Cycle Diagram)	Specifies that the group represents a cycle diagram.
orgchart (Organization Chart Diagram)	Specifies that the group represents an organization chart.
radial (Radial Diagram)	Specifies that the group represents a radial diagram.
stacked (Pyramid Diagram)	Specifies that the group represents a pyramid diagram.
venn (Venn Diagram)	Specifies that the group represents a Venn diagram.

Referenced By
group@editas (§6.1.2.7)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_EditAs">
   <restriction base="xsd:string">
      <enumeration value="canvas"/>
      <enumeration value="orgchart"/>
      <enumeration value="radial"/>
      <enumeration value="cycle"/>
      <enumeration value="stacked"/>
      <enumeration value="venn"/>
      <enumeration value="bullseye"/>
   </restriction>
</simpleType>
```

6.1.3.3 ST_Ext (VML Extension Handling Behaviors)

This simple type specifies VML extension handling behaviors.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
backwardCompatible (Renderable)	Specifies that the VML entity may be rendered by ignoring the extension information. If edited, the extension information must be discarded.
edit (Editable)	Specifies that the VML entity may be safely rendered and edited without invalidating the extension information.
view (Not renderable)	Specifies that the VML entity is not be renderable without understanding the extension information. If the extension information cannot be understood, the downlevel image should be used to render the object.

Referenced By
bottom@ext (§6.2.2.1); callout@ext (§6.2.2.2); colormenu@ext (§6.2.2.4); colormru@ext (§6.2.2.5); column@ext (§6.2.2.6); complex@ext (§6.2.2.7); diagram@ext (§6.2.2.8); extrusion@ext (§6.2.2.10); fill@ext (§6.2.2.12); idmap@ext (§6.2.2.13); left@ext (§6.2.2.15); lock@ext (§6.2.2.17); regrouptable@ext (§6.2.2.22); rel@ext (§6.2.2.23); relationtable@ext (§6.2.2.24); right@ext (§6.2.2.25); rules@ext (§6.2.2.26); shapedefaults@ext (§6.2.2.27); shapelayout@ext (§6.2.2.28); signatureline@ext (§6.2.2.29); skew@ext (§6.2.2.30); top@ext (§6.2.2.31)

The following XML Schema fragment defines the contents of this simple type:

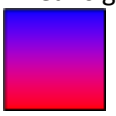
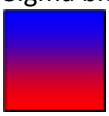
```
<simpleType name="ST_Ext">
   <restriction base="xsd:string">
      <enumeration value="view"/>
      <enumeration value="edit"/>
      <enumeration value="backwardCompatible"/>
   </restriction>
</simpleType>
```

6.1.3.4 ST_FillMethod (Gradient Fill Computation Type)

This simple type specifies ways in which a gradient fill is computed.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
any (Application Default Fill)	Default blend 
linear (Linear Fill)	Linear blend 
linear sigma (Linear Sigma Fill)	Linear sigma blend 
none (No Gradient Fill)	No blend 
sigma (Sigma Fill)	Sigma blend 

Referenced By
fill@method (§6.1.2.5)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FillMethod">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="linear"/>
      <enumeration value="sigma"/>
      <enumeration value="any"/>
      <enumeration value="linear sigma"/>
   </restriction>
</simpleType>
```

6.1.3.5 ST_FillType (Shape Fill Type)

This simple type specifies the types for fills applied to a shape.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
frame (Stretch Image to Fit)	The image is stretched to fill the shape. 
gradient (Linear Gradient)	The fill colors blend together in a linear gradient from bottom to top. 
gradientRadial (Radial Gradient)	The fill colors blend together in a radial gradient. 
pattern (Image Pattern)	The image is used to create a pattern using the fill colors. 
solid (Solid Fill)	The fill pattern is a solid color. 
tile (Tiled Image)	The fill image is tiled. 

Referenced By
bottom@filltype (§6.2.2.1); column@filltype (§6.2.2.6); fill@type (§6.1.2.5); left@filltype (§6.2.2.15); right@filltype (§6.2.2.25); stroke@filltype (§6.1.2.21); top@filltype (§6.2.2.31)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FillType">
   <restriction base="xsd:string">
      <enumeration value="solid"/>
      <enumeration value="gradient"/>
      <enumeration value="gradientRadial"/>
      <enumeration value="tile"/>
      <enumeration value="pattern"/>
      <enumeration value="frame"/>
   </restriction>
</simpleType>
```

6.1.3.6 ST_ImageAspect (Image Scaling Behavior)

This simple type specifies the scaling behaviors for an image applied to a stroke.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
atLeast (At Least)	Image is at least as big as imagesize. 
atMost (At Most)	Image is no bigger than imagesize. 
ignore (Ignore Aspect Ratio)	Ignore aspect issues. 

Referenced By
bottom@imageaspect (§6.2.2.1); column@imageaspect (§6.2.2.6); fill@aspect (§6.1.2.5); left@imageaspect (§6.2.2.15); right@imageaspect (§6.2.2.25); stroke@imageaspect (§6.1.2.21); top@imageaspect (§6.2.2.31)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ImageAspect">
   <restriction base="xsd:string">
      <enumeration value="ignore"/>
      <enumeration value="atMost"/>
      <enumeration value="atLeast"/>
   </restriction>
</simpleType>
```

6.1.3.7 ST_ShadowType (Shadow Type)

This simple type specifies the types of shadows applied to a shape.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
double (Double Shadow)	Double shadow. color2 and offset2 are used for the second shadow's color and offset.
emboss (Embossed Shadow)	The shadow has an embossed look. Similar to double.
perspective (Perspective Shadow)	Perspective shadow.
single (Single Shadow)	Single shadow.

Referenced By
shadow@type (§6.1.2.18)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ShadowType">
   <restriction base="xsd:string">
      <enumeration value="single"/>
      <enumeration value="double"/>
      <enumeration value="emboss"/>
      <enumeration value="perspective"/>
   </restriction>
</simpleType>
```

6.1.3.8 ST_StrokeArrowLength (Stroke Arrowhead Length)

This simple type specifies the lengths of a stroke arrowhead.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description

Enumeration Value	Description
long (Long Arrowhead)	Long length 
medium (Medium Arrowhead)	Medium length 
short (Short Arrowhead)	Short length 

Referenced By
bottom@endarrowlength (§6.2.2.1); bottom@startarrowlength (§6.2.2.1); column@endarrowlength (§6.2.2.6); column@startarrowlength (§6.2.2.6); left@endarrowlength (§6.2.2.15); left@startarrowlength (§6.2.2.15); right@endarrowlength (§6.2.2.25); right@startarrowlength (§6.2.2.25); stroke@endarrowlength (§6.1.2.21); stroke@startarrowlength (§6.1.2.21); top@endarrowlength (§6.2.2.31); top@startarrowlength (§6.2.2.31)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_StrokeArrowLength">
   <restriction base="xsd:string">
      <enumeration value="short"/>
      <enumeration value="medium"/>
      <enumeration value="long"/>
   </restriction>
</simpleType>
```

6.1.3.9 ST_StrokeArrowType (Stroke Arrowhead Type)

This simple type specifies the types of arrowhead for a stroke.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
block (Block Arrowhead)	Block arrowhead 
classic (Classic Arrowhead)	Classic curved arrowhead 
diamond (Diamond Arrowhead)	Diamond arrowhead 
none (No Arrowhead)	No arrowhead 
open (Open Arrowhead)	Open arrowhead 
oval (Oval Arrowhead)	Round arrowhead

Enumeration Value	Description
	

Referenced By
bottom@endarrow (§6.2.2.1); bottom@startarrow (§6.2.2.1); column@endarrow (§6.2.2.6); column@startarrow (§6.2.2.6); left@endarrow (§6.2.2.15); left@startarrow (§6.2.2.15); right@endarrow (§6.2.2.25); right@startarrow (§6.2.2.25); stroke@endarrow (§6.1.2.21); stroke@startarrow (§6.1.2.21); top@endarrow (§6.2.2.31); top@startarrow (§6.2.2.31)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_StrokeArrowType">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="block"/>
      <enumeration value="classic"/>
      <enumeration value="oval"/>
      <enumeration value="diamond"/>
      <enumeration value="open"/>
   </restriction>
</simpleType>
```

6.1.3.10 ST_StrokeArrowWidth (Stroke Arrowhead Width)

This simple type specifies the widths of a stroke arrowhead.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
medium (Medium Arrowhead)	Medium width 
narrow (Narrow Arrowhead)	Narrow width 
wide (Wide Arrowhead)	Wide width 

Referenced By
bottom@endarrowwidth (§6.2.2.1); bottom@startarrowwidth (§6.2.2.1); column@endarrowwidth (§6.2.2.6); column@startarrowwidth (§6.2.2.6); left@endarrowwidth (§6.2.2.15); left@startarrowwidth (§6.2.2.15); right@endarrowwidth (§6.2.2.25); right@startarrowwidth (§6.2.2.25); stroke@endarrowwidth (§6.1.2.21); stroke@startarrowwidth (§6.1.2.21); top@endarrowwidth (§6.2.2.31); top@startarrowwidth (§6.2.2.31)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_StrokeArrowWidth">
   <restriction base="xsd:string">
      <enumeration value="narrow"/>
      <enumeration value="medium"/>
      <enumeration value="wide"/>
   </restriction>
</simpleType>
```

6.1.3.11 ST_StrokeEndCap (Stroke End Cap Type)

This simple type specifies the styles for the end of a stroke.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
flat (Flat End)	Flat end 
round (Round End)	Round end 
square (Square End)	Square end 

Referenced By
bottom@endcap (§6.2.2.1); column@endcap (§6.2.2.6); left@endcap (§6.2.2.15); right@endcap (§6.2.2.25); stroke@endcap (§6.1.2.21); top@endcap (§6.2.2.31)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_StrokeEndCap">
   <restriction base="xsd:string">
      <enumeration value="flat"/>
      <enumeration value="square"/>
      <enumeration value="round"/>
   </restriction>
</simpleType>
```

6.1.3.12 ST_StrokeJoinStyle (Line Join Type)

This simple type specifies the join styles for a polyline (§6.1.2.15).

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
bevel (Bevel Joint)	Bevel joint 
miter (Miter Joint)	Miter joint 
round (Round Joint)	Round joint 

Referenced By
bottom@joinstyle (§6.2.2.1); column@joinstyle (§6.2.2.6); left@joinstyle (§6.2.2.15); right@joinstyle (§6.2.2.25); stroke@joinstyle (§6.1.2.21); top@joinstyle (§6.2.2.31)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_StrokeJoinStyle">
   <restriction base="xsd:string">
      <enumeration value="round"/>
      <enumeration value="bevel"/>
      <enumeration value="miter"/>
   </restriction>
</simpleType>
```

6.1.3.13 ST_StrokeLineStyle (Stroke Line Style)

This simple type specifies the line styles for a stroke.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description

Enumeration Value	Description
single (Single Line)	Single line 
thickBetweenThin (Thck Line Between Thin Lines)	Thick line between thin lines 
thickThin (Thick Line Outside Thin Line)	Thick line outside thin line 
thinThick (Thin Line Outside Thick Line)	Thin line outside thick line 
thinThin (Two Thin Lines)	Two thin lines 

Referenced By
bottom@linestyle (§6.2.2.1); column@linestyle (§6.2.2.6); left@linestyle (§6.2.2.15); right@linestyle (§6.2.2.25); stroke@linestyle (§6.1.2.21); top@linestyle (§6.2.2.31)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_StrokeLineStyle">
   <restriction base="xsd:string">
      <enumeration value="single"/>
      <enumeration value="thinThin"/>
      <enumeration value="thinThick"/>
      <enumeration value="thickThin"/>
      <enumeration value="thickBetweenThin"/>
   </restriction>
</simpleType>
```

6.1.3.14 ST_TrueFalse (Boolean Value)

This type specifies logical true and false.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
f (False)	Logical false.

Enumeration Value	Description
false (False)	Logical false.
t (True)	Logical true.
true (True)	Logical true.

Referenced By
arc@filled (§6.1.2.1); arc@insetpen (§6.1.2.1); arc@print (§6.1.2.1); arc@stroked (§6.1.2.1); background@filled (§6.1.2.2); curve@filled (§6.1.2.3); curve@insetpen (§6.1.2.3); curve@print (§6.1.2.3); curve@stroked (§6.1.2.3); fill@alignshape (§6.1.2.5); fill@on (§6.1.2.5); fill@recolor (§6.1.2.5); fill@rotate (§6.1.2.5); group@filled (§6.1.2.7); group@print (§6.1.2.7); h@invx (§6.1.2.8); h@invy (§6.1.2.8); image@bilevel (§6.1.2.10); image@filled (§6.1.2.10); image@grayscale (§6.1.2.10); image@insetpen (§6.1.2.10); image@print (§6.1.2.10); image@stroked (§6.1.2.10); imagedata@bilevel (§6.1.2.11); imagedata@grayscale (§6.1.2.11); line@filled (§6.1.2.12); line@insetpen (§6.1.2.12); line@print (§6.1.2.12); line@stroked (§6.1.2.12); oval@filled (§6.1.2.13); oval@insetpen (§6.1.2.13); oval@print (§6.1.2.13); oval@stroked (§6.1.2.13); path@arrowok (§6.1.2.14); path@fillok (§6.1.2.14); path@gradientshapeok (§6.1.2.14); path@insetpenok (§6.1.2.14); path@shadowok (§6.1.2.14); path@strokeok (§6.1.2.14); path@textpathok (§6.1.2.14); polyline@filled (§6.1.2.15); polyline@insetpen (§6.1.2.15); polyline@print (§6.1.2.15); polyline@stroked (§6.1.2.15); rect@filled (§6.1.2.16); rect@insetpen (§6.1.2.16); rect@print (§6.1.2.16); rect@stroked (§6.1.2.16); roundrect@filled (§6.1.2.17); roundrect@insetpen (§6.1.2.17); roundrect@print (§6.1.2.17); roundrect@stroked (§6.1.2.17); shadow@obscured (§6.1.2.18); shadow@on (§6.1.2.18); shape@filled (§6.1.2.19); shape@insetpen (§6.1.2.19); shape@print (§6.1.2.19); shape@stroked (§6.1.2.19); shapetype@filled (§6.1.2.20); shapetype@insetpen (§6.1.2.20); shapetype@print (§6.1.2.20); shapetype@stroked (§6.1.2.20); stroke@imagealignshape (§6.1.2.21); stroke@insetpen (§6.1.2.21); stroke@on (§6.1.2.21); textpath@fitpath (§6.1.2.23); textpath@fitshape (§6.1.2.23); textpath@on (§6.1.2.23); textpath@trim (§6.1.2.23); textpath@xscale (§6.1.2.23)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TrueFalse">
   <restriction base="xsd:string">
      <enumeration value="t"/>
      <enumeration value="f"/>
      <enumeration value="true"/>
      <enumeration value="false"/>
   </restriction>
</simpleType>
```

6.1.3.15 ST_TrueFalseBlank (Boolean Value with Blank [False] State)

This simple type specifies a boolean value with a third state, using a blank attribute, which specifies that the value be `false`.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description

Enumeration Value	Description
(Blank – Logical False)	Logical false.
f (Logical False)	Logical false.
false (Logical False)	Logical false.
t (Logical True)	Logical true.
true (Logical True)	Logical true.

Referenced By
h@switch (§6.1.2.8)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TrueFalseBlank">
   <restriction base="xsd:string">
      <enumeration value="t"/>
      <enumeration value="f"/>
      <enumeration value="true"/>
      <enumeration value="false"/>
      <enumeration value=""/>
   </restriction>
</simpleType>
```

6.2 VML - Office Drawing

It is possible to include graphical VML objects in Office Open XML documents. The elements describing the core graphical objects are defined in the VML namespace. Additional elements that describe certain advanced shape effects, metadata and relationships are defined in this namespace.

[*Note*: The VML format is a legacy format originally introduced with Office 2000 and is included and fully defined in this Standard for backwards compatibility reasons. The DrawingML format is a newer and richer format created with the goal of eventually replacing any uses of VML in the Office Open XML formats. VML should be considered a deprecated format included in Office Open XML for legacy reasons only and new applications that need a file format for drawings are strongly encouraged to use preferentially DrawingML .*end note*]

[*Example:* Assume the following shape exists in a document:

The basic speech bubble shape is defined using VML. The 3-D effect is defined using the extrusion element in this namespace. The specularity attribute defines the subtle sharp reflection on the edge of the shape. The

color attribute sets the extrusion to a different color than the face of the shape. The rotationangle attribute sets the shape's rotation about the X- and Y-axes. The lightposition and lightposition2 attributes set the positions of the light sources that illuminate the shape.

```
<o:extrusion v:ext="view" specularity="80000f" color="#c4bc96 [2414]" on="t"
rotationangle="-5,15" lightposition="0,-50000" lightposition2="0,50000"
type="perspective"/>
```

This element is a child of the primary shape definition:

```
<v:shape id="_x0000_s1030" type="#_x0000_t62"
  style="position:absolute;left:0;text-align:left;margin-left:35.25pt;
  margin-top:60pt;width:69pt;height:57pt;z-index:251658240" adj="1675,27171"
  fillcolor="#ddd8c2 [2894]">
  <o:extrusion ... />
</v:shape>
```

end example]

Note that, throughout VML, numeric values that are allowed to take units may be specified in: cm (centimeters), mm (millimeters), in (inches), pt (points), pc (picas), px (pixels).

6.2.1 Table of Contents

This subclause is informative.

End of informative text.

6.2.2 Elements

The following elements comprise the contents of the urn:schemas-microsoft-com:office:office namespace:

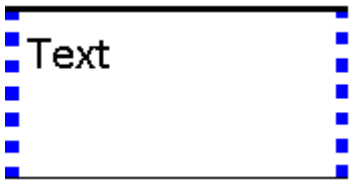
[*Note*: As the VML format is a format provided for backward compatibility, those VML elements defined in the same urn:schemas-microsoft-com:office:office namespace will remain in that namespace is it is already used by millions of documents already using VML. *end note*]

6.2.2.1 bottom (Text Box Bottom Stroke)

This element specifies the stroke properties for the bottom border of a text box. It entirely supercedes its parent stroke element if its on attribute is true. Thus the default value of an unspecified attribute overrides a value specified in the parent. If the on attribute is `false` or not specified, the border is not shown.

[*Example:* The text box borders are set independently. Note that the bottom border does not inherit the weight from the parent stroke element.

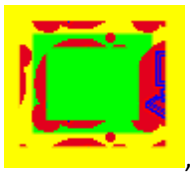
```
<v:stroke weight="2.25pt">
  <o:left v:ext="view" dashstyle="1 1" color="blue" weight="5pt" on="t"/>
  <o:top v:ext="view" color="black" weight="2.25pt" on="t"/>
  <o:right v:ext="view" dashstyle="1 1" color="blue" weight="5pt" on="t"/>
  <o:bottom v:ext="view" color="black" on="t"/>
  <o:column v:ext="view" color="#f60" on="t"/>
</v:stroke>
```



end example]

Parent Elements
background (§2.2.1); hdrShapeDefaults (§2.15.1.50); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23); shapeDefaults (§2.15.1.79); stroke (§6.1.2.21)

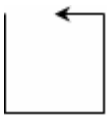
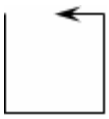
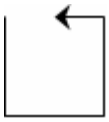
Attributes	Description
althref (Alternate Image Reference)	Specifies an alternate reference for an image in Macintosh PICT format. [*Example:* `<v:stroke ... althref="myimage.pcz" ... ></v:stroke>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
color (Stroke Color)	Specifies the stroke color. Overrides the strokecolor attribute of a shape. Default is `black`. See the fillcolor attribute for a list of supported named colors. [*Example:* The shape stroke is blue:

Attributes	Description
	<v:shape ... strokecolor="red" ... > <v:stroke color="blue"/> </v:shape> *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.2.3.5).
color2 (Stroke Alternate Pattern Color)	Specifies a second color for strokes, used when filltype is `pattern`. Default is no value. When a pattern fill is used for the stroke, the stroke color is used in colored parts of the source image. The color2 defines an alternate color to use in place of black in the source image. [*Example:* This unusual example is intended to demonstrate how the image and colors interact to create a patterned stroke. The yellow background shows transparency. The non-square shape and square image create an effective offset. The heavy stroke weight shows more of the image. The green shape fill shows how the stroke is overlaid on the shape. <v:background fillcolor="yellow"/> <v:shape style="width:60;height:50" strokecolor="red" fillcolor="lime" path="m 0,0 l 0,1000 1000,1000 1000,0 x e"> <v:stroke filltype="pattern" weight="10pt" src="myimage.gif" color2="blue"/> </v:shape>  , where myimage.gif is:  *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.2.3.5).
dashstyle (Stroke Dash Pattern)	Specifies the dot and dash pattern for a stroke. Default is `solid`. Pre-defined values are: • `solid` • `shortdash` • `shortdot` • `shortdashdot` • `shortdashdotdot` • `dot`

Attributes	Description

	• dash
	• longdash
	• dashdot
	• longdashdot
	• longdashdotdot
	
	A custom-defined dash pattern may also be specified using a series of numbers. These define the length of the dash (the drawn part of the stroke) and the length of the space between the dashes. The lengths are relative to the line width: a length of 1 is equal to the line width. The endcap style is applied to each dash but the arrow style is not. The string defines the length of the dash then the length of the space. This may be repeated to form complex dash styles. The string should always contain a pair of numbers; if it contains an odd number of numbers the last is disregarded. 0 implies a dot that is fourfold symmetrical (with round end caps, this is a circle).
	
	[*Example:*
	
	`<v:stroke dashstyle="0 2" weight="3pt" endcap="round"></v:stroke>`
	
	 
	
	`<v:stroke dashstyle="longdashdotdot" weight="2pt"></v:stroke>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

endarrow (Line End Arrowhead)	Specifies an arrowhead for the end of a line. Default is none. Note that the path must not be closed with the x command for the arrowhead to show. Allowed values are: • none • block • classic • diamond • oval • open [*Example:*

Attributes	Description
	<v:stroke endarrow="classic"/>  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowType simple type (§6.1.3.9).
endarrowlength (Line End Arrowhead Length)	Specifies the length of the arrowhead at the end of a line. Default is `medium`. Allowed values are: • `short` • `medium` • `long` [*Example:* <v:stroke ... endarrowlength="long" ... />  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowLength simple type (§6.1.3.8).
endarrowwidth (Line End Arrowhead Width)	Specifies the width of the arrowhead at the end of a line. Default is `medium`. Allowed values are: • `narrow` • `medium` • `wide` [*Example:* <v:stroke ... endarrowwidth="wide" ... /> 

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_StrokeArrowWidth simple type (§6.1.3.10).

endcap (Line End Cap)	Specifies the cap style for the end of a stroke. Default is `flat`. Allowed values are:
	
	• `flat`
	• `square`
	• `round`
	
	[*Example:*
	
	`<v:stroke ... endcap="round" weight="10pt" ... />`
	
	 
	`endcap="flat"`
	`endcap="square"`
	`endcap="round"`
	
	end example]
	
	The possible values for this attribute are defined by the ST_StrokeEndCap simple type (§6.1.3.11).

ext (VML Extension Handling Behavior) Namespace: urn:schemas-microsoft-com:vml	Specifies an optional value that indicates how applications that implement VML should interpret extensions not defined as part of the original specification of core VML. [*Rationale*: This part of the original VML specification is included to assist applications that leverage existing VML support in implementing the Office Open XML Format. *end rationale*] The possible values for this attribute are defined by the ST_Ext simple type (§6.1.3.3).
filltype (Stroke Image Style)	Specifies the type of fill used for the background of a stroke. Default is `solid`. Allowed values are: • `solid` - The fill pattern is solid. • `tile` - The fill image is tiled. • `pattern` - The fill image is stretched to form a pattern. • `frame` - The fill image becomes a border for the shape.

Attributes	Description

	[*Example:*
	
	`<v:shape style="width:50;height:50"`
	`  strokecolor="red"`
	`  path="m 0,0 l 0,1000 1000,1000 1000,0 x e">`
	`  <v:stroke filltype="frame" weight="10pt"`
	`  src="border.gif"/>`
	`</v:shape>`
	
	  , where border.gif is: 
	
	end example]
	
	The possible values for this attribute are defined by the ST_FillType simple type (§6.1.3.5).

forcedash (Force Dashed Outline)	Specifies whether a dashed outline is used to draw a shape when a shape has no line or fill. Default is `false`.
	
	Used by PresentationML placeholders to draw a dashed outline when there is no line and no fill for a shape.
	
	[*Example:*
	
	`<v:shape ... o:forcedash="true" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

href (Original Image Reference)	Specifies the URL to the original image file. Used only if the picture has been linked and embedded. Default is no value.
	
	[*Example:*
	
	`<v:fill ... o:href="myimage.gif" ... >`
	`</v:fill>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

imagealignshape	Specifies the alignment of the stroke image. If `true`, the image is aligned with the shape.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>(Stoke Image Alignment)</td><td>Otherwise, it is aligned with the containing scope. Default is <code>true</code>.

[Example: The top position offset shifts the image alignment relative to the containing window:

<pre><v:shape fillcolor="silver"
  style="top:20;width:50;height:50"
  path="m 0,0 l 0,1000 1000,1000 1000,0 x e">
  <v:stroke imagealignshape="false" weight="20pt"
    filltype="tile" src="myimage.gif"/>
</v:shape></pre>
  imagealignshape="false"

  imagealignshape="false"

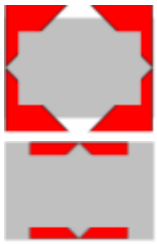
end example]

The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).</td></tr>
<tr><td>imageaspect (Stroke Image Aspect Ratio)</td><td>Specifies how the stroke image aspect ratio is preserved. Default is <code>ignore</code>. Allowed values are:

<table>
<tr><td>Value</td><td>Description</td></tr>
<tr><td><code>ignore</code></td><td>Ignore aspect issues.</td></tr>
<tr><td><code>atleast</code></td><td>Image is at least as big as imagesize.</td></tr>
<tr><td><code>atmost</code></td><td>Image is no bigger than imagesize.</td></tr>
</table>

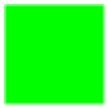
[Example:

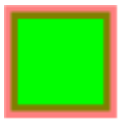
<pre><v:stroke filltype="frame" weight="10pt"
  src="border.gif" imagealignshape="true"
  imageaspect="atleast">
</v:stroke></pre>
  imagealignshape="ignore"</td></tr>
</table>

Attributes	Description
	 imagealignshape="atleast" imagealignshape="atmost" *end example*] The possible values for this attribute are defined by the ST_ImageAspect simple type (§6.1.3.6).
imagesize (Stroke Image Size)	Specifies the size of the image for the stroke. Default is the size of the image. [*Example:* `<v:stroke ... imagesize="10pt,10pt" ... />` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
insetpen (Inset Border From Path)	Specifies that the border shall be displayed inside of the path defining the shape, rather than along the path (the default border placement), or outside of the path as might be done with an image. [*Example:* `<v:shape ... insetpen="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
joinstyle (Line End Join Style))	Specifies the join style for line ends. Default is round. • round • bevel • miter [*Example:* `<v:polyline strokeweight="10pt" strokecolor="navy" points="10pt,10pt,50pt,50pt,90pt,10pt"><v:stroke joinstyle="bevel"/>`

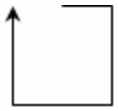
Attributes	Description
	</v:polyline>  joinstyle="round" joinstyle="bevel" joinstyle="miter" *end example*] The possible values for this attribute are defined by the ST_StrokeJoinStyle simple type (§6.1.3.12).
linestyle (Stroke Line Style)	Specifies the line style of the stroke. Default is single. • single • thinThin • thinThick • thickThin • thickBetweenThin [*Example:* <v:stroke linestyle="thickThin" weight="5pt"> </v:stroke>  *end example*] The possible values for this attribute are defined by the ST_StrokeLineStyle simple type (§6.1.3.13).
miterlimit (Miter Joint Limit)	Specifies the smoothness of the miter joint, or the maximum distance between the inner point and outer point of a joint. This number is a multiple of the thickness of the line. Default is 8. [*Example:* <v:stroke joinstyle="miter" weight="10pt"

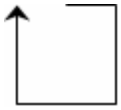
Attributes	Description
	<pre> miterlimit="2">
</v:stroke></pre>  *end example*] The possible values for this attribute are defined by the XML Schema decimal datatype.

on (Stroke Toggle)	Specifies whether the stroke is displayed. Default is `true`. This attribute overrides the shape's stroke attribute.
	
	[*Example:*
	
	<pre><v:rect style="width:50;height:50" stroked="true"
	  fillcolor="lime" strokecolor="red">
	  <v:stroke on="false" weight="5pt"/>
	</v:rect></pre> 
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

opacity (Stroke Opacity)	Specifies the amount of transparency of a stroke. Default is 1.0.
	
	[*Example:*
	
	<pre><v:rect style="width:50;height:50"
	  fillcolor="lime" strokecolor="red">
	  <v:stroke weight="5pt" opacity="50%"/>
	</v:rect></pre> 
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

src (Stroke Image Location)	Specifies the source image to load for a stroke fill. Default is no value.

Attributes	Description
	[*Example:* `<v:stroke ... src="myimage.gif" ... ></v:stroke>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
startarrow (Line Start Arrowhead)	Specifies an arrowhead for the start of a line. Default is none. Note that the path must not be closed with the x command for the arrowhead to show. Allowed values are: • none • block • classic • diamond • oval • open [*Example:* `<v:stroke startarrow="classic"/>`  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowType simple type (§6.1.3.9).
startarrowlength (Line Start Arrowhead Length)	Specifies the length of the arrowhead at the start of a line. Default is medium. Allowed values are: • short • medium • long [*Example:* `<v:stroke ... startarrowlength="long" ... />`  *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_StrokeArrowLength simple type (§6.1.3.8).
startarrowwidth (Line Start Arrowhead Width)	Specifies the width of the arrowhead at the start of a line. Default is `medium`. Allowed values are: • `narrow` • `medium` • `wide` [*Example:* `<v:stroke ... startarrowwidth="wide" ... />`  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowWidth simple type (§6.1.3.10).
title (Stroke Title)	Specifies the title of an embedded stroke image. This is typically set to the comment property of the image, which is often blank. [*Example:* `<v:fill ... o:title="alt text" ... ></v:fill>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
weight (Stroke Weight)	Specifies the thickness of a stroke. Default is 1. This attribute overrides the shape's strokeweight attribute. The possible values for this attribute are defined by the XML Schema string datatype.

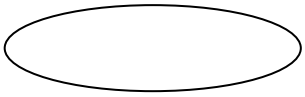
The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_StrokeChild">
    <attributeGroup ref="v:AG_Ext"/>
    <attribute name="on" type="ST_TrueFalse" use="optional"/>
    <attribute name="weight" type="xsd:string" use="optional"/>
    <attribute name="color" type="ST_ColorType" use="optional"/>
    <attribute name="color2" type="ST_ColorType" use="optional"/>
    <attribute name="opacity" type="xsd:string" use="optional"/>
    <attribute name="linestyle" type="v:ST_StrokeLineStyle" use="optional"/>
    <attribute name="miterlimit" type="xsd:decimal" use="optional"/>
    <attribute name="joinstyle" type="v:ST_StrokeJoinStyle" use="optional"/>
    <attribute name="endcap" type="v:ST_StrokeEndCap" use="optional"/>
    <attribute name="dashstyle" type="xsd:string" use="optional"/>
    <attribute name="insetpen" type="ST_TrueFalse" use="optional"/>
    <attribute name="filltype" type="v:ST_FillType" use="optional"/>
    <attribute name="src" type="xsd:string" use="optional"/>
    <attribute name="imageaspect" type="v:ST_ImageAspect" use="optional"/>
    <attribute name="imagesize" type="xsd:string" use="optional"/>
    <attribute name="imagealignshape" type="ST_TrueFalse" use="optional"/>
    <attribute name="startarrow" type="v:ST_StrokeArrowType" use="optional"/>
    <attribute name="startarrowwidth" type="v:ST_StrokeArrowWidth" use="optional"/>
    <attribute name="startarrowlength" type="v:ST_StrokeArrowLength" use="optional"/>
    <attribute name="endarrow" type="v:ST_StrokeArrowType" use="optional"/>
    <attribute name="endarrowwidth" type="v:ST_StrokeArrowWidth" use="optional"/>
    <attribute name="endarrowlength" type="v:ST_StrokeArrowLength" use="optional"/>
    <attribute ref="href"/>
    <attribute ref="althref"/>
    <attribute ref="title"/>
    <attribute ref="forcedash"/>
</complexType>
```

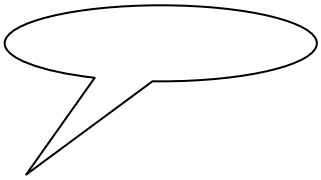
6.2.2.2 callout (Callout)

This element specifies the automatic behavior and layout parameters of callout shapes. Callout shapes are standard VML shapes that behave as callouts, providing an additional callout object which can be used to point at another location:

[*Example*: Consider the following VML shape:



If this shape is made a callout shape by adding the callout element to its shape definition, then the shape will have a callout object, for example:



end example]

Parent Elements
arc (§6.1.2.1); background (§2.2.1); curve (§6.1.2.3); group (§6.1.2.7); hdrShapeDefaults (§2.15.1.50); image (§6.1.2.10); line (§6.1.2.12); object (§2.3.3.19); oval (§6.1.2.13); pict (§2.3.3.21); pict (§2.9.23); polyline (§6.1.2.15); rect (§6.1.2.16); roundrect (§6.1.2.17); shape (§6.1.2.19); shapedefaults (§6.2.2.27); shapeDefaults (§2.15.1.79); shapetype (§6.1.2.20)

Attributes	Description
accentbar (Callout accent bar toggle)	Specifies whether an accent bar will be used with the callout. Default is `false`. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
angle (Callout angle)	Specifies the angle that the callout makes with respect to the bounding box of the shape. Default is no value. The possible values for this attribute are defined by the ST_Angle simple type (§6.2.3.1).
distance (Callout drop distance)	Specifies the drop distance of a callout. The drop distance of a callout is measured from the edge of the shape where the pointer line starts and continues the absolute length of the distance value. If specified with no units, EMUs are assumed. Default is no value. The possible values for this attribute are defined by the XML Schema string datatype.
drop (Callout drop position)	Specifies where the drop of a callout will be placed. The possible values for this attribute are defined by the ST_CalloutDrop simple type (§6.2.3.3).
dropauto (Callout automatic drop toggle)	Specifies whether the callout has an automatic drop. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
ext (VML Extension Handling Behavior) Namespace: urn:schemas-microsoft-com:vml	Specifies an optional value that indicates how applications that implement VML should interpret extensions not defined as part of the original specification of core VML. [*Rationale*: This part of the original VML specification is included to assist applications that leverage existing VML support in implementing the Office Open XML Format. *end rationale*] The possible values for this attribute are defined by the ST_Ext simple type (§6.1.3.3).
gap (Callout gap)	Specifies the distance of the callout line from the bounding rectangle of the callout. Default value is one-twelfth of an inch, in EMUs (76200). The possible values for this attribute are defined by the XML Schema string datatype.
length (Callout	Specifies the length of the first part of a multi-segmented callout line. If specified with no

Attributes	Description
length)	units, EMUs are assumed. Default is 0. The possible values for this attribute are defined by the XML Schema string datatype.
lengthspecified (Callout length toggle)	Specifies whether the length attribute will be used for the callout. Default is `false`. If `true`, the length attribute is used. If `false`, a best fit is used. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
minusx (Callout flip x)	Specifies whether the callout flips to the other side of the drop tip along the x-axis when moved or resized. Default is `false`. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
minusy (Callout flip y)	Specifies whether the callout flips to the other side of the drop tip along the y-axis when moved or resized. Default is `false`. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
on (Callout toggle)	Specifies whether a shape is a callout. Default is `false`. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
textborder (Callout text border toggle)	Specifies whether a callout has a text border. Default is `true`. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
type (Callout type)	Specifies the type of callout. Default is `rectangle`. Allowed values are: • `rectangle` • `roundedrectangle` • `oval` • `cloud` The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Callout">
   <attributeGroup ref="v:AG_Ext"/>
   <attribute name="on" type="ST_TrueFalse" use="optional"/>
   <attribute name="type" type="xsd:string" use="optional"/>
   <attribute name="gap" type="xsd:string" use="optional"/>
   <attribute name="angle" type="ST_Angle" use="optional"/>
   <attribute name="dropauto" type="ST_TrueFalse" use="optional"/>
   <attribute name="drop" type="ST_CalloutDrop" use="optional"/>
   <attribute name="distance" type="xsd:string" use="optional"/>
   <attribute name="lengthspecified" type="ST_TrueFalse" default="f" use="optional"/>
   <attribute name="length" type="xsd:string" use="optional"/>
   <attribute name="accentbar" type="ST_TrueFalse" use="optional"/>
   <attribute name="textborder" type="ST_TrueFalse" use="optional"/>
   <attribute name="minusx" type="ST_TrueFalse" use="optional"/>
   <attribute name="minusy" type="ST_TrueFalse" use="optional"/>
</complexType>
```

6.2.2.3 clippath (Shape Clipping Path)

This element specifies the path of the clipping polygon for the shape.

[*Example:*

```
<v:rect ... wrapcoords="-207 -433 -207 21925 21807 21925 21807 -433 -207 -433"
o:clip="t" o:cliptowrap="t">
  <o:clippath o:v="m-207,-433r,22358l21807,21925r,-22358l-207,-433xe"/>
</v:rect>
```

end example]

Parent Elements
arc (§6.1.2.1); background (§2.2.1); curve (§6.1.2.3); group (§6.1.2.7); hdrShapeDefaults (§2.15.1.50); image (§6.1.2.10); line (§6.1.2.12); object (§2.3.3.19); oval (§6.1.2.13); pict (§2.3.3.21); pict (§2.9.23); polyline (§6.1.2.15); rect (§6.1.2.16); roundrect (§6.1.2.17); shape (§6.1.2.19); shapeDefaults (§2.15.1.79); shapetype (§6.1.2.20)

Attributes	Description
v (Path Definition)	Specifies a string containing the commands that define the shape's path. This value consists of commands followed by zero or more parameters. Default is no value. The following rules apply to path strings: • Commas or spaces delimit parameters for each command. Both "m 0,0" and "m0 0" are acceptable. • A parameter that is omitted using commas is treated as having a value of zero. Thus, "c 10,10,0,0,25,13" and "c 10,10,,,25,13" are equivalent. • Parameterized paths are also allowed. In this case, the shape must also have a

Attributes	Description

formulas element (§6.1.2.6) with a list of formulas that are substituted into the path using the @ symbol followed by the number of the formula. The adj property of the shape contains the input parameters for these formulas. For example, "moveto @1@4". The evaluations of the formulas are substituted into the appropriate positions. Note that @ also serves as a delimiter.

The allowed commands are given below. An asterisk (*) indicates that the command is allowed to be repeated. For the qb command, the controlpoint parameter is also allowed to be repeated.

Command	Name	Parameters	Description
m	moveto	2	Start a new sub-path at the given (x,y) coordinate.
l	lineto	2*	Draw a line from the current point to the given (x,y) coordinate which becomes the new current point. Specifying a number of coordinate pairs forms a polyline.
c	curveto	6*	Draw a cubic bézier curve from the current point to the coordinate given by the final two parameters. The control points are given by the first four parameters.
x	close	0	Close the current sub-path by drawing a straight line from the current point to the original moveto point.
e	end	0	End the current set of sub-paths. A given set of sub-paths (as delimited by end) is filled. Subsequent sets of sub-paths are filled independently and superimposed on existing ones.
t	rmoveto	2*	Start a new sub-path at a coordinate relative to the current point, cp (cpx+x, cpy+y).
r	rlineto	2*	Draw a line from the current point to the given relative coordinate (cpx+x, cpy+y).
v	rcurveto	6*	Cubic bézier curve using the given coordinate relative to the current point.
nf	nofill	0	The current set of sub-paths (delimited by e) will not be filled.

Attributes	Description			
	ns	nostroke	0	The current set of sub-paths (delimited by e) will not be stroked.
	ae	angleellipseto	6*	Draws a segment of an ellipse as described using these parameters. A straight line is drawn from the current point to the start point of the segment. The parameters are: center (x,y), size(w,h), start angle, end angle.
	al	angleellipse	6*	Same as angleellipseto except that there is an implied moveto the starting point of the segment.
	at	arcto	8*	A segment of the ellipse is drawn which starts at the angle defined by the start radius vector and ends at the angle defined by the end vector. A straight line is drawn from the current point to the start of the arc. The arc is always drawn in a counterclockwise direction. The parameters are: left, top, right, bottom, start(x,y), end(x,y). The first four values define the bounding box of an ellipse. The last four define two radial vectors.
	ar	arc	8*	Same as arcto except there is an implied moveto the start point of the arc.
	wa	clockwisearcto	8*	Same as arcto but the arc is drawn in a clockwise direction.
	wr	clockwisearc	8*	Same as arc but the arc is drawn in a clockwise direction
	qx	ellipticalqaudrantx	2*	A quarter ellipse is drawn from the current point to the given end point. The elliptical segment is initially tangential to a line parallel to the x-axis. (i.e. the segment starts out horizontal). The parameters are: end(x,y).
	qy	ellipticalquadranty	2*	Same as ellipticalquadrantx except that the elliptical segment is initially tangential to a line parallel to the y-axis (i.e. the segment starts out vertical).

Attributes	Description
	qb | quadraticbezier | 2+2* | Defines one or more quadratic bézier curves by means of control points and an end point. Intermediate (on-curve) points are obtained by interpolation between successive control points as in the OpenType font specification. The sub-path need not be started in which case the sub-path will be closed. In this case the last point of the sub-path defines the start point of the quadratic bézier. The parameters are: controlpoint(x,y)*, end(x,y). The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ClipPath">
   <attribute name="v" type="xsd:string" use="required" form="qualified"/>
</complexType>
```

6.2.2.4 colormenu (UI Default Colors)

This element determines the default colors for different types of colors that can be applied to VML shapes. [*Rationale:* An application may choose to retain default colors or the last color choices a user made and present those in parts of its user interface. *end rationale*]

[*Example:*

```
<o:shapedefaults ... >
  <o:colormenu v:ext="edit" fillcolor="none" extrusioncolor="#36f"/>
</o:shapedefaults>
```

end example]

Parent Elements
shapedefaults (§6.2.2.27)

Attributes	Description
ext (VML Extension Handling Behavior) Namespace: urn:schemas-	Specifies an optional value that indicates how applications that implement VML should interpret extensions not defined as part of the original specification of core VML. [*Rationale*: This part of the original VML specification is included to assist applications that leverage existing VML support in implementing the Office Open XML Format. *end*

Attributes	Description
microsoft-com:vml	*rationale*] The possible values for this attribute are defined by the ST_Ext simple type (§6.1.3.3).
extrusioncolor (Default extrusion color)	The default color associated with the 3D extrusion of a VML shape. Default is "#000000". The possible values for this attribute are defined by the ST_ColorType simple type (§6.2.3.5).
fillcolor (Default fill color)	The default color associated with the fill of a VML shape. Default is "#0000FF". The possible values for this attribute are defined by the ST_ColorType simple type (§6.2.3.5).
shadowcolor (Default shadow color)	The default color associated with the shadow of a VML shape. Default is "#80800C". The possible values for this attribute are defined by the ST_ColorType simple type (§6.2.3.5).
strokecolor (Default stroke color)	The default color associated with the stroke of a VML shape. Default is "#FFFF00". The possible values for this attribute are defined by the ST_ColorType simple type (§6.2.3.5).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ColorMenu">
   <attributeGroup ref="v:AG_Ext"/>
   <attribute name="strokecolor" type="ST_ColorType"/>
   <attribute name="fillcolor" type="ST_ColorType"/>
   <attribute name="shadowcolor" type="ST_ColorType"/>
   <attribute name="extrusioncolor" type="ST_ColorType"/>
</complexType>
```

6.2.2.5 colormru (Most Recently Used Colors)

This element defines a list of up to eight colors which represent the colors most recently used by the user. [*Rationale:* An application may choose to retain the last color choices a user made, regardless of where on VML shapes they are used, and present those in parts of its user interface. *end rationale*]

[*Example:*

```
<o:shapedefaults ... >
  <o:colormru v:ext="edit" colors="#a01aae,#456b69,#06f,#a1ae24,#d57811"/>
</o:shapedefaults>
```

end example]

Parent Elements
shapedefaults (§6.2.2.27)

Attributes	Description
colors (Recent colors)	A comma-separated list of up to eight most recently used colors. Default is no value. Colors should be defined using hexadecimal notation - see the ST_ColorType simple type (§6.2.3.5) for a full description. [*Example:* `<o:colormru v:ext="edit" colors="#a01aae,#456b69,#06f,#a1ae24,#d57811"/>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
ext (VML Extension Handling Behavior) Namespace: urn:schemas-microsoft-com:vml	Specifies an optional value that indicates how applications that implement VML should interpret extensions not defined as part of the original specification of core VML. [*Rationale*: This part of the original VML specification is included to assist applications that leverage existing VML support in implementing the Office Open XML Format. *end rationale*] The possible values for this attribute are defined by the ST_Ext simple type (§6.1.3.3).

The following XML Schema fragment defines the contents of this element:

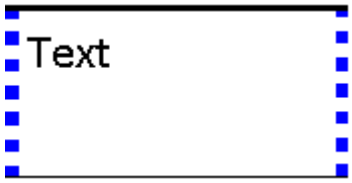
```
<complexType name="CT_ColorMru">
   <attributeGroup ref="v:AG_Ext"/>
   <attribute name="colors" type="xsd:string"/>
</complexType>
```

6.2.2.6 column (Text Box Interior Stroke)

This element specifies the stroke properties for the interior border of a text box. It entirely supercedes its parent stroke element if its on attribute is true. Thus the default value of an unspecified attribute overrides a value specified in the parent. If the on attribute is `false` or not specified, the border is not shown. [*Note:* This element is ignored if an implementation does not support multi-column text boxes. *end note*]

[*Example:* The text box borders are set independently. Note that the bottom border does not inherit the weight from the parent stroke element.

```
<v:stroke weight="2.25pt">
  <o:left v:ext="view" dashstyle="1 1" color="blue" weight="5pt" on="t"/>
  <o:top v:ext="view" color="black" weight="2.25pt" on="t"/>
  <o:right v:ext="view" dashstyle="1 1" color="blue" weight="5pt" on="t"/>
  <o:bottom v:ext="view" color="black" on="t"/>
  <o:column v:ext="view" color="#f60" on="t"/>
</v:stroke>
```

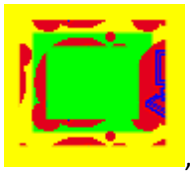


end example]

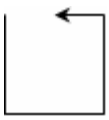
Parent Elements
background (§2.2.1); hdrShapeDefaults (§2.15.1.50); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23); shapeDefaults (§2.15.1.79); stroke (§6.1.2.21)

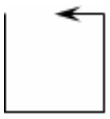
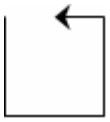
Attributes	Description
althref (Alternate Image Reference)	Specifies an alternate reference for an image in Macintosh PICT format. [*Example:* `<v:stroke ... althref="myimage.pcz" ... ></v:stroke>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
color (Stroke Color)	Specifies the stroke color. Overrides the strokecolor attribute of a shape. Default is `black`. See the fillcolor attribute for a list of supported named colors. [*Example:* The shape stroke is blue: `<v:shape ... strokecolor="red" ... ><v:stroke color="blue"/></v:shape>` *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.2.3.5).
color2 (Stroke Alternate Pattern Color)	Specifies a second color for strokes, used when filltype is `pattern`. Default is no value. When a pattern fill is used for the stroke, the stroke color is used in colored parts of the source image. The color2 defines an alternate color to use in place of black in the source image. [*Example:* This unusual example is intended to demonstrate how the image and colors interact to create a patterned stroke. The yellow background shows transparency. The non-square shape and square image create an effective offset. The heavy stroke weight

Attributes	Description

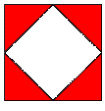
	shows more of the image. The green shape fill shows how the stroke is overlaid on the shape.
	
	`<v:background fillcolor="yellow"/>`
	`<v:shape style="width:60;height:50"`
	`  strokecolor="red" fillcolor="lime"`
	`  path="m 0,0 l 0,1000 1000,1000 1000,0 x e">`
	`  <v:stroke filltype="pattern" weight="10pt"`
	`  src="myimage.gif" color2="blue"/>`
	`</v:shape>`
	
	  , where myimage.gif is: 
	
	end example]
	
	The possible values for this attribute are defined by the ST_ColorType simple type (§6.2.3.5).

dashstyle (Stroke Dash Pattern)	Specifies the dot and dash pattern for a stroke. Default is `solid`. Pre-defined values are: • `solid` • `shortdash` • `shortdot` • `shortdashdot` • `shortdashdotdot` • `dot` • `dash` • `longdash` • `dashdot` • `longdashdot` • `longdashdotdot` A custom-defined dash pattern may also be specified using a series of numbers. These define the length of the dash (the drawn part of the stroke) and the length of the space between the dashes. The lengths are relative to the line width: a length of 1 is equal to the line width. The endcap style is applied to each dash but the arrow style is not. The string defines the length of the dash then the length of the space. This may be repeated to form complex dash styles. The string should always contain a pair of numbers; if it contains an odd number of numbers the last is disregarded. 0 implies a dot that is fourfold symmetrical (with round end caps, this is a circle). [*Example:* `<v:stroke dashstyle="0 2" weight="3pt"`

Attributes	Description
	endcap="round"> </v:stroke>  <v:stroke dashstyle="longdashdotdot" weight="2pt"> </v:stroke>  *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
endarrow (Line End Arrowhead)	Specifies an arrowhead for the end of a line. Default is none. Note that the path must not be closed with the x command for the arrowhead to show. Allowed values are: • none • block • classic • diamond • oval • open [*Example:* <v:stroke endarrow="classic"/>  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowType simple type (§6.1.3.9).
endarrowlength (Line End Arrowhead Length)	Specifies the length of the arrowhead at the end of a line. Default is medium. Allowed values are: • short • medium • long

Attributes	Description
	[*Example:* `<v:stroke ... endarrowlength="long" ... />`  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowLength simple type (§6.1.3.8).
endarrowwidth (Line End Arrowhead Width)	Specifies the width of the arrowhead at the end of a line. Default is `medium`. Allowed values are: • `narrow` • `medium` • `wide` [*Example:* `<v:stroke ... endarrowwidth="wide" ... />`  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowWidth simple type (§6.1.3.10).
endcap (Line End Cap)	Specifies the cap style for the end of a stroke. Default is `flat`. Allowed values are: • `flat` • `square` • `round` [*Example:* `<v:stroke ... endcap="round" weight="10pt" ... />`  `endcap="flat"`

Attributes	Description
	 endcap="square" endcap="round" *end example*] The possible values for this attribute are defined by the ST_StrokeEndCap simple type (§6.1.3.11).
ext (VML Extension Handling Behavior) Namespace: urn:schemas-microsoft-com:vml	Specifies an optional value that indicates how applications that implement VML should interpret extensions not defined as part of the original specification of core VML. [*Rationale*: This part of the original VML specification is included to assist applications that leverage existing VML support in implementing the Office Open XML Format. *end rationale*] The possible values for this attribute are defined by the ST_Ext simple type (§6.1.3.3).

filltype (Stroke Image Style)	Specifies the type of fill used for the background of a stroke. Default is `solid`. Allowed values are:
	
	• `solid` - The fill pattern is solid.
	• `tile` - The fill image is tiled.
	• `pattern` - The fill image is stretched to form a pattern.
	• `frame` - The fill image becomes a border for the shape.
	
	[*Example:*
	
	`<v:shape style="width:50;height:50"`
	`  strokecolor="red"`
	`  path="m 0,0 l 0,1000 1000,1000 1000,0 x e">`
	`  <v:stroke filltype="frame" weight="10pt"`
	`  src="border.gif"/>`
	`</v:shape>`
	
	  , where border.gif is: 
	
	end example]
	
	The possible values for this attribute are defined by the ST_FillType simple type (§6.1.3.5).

Attributes	Description
forcedash (Force Dashed Outline)	Specifies whether a dashed outline is used to draw a shape when a shape has no line or fill. Default is `false`. Used by PresentationML placeholders to draw a dashed outline when there is no line and no fill for a shape. [*Example:* `<v:shape ... o:forcedash="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
href (Original Image Reference)	Specifies the URL to the original image file. Used only if the picture has been linked and embedded. Default is no value. [*Example:* `<v:fill ... o:href="myimage.gif" ... ></v:fill>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
imagealignshape (Stoke Image Alignment)	Specifies the alignment of the stroke image. If `true`, the image is aligned with the shape. Otherwise, it is aligned with the containing scope. Default is `true`. [*Example:* The top position offset shifts the image alignment relative to the containing window: `<v:shape fillcolor="silver" style="top:20;width:50;height:50" path="m 0,0 l 0,1000 1000,1000 1000,0 x e"><v:stroke imagealignshape="false" weight="20pt" filltype="tile" src="myimage.gif"/></v:shape>`  `imagealignshape="false"`

Attributes	Description
	 imagealignshape="false" *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
imageaspect (Stroke Image Aspect Ratio)	Specifies how the stroke image aspect ratio is preserved. Default is ignore. Allowed values are: Value / Description ignore — Ignore aspect issues. atleast — Image is at least as big as imagesize. atmost — Image is no bigger than imagesize. [*Example:* `<v:stroke filltype="frame" weight="10pt" src="border.gif" imagealignshape="true" imageaspect="atleast"> </v:stroke>`  imagealignshape="ignore"  imagealignshape="atleast"  imagealignshape="atmost" *end example*] The possible values for this attribute are defined by the ST_ImageAspect simple type (§6.1.3.6).
imagesize (Stroke Image Size)	Specifies the size of the image for the stroke. Default is the size of the image. [*Example:* `<v:stroke ... imagesize="10pt,10pt" ... />`

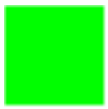
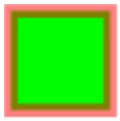
Attributes	Description
	end example] The possible values for this attribute are defined by the XML Schema string datatype.
insetpen (Inset Border From Path)	Specifies that the border shall be displayed inside of the path defining the shape, rather than along the path (the default border placement), or outside of the path as might be done with an image. [*Example:* `<v:shape ... insetpen="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
joinstyle (Line End Join Style))	Specifies the join style for line ends. Default is round. • round • bevel • miter [*Example:* `<v:polyline strokeweight="10pt" strokecolor="navy" points="10pt,10pt,50pt,50pt,90pt,10pt"><v:stroke joinstyle="bevel"/></v:polyline>`  joinstyle="round" joinstyle="bevel" joinstyle="miter" *end example*] The possible values for this attribute are defined by the ST_StrokeJoinStyle simple type

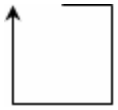
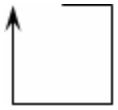
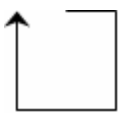
Attributes	Description
	(§6.1.3.12).

linestyle (Stroke Line Style)	Specifies the line style of the stroke. Default is `single`.
	
	• `single`
	• `thinThin`
	• `thinThick`
	• `thickThin`
	• `thickBetweenThin`
	
	[*Example:*
	
	`<v:stroke linestyle="thickThin" weight="5pt">`
	`</v:stroke>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_StrokeLineStyle simple type (§6.1.3.13).

miterlimit (Miter Joint Limit)	Specifies the smoothness of the miter joint, or the maximum distance between the inner point and outer point of a joint. This number is a multiple of the thickness of the line. Default is 8.
	
	[*Example:*
	
	`<v:stroke joinstyle="miter" weight="10pt"`
	`miterlimit="2">`
	`</v:stroke>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema decimal datatype.

on (Stroke Toggle)	Specifies whether the stroke is displayed. Default is `true`. This attribute overrides the shape's stroke attribute.
	
	[*Example:*
	
	`<v:rect style="width:50;height:50" stroked="true"`
	`fillcolor="lime" strokecolor="red">`

Attributes	Description
	<v:stroke on="false" weight="5pt"/> </v:rect>  *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
opacity (Stroke Opacity)	Specifies the amount of transparency of a stroke. Default is 1.0. [*Example:* <v:rect style="width:50;height:50" fillcolor="lime" strokecolor="red"> <v:stroke weight="5pt" opacity="50%"/> </v:rect>  *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
src (Stroke Image Location)	Specifies the source image to load for a stroke fill. Default is no value. [*Example:* <v:stroke ... src="myimage.gif" ... > </v:stroke> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
startarrow (Line Start Arrowhead)	Specifies an arrowhead for the start of a line. Default is none. Note that the path must not be closed with the x command for the arrowhead to show. Allowed values are: • none • block • classic • diamond • oval • open

Attributes	Description
	[*Example:* `<v:stroke startarrow="classic"/>`  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowType simple type (§6.1.3.9).
startarrowlength (Line Start Arrowhead Length)	Specifies the length of the arrowhead at the start of a line. Default is `medium`. Allowed values are: • `short` • `medium` • `long` [*Example:* `<v:stroke ... startarrowlength="long" ... />`  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowLength simple type (§6.1.3.8).
startarrowwidth (Line Start Arrowhead Width)	Specifies the width of the arrowhead at the start of a line. Default is `medium`. Allowed values are: • `narrow` • `medium` • `wide` [*Example:* `<v:stroke ... startarrowwidth="wide" ... />` 

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_StrokeArrowWidth simple type (§6.1.3.10).
title (Stroke Title)	Specifies the title of an embedded stroke image. This is typically set to the comment property of the image, which is often blank. [*Example:* `<v:fill ... o:title="alt text" ... ></v:fill>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
weight (Stroke Weight)	Specifies the thickness of a stroke. Default is 1. This attribute overrides the shape's strokeweight attribute. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_StrokeChild">
   <attributeGroup ref="v:AG_Ext"/>
   <attribute name="on" type="ST_TrueFalse" use="optional"/>
   <attribute name="weight" type="xsd:string" use="optional"/>
   <attribute name="color" type="ST_ColorType" use="optional"/>
   <attribute name="color2" type="ST_ColorType" use="optional"/>
   <attribute name="opacity" type="xsd:string" use="optional"/>
   <attribute name="linestyle" type="v:ST_StrokeLineStyle" use="optional"/>
   <attribute name="miterlimit" type="xsd:decimal" use="optional"/>
   <attribute name="joinstyle" type="v:ST_StrokeJoinStyle" use="optional"/>
   <attribute name="endcap" type="v:ST_StrokeEndCap" use="optional"/>
   <attribute name="dashstyle" type="xsd:string" use="optional"/>
   <attribute name="insetpen" type="ST_TrueFalse" use="optional"/>
   <attribute name="filltype" type="v:ST_FillType" use="optional"/>
   <attribute name="src" type="xsd:string" use="optional"/>
   <attribute name="imageaspect" type="v:ST_ImageAspect" use="optional"/>
   <attribute name="imagesize" type="xsd:string" use="optional"/>
   <attribute name="imagealignshape" type="ST_TrueFalse" use="optional"/>
   <attribute name="startarrow" type="v:ST_StrokeArrowType" use="optional"/>
   <attribute name="startarrowwidth" type="v:ST_StrokeArrowWidth" use="optional"/>
   <attribute name="startarrowlength" type="v:ST_StrokeArrowLength" use="optional"/>
   <attribute name="endarrow" type="v:ST_StrokeArrowType" use="optional"/>
   <attribute name="endarrowwidth" type="v:ST_StrokeArrowWidth" use="optional"/>
   <attribute name="endarrowlength" type="v:ST_StrokeArrowLength" use="optional"/>
   <attribute ref="href"/>
   <attribute ref="althref"/>
   <attribute ref="title"/>
   <attribute ref="forcedash"/>
</complexType>
```

6.2.2.7 complex (Complex)

This element specifies that a shapetype contains fragments.

Parent Elements
background (§2.2.1); hdrShapeDefaults (§2.15.1.50); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23); shapeDefaults (§2.15.1.79); shapetype (§6.1.2.20)

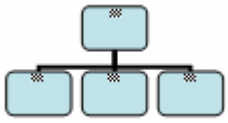
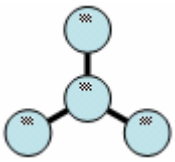
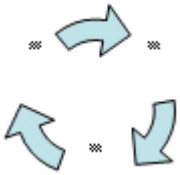
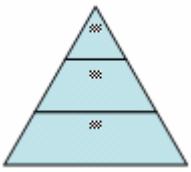
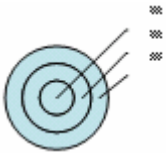
Attributes	Description
ext (VML Extension Handling Behavior) Namespace: urn:schemas-microsoft-com:vml	Specifies an optional value that indicates how applications that implement VML should interpret extensions not defined as part of the original specification of core VML. [*Rationale*: This part of the original VML specification is included to assist applications that leverage existing VML support in implementing the Office Open XML Format. *end rationale*] The possible values for this attribute are defined by the ST_Ext simple type (§6.1.3.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Complex">
   <attributeGroup ref="v:AG_Ext"/>
</complexType>
```

6.2.2.8 diagram (VML Diagram)

This element specifies semantic information for a limited set of structured diagrams that have VML representations. Note that diagrams should be defined using DrawingML; this representation is included for compatibility with applications that rely on VML. The following diagram types have VML representations:

Diagram Type	Example (non-normative)
Organization chart	
Radial	
Cycle	
Pyramid	
Venn	
Bulls-eye	

Each of these types of diagrams contains shapes that are positioned relative to one another. Each shape also has optional associated text.

Parent Elements
background (§2.2.1); group (§6.1.2.7); hdrShapeDefaults (§2.15.1.50); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23); shapeDefaults (§2.15.1.79)

Child Elements	Subclause
relationtable (Diagram Relationship Table)	§6.2.2.24

Attributes	Description

autoformat (Diagram Automatic Format)	Specifies whether the diagram is formatted automatically by the application and user overrides are locked. Default is `false`.
	
	[*Example:*
	
	`<o:diagram ... autoformat="true">`
	`</o:diagram>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

autolayout (Diagram Automatic Layout)	Specifies whether the diagram elements are laid out automatically by the application and user overrides are locked. Default is `true`.
	
	[*Example:*
	
	`<o:diagram ... autolayout="false">`
	`</o:diagram>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

constrainbounds (Diagram Layout Extents)	Specifies an optional, application-specific parameter related to the diagram's extents intended to be used by the application to assist laying out the diagram.
	
	[*Example:*
	
	`<o:diagram ... constrainbounds="2910,2696,9773,9558">`
	`</o:diagram>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

dgmbasetextscale (Diagram Base Font Size)	Specifies the diagram's original font size. This is used in subsequent font size recalculations. If the most recent diagram font size is used to calculate the font size after a rescale, the font size would be wrong after non-isometric diagram rescalings.

Attributes	Description

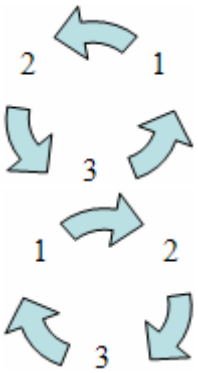
	[*Example:*
	
	`<o:diagram ... dgmbasetextscale="12">`
	`</o:diagram>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema integer datatype.

dgmfontsize (Diagram Font Size)	Specifies the font size for attached text when a new diagram node is added.
	
	[*Example:*
	
	`<o:diagram ... dgmfontsize="12">`
	`</o:diagram>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema integer datatype.

dgmscalex (Diagram Layout X Scale)	Specifies an optional, application-specific parameter related to the horizontal scaling of the diagram that is intended to be used by the application to assist laying out the diagram.
	
	[*Example:*
	
	`<o:diagram ... dgmscalex="50000">`
	`</o:diagram>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema integer datatype.

dgmscaley (Diagram Layout Y Scale)	Specifies an optional, application-specific parameter related to the vertical scaling of the diagram that is intended to be used by the application to assist laying out the diagram.
	
	[*Example:*
	
	`<o:diagram ... dgmscaley="75000">`
	`</o:diagram>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema integer datatype.

dgmstyle (Diagram Style Options)	Specifies an optional, application-specific parameter related to the styling of the diagram that is intended to be used by the application to assist in formatting the diagram. [*Example:*

Attributes	Description
	<o:diagram ... dgmstyle="1"> </o:diagram> *end example*] The possible values for this attribute are defined by the XML Schema integer datatype.
ext (VML Extension Handling Behavior) Namespace: urn:schemas-microsoft-com:vml	Specifies an optional value that indicates how applications that implement VML should interpret extensions not defined as part of the original specification of core VML. [*Rationale*: This part of the original VML specification is included to assist applications that leverage existing VML support in implementing the Office Open XML Format. *end rationale*] The possible values for this attribute are defined by the ST_Ext simple type (§6.1.3.3).
reverse (Diagram Reverse Direction)	Specifies whether the order of the diagram nodes is reversed. This is only relevant to diagrams that have linear ordering. [*Example:* <o:diagram ... reverse="true"> </o:diagram>  reverse="false" reverse="true" *end example* The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Diagram">
   <sequence>
      <element name="relationtable" type="CT_RelationTable" minOccurs="0"/>
   </sequence>
   <attributeGroup ref="v:AG_Ext"/>
   <attribute name="dgmstyle" type="xsd:integer" use="optional"/>
   <attribute name="autoformat" type="ST_TrueFalse" use="optional"/>
   <attribute name="reverse" type="ST_TrueFalse" use="optional"/>
   <attribute name="autolayout" type="ST_TrueFalse" use="optional"/>
   <attribute name="dgmscalex" type="xsd:integer" use="optional"/>
   <attribute name="dgmscaley" type="xsd:integer" use="optional"/>
   <attribute name="dgmfontsize" type="xsd:integer" use="optional"/>
   <attribute name="constrainbounds" type="xsd:string" use="optional"/>
   <attribute name="dgmbasetextscale" type="xsd:integer" use="optional"/>
</complexType>
```

6.2.2.9 entry (Regroup Entry)

This element specifies a single entry in a regrouptable (§6.2.2.22). Each entry is a pair mapping a current regroupid value to an old one. This is used to restore regrouping information on the regrouped object. A value of zero indicates no previous group.

[*Example:* The zero value of the old attribute indicates that if the shapes with regroupid 1 are regrouped, the restored group was not previously grouped with any other shapes:

```
<o:regrouptable v:ext="edit">
  <o:entry new="1" old="0"/>
</o:regrouptable>
```

end example]

Parent Elements
regrouptable (§6.2.2.22)

Attributes	Description
new (New Group ID)	Specifies the ID of the new group. Default is 0. The possible values for this attribute are defined by the XML Schema int datatype.
old (Old Group ID)	Specifies the ID of the old group. Default is 0. The possible values for this attribute are defined by the XML Schema int datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Entry">
   <attribute name="new" type="xsd:int" use="optional"/>
   <attribute name="old" type="xsd:int" use="optional"/>
</complexType>
```

6.2.2.10 extrusion (3D Extrusion)

This element specifies a parallel or perspective extrusion of a 2-D shape, creating the appearance of a 3-D shape. Lighting is controlled via two independent point light sources. Extrusions are defined as either perspective or parallel.

[*Example:*

```
<v:polyline points="0pt,75pt 20pt,45pt 10pt,50pt 30pt,10pt
  50pt,50pt 40pt,45pt 60pt,75pt 0pt,75pt" fillcolor="#00a000">
  <o:extrusion on="t" backdepth="20pt"
    lightposition="30000,10000,10000"/>
</v:polyline>
```

end example]

Parent Elements
arc (§6.1.2.1); background (§2.2.1); curve (§6.1.2.3); group (§6.1.2.7); hdrShapeDefaults (§2.15.1.50); image (§6.1.2.10); line (§6.1.2.12); object (§2.3.3.19); oval (§6.1.2.13); pict (§2.3.3.21); pict (§2.9.23); polyline (§6.1.2.15); rect (§6.1.2.16); roundrect (§6.1.2.17); shape (§6.1.2.19); shapedefaults (§6.2.2.27); shapeDefaults (§2.15.1.79); shapetype (§6.1.2.20)

Attributes	Description
autorotationcenter (Center of Rotation Toggle)	Specifies whether the center of rotation is the geometric center of the extrusion. Default is `false`. If `true`, the geometric center of an extruded shape is (0,0,0). If `false`, the center of rotation is determined by the rotationcenter attribute. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
backdepth (Backward Extrusion Depth)	Specifies the amount of backward extrusion. Default is 36 pt, default units are points. [*Example:*

Attributes	Description
	<o:extrusion on="true" backdepth="15pt"> </o:extrusion>  *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
brightness (Brightness)	Specifies the overall brightness of a scene. Default is 0.3. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied (as "f" indicates the value is a fraction). For example, a value of "52429f" represents 52429/65536 or 0.8. This quantity is not specified using units. The numeric values range from 0 to 1 (0f to 65536f), where 0 implies darkness and 1 implies light saturation. [*Example:* <o:extrusion on="true" brightness="0.4"> </o:extrusion>  brightness="0" brightness="25000f" brightness="0.4" brightness="0.75" *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
color (Extrusion Color)	Specifies the color of the extrusion faces. This attribute is only used when colormode is custom. Colors are typically specified as either a named color, such as red, or six hexadecimal digits representing the red, green and blue values of the color, such as

Attributes	Description

	#00FF30. Full details are specified in the simple type description.
	
	[*Example:*
	
	`<o:extrusion on="true" color="lime"`
	`  colormode="custom">`
	`</o:extrusion>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_ColorType simple type (§6.2.3.5).

colormode (Extrusion Color Mode)	Specifies whether the extrusion color is defined by the color attribute or is the same as the shape's fill color. Default is auto.
	
	[*Example:*
	
	`<o:extrusion on="true" color="lime"`
	`  colormode="auto">`
	`</o:extrusion>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_ColorMode simple type (§6.2.3.4).

diffusity (Diffuse Reflection)	Specifies the amount of diffusion of reflected light from an extruded shape, defined as the ratio of incident light to diffused reflected light. Default is 1. Normal values range from 0 to 1. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied (as "f" indicates the value is a fraction). For example, a value of "52429f" represents 52429/65536 or 0.8. This quantity is not specified using units. The numeric values range from 0 to 1 (0f to 65536f), where 0 implies all reflected light is diffuse and 1 implies no reflected light is diffuse. Note that specularity and diffusity should be considered together as it is possible, though

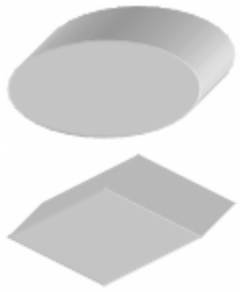
Attributes	Description

	physically incorrect, to define more reflected light than incident light. This is the case if the amount of specularly reflected light and diffusely reflected light add up to more than the amount of incident light.
	
	[*Example:*
	
	`<o:extrusion on="true" diffusity=".75">`
	`</o:extrusion>`
	
	  `diffusity="0"`
	  `diffusity="0.5"`
	  `diffusity="0.75"`
	  `diffusity="1"`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

edge (Simulated Bevel)	Specifies the apparent bevel of the extrusion edges. Default is 1 point.
	
	[*Example:*
	
	`<o:extrusion on="true" edge="2pt">`
	`</o:extrusion>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

ext (VML Extension	Specifies an optional value that indicates how applications that implement VML should

Attributes	Description
Handling Behavior) Namespace: urn:schemas-microsoft-com:vml	interpret extensions not defined as part of the original specification of core VML. [*Rationale*: This part of the original VML specification is included to assist applications that leverage existing VML support in implementing the Office Open XML Format. *end rationale*] The possible values for this attribute are defined by the ST_Ext simple type (§6.1.3.3).

facet (Faceting Quality)	Specifies the quality with which the application approximates curved surfaces of an extrusion. A higher facet value produces shapes with smoother curves. A lower value reduces smoothing, resulting in curves with sharper, jagged edges. Default is 30000.
	
	Valid values range from 1 to 65536, where 1 implies extremely low quality curve approximation and 65536 implies extremely high quality.
	
	[*Example:*
	
	`<o:extrusion on="true" facet="65536">`
	`</o:extrusion>`
	
	 
	`facet="65536"`
	
	`facet="100"`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

foredepth (Forward Extrusion)	Specifies the amount of forward extrusion. Default is 0 pt, default units are points.
	
	[*Example:*
	
	`<o:extrusion on="true" foredepth="25pt">`
	`</o:extrusion>`
	
	 
	
	end example]

Attributes	Description
	The possible values for this attribute are defined by the XML Schema string datatype.

lightface (Shape Face Lighting Toggle)	Specifies whether the front face of the extrusion responds to changes in the lighting. If `false`, the front face does not respond when a lighting value changes. Default is `true`.
	
	[*Example:* The front face is colored as if the shape were not extruded and lit by a 3-D light source:
	
	`<o:extrusion on="true" lightface="false">`
	`</o:extrusion>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

lightharsh (Primary Light Harshness Toggle)	Specifies whether the primary light source is harsh. If `false`, shadow boundaries are diffused. Default is `true`.
	
	[*Example:* The secondary light source is turned off so only the primary has an effect:
	
	`<o:extrusion on="true" lightharsh="false"`
	`lightlevel2="0">`
	`</o:extrusion>`
	
	 
	`lightharsh="false"`
	
	`lightharsh="true"`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

lightharsh2 (Secondary Light Harshness Toggle)	Specifies whether the secondary light source is harsh. If `false`, shadow boundaries defined by the secondary light source are diffused. Default is `false`. The possible values for this attribute are defined by the ST_TrueFalse simple type

Attributes	Description
	(§6.2.3.23).

lightlevel (Primary Light Intensity)	Specifies the intensity of the primary light source for the scene. Default is 0.6. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied (as "f" indicates the value is a fraction). For example, a value of "52429f" represents 52429/65536 or 0.8.
	
	This quantity is not specified using units. The numeric values range from 0 to 1 (0f to 65536f), where 0 implies no direct light and 1 implies saturated direct light.
	
	[*Example:* The secondary light source is turned off so only the primary has an effect:
	
	`<o:extrusion on="true" lightlevel=".5"`
	`  lightlevel2="0">`
	`</o:extrusion>`
	
	  `lightlevel="1"`
	  `lightlevel="0.5"`
	  `lightlevel="0"`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

lightlevel2 (Secondary Light Intensity)	Specifies the intensity of the secondary light source for the scene. Default is 0.6. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied (as "f" indicates the value is a fraction). For example, a value of "52429f" represents 52429/65536 or 0.8. This quantity is not specified using units. The numeric values range from 0 to 1 (0f to 65536f), where 0 implies no direct light and 1 implies saturated direct light. The possible values for this attribute are defined by the XML Schema string datatype.
lightposition (Primary Light Position)	Specifies the normalized X,Y,Z position of the primary light in a scene in 1/65536-ths. Default is "50000,0,10000". The use of a normalized vector from the shape origin effectively establishes the direction of the light relative to the shape. The distance of the light from the shape is irrelevant as the light source is treated as a directional light.

Attributes	Description

	The position "0,0,0" is at the center of the shape. Positive numbers move the light to the right, down and toward the viewer, respectively.
	
	[*Example:* The secondary light source is turned off so only the primary has an effect:
	
	`<o:extrusion on="true" lightlevel2="0"`
	`  lightposition="7000,-13000,20000">`
	`</o:extrusion>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

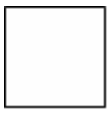
lightposition2 (Secondary Light Position)	Specifies the X,Y,Z position of the secondary light in a scene in 1/65536-ths. Default is "-50000,0,10000". The use of a normalized vector from the shape origin effectively establishes the direction of the light relative to the shape. The distance of the light from the shape is irrelevant as the light source is treated as a directional light. The position "0,0,0" is at the center of the shape. Positive numbers move the light to the right, down and toward the viewer, respectively. The possible values for this attribute are defined by the XML Schema string datatype.

lockrotationcenter (Rotation Toggle)	Specifies whether the rotation of the extruded object is specified by the rotationangle attribute. If `false`, the rotation is specified by the orientation attribute. Default is `true`.
	
	[*Example:* The following snippets are equivalent:
	
	`<o:extrusion lockrotationcenter="false"`
	`  orientationangle="45" orientation="0,1,0">`
	`</o:extrusion>`
	
	`<o:extrusion lockrotationcenter=true`
	`  rotationangle="45"/>`
	`</o:extrusion>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

metal (Metallic Surface Toggle)	Specifies whether the surface of the extruded shape resembles metal. Default is `false`.

Attributes	Description

	If `true`, this attribute causes the specularly reflected light to be the material color instead of the light source color, making the object seem more metallic. To further approximate a metallic material requires that specularity be relatively high (about 1.2) and diffusity be relatively low (about 0.6).
	
	[*Example:*
	
	`<o:extrusion on="true" metal="true"`
	`lightposition="10000,-10000,10000"`
	`lightlevel2="0" specularity="1.2"`
	`diffusity="0.6">`
	`</o:extrusion>`
	
	  `metal="true"`
	  `metal="false"`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

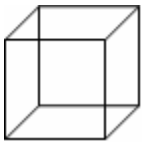
on (Extrusion Toggle)	Specifies whether an extrusion is displayed. Default is `false`.
	
	[*Example:*
	
	`<v:rect style="width=50;height=50">`
	`<o:extrusion />`
	`</v:rect>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

orientation (Rotation Axis)	Specifies a vector in 3D space around which the shape is rotated, as given by the orientationangle attribute. Default is "100,0,0". The position "0,0,0" is at the center of the shape. Positive numbers are to the right,

Attributes	Description
	down and toward the viewer, respectively. [*Example:* `<o:extrusion ... orientation="200,0,0"> </o:extrusion>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
orientationangle (Rotation Around Axis)	Specifies the angle, in degrees, that an extrusion rotates around the orientation. Default is 0. [*Example:* `<o:extrusion ... orientationangle="30"> </o:extrusion>` *end example*] The possible values for this attribute are defined by the XML Schema float datatype.

plane (Extrusion Direction)	Specifies the plane that is at right angles to the extrusion. Default is `xy`. Allowed values are:
	
	• `xy`
	• `zx`
	• `yz`
	
	[*Example:*
	
	`<o:extrusion on="true" plane="yz"`
	`backdepth="100pt">`
	`</o:extrusion>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_ExtrusionPlane simple type (§6.2.3.8).

render (Extrusion	Specifies the rendering mode of the extrusion. Default is `solid`. Allowed values are:

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>Render Mode)</td><td>
<table>
<tr><td>Value</td><td>Description</td></tr>
<tr><td><code>solid</code></td><td>Rendering displays a solid shape.</td></tr>
<tr><td><code>wireframe</code></td><td>Rendering displays a wireframe shape.</td></tr>
<tr><td><code>boundingcube</code></td><td>Rendering displays the bounding cube that contains the shape.</td></tr>
</table>
[Example:
<pre>
<o:extrusion on="true" render="wireframe">
</o:extrusion>
</pre>

end example]
The possible values for this attribute are defined by the ST_ExtrusionRender simple type (§6.2.3.9).
</td></tr>
<tr><td>rotationangle (X-Y Rotation Angle)</td><td>
Specifies the rotation of the object about the x- and y-axes, in degrees. Default is "0,0". Positive angles are measured clockwise around the axis (as if viewing from the positive axis).
The rotation of the object is defined by a rotation angle about the y-axis followed by the rotation angle about the x-axis. The z-axis angle is controlled by the value of the CSS style attribute's <code>rotation</code> property.
[Example:
<pre>
<o:extrusion on="t" lockrotationcenter="true"
  rotationangle="10,20">
</o:extrusion>
</pre>

end example]
The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
</table>

Attributes	Description
rotationcenter (Rotation Center)	Specifies the center of rotation for a shape if autorotationcenter is `false`. The offset of the rotation is specified in terms of fractions of the shape's size. Default is "0,0,0". The position "0,0,0" is at the center of the shape. Positive numbers are to the right, down and toward the viewer, respectively. The possible values for this attribute are defined by the XML Schema string datatype.
shininess (Shininess)	Specifies the concentration of the reflected light on an extrusion surface. Default is 5. The range of values should be constrained to 0-10. Reflection intensity typically grows exponentially with the shininess value. High values (8-10) approximate the shininess of a mirror and low values (2-3) approximate a speckled effect. Reflections do not mirror other objects; only pinpoint light sources are reflected. The possible values for this attribute are defined by the XML Schema float datatype.

skewamt (Extrusion Skew)	Specifies the amount of skew, or length, of a parallel extrusion. Default is 50%. Applies only if the extrusion type is `parallel`. Note that this attribute and backdepth interact to create the actual extrusion length. Valid values range from 0 (0%) to 1 (100%).
	
	[*Example:*
	
	`<o:extrusion on="true" skewamt="100%">`
	`</o:extrusion>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

skewangle (Extrusion Skew Angle)	Specifies the angle of the skew of a parallel extrusion. Default is 225 degrees. Angles are measured in degrees, counterclockwise from the negative x-axis. Applies only if the extrusion type is `parallel`.
	
	[*Example:*
	
	`<o:extrusion on="true" skewangle="25">`
	`</o:extrusion>`

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td> 

end example]

The possible values for this attribute are defined by the XML Schema float datatype.</td></tr>
<tr><td>specularity (Specularity)</td><td>Specifies the specularity of an extruded shape, defined as the ratio of incident light to specularly reflected light. Default is 0. Normal values range from 0 to 1. This numeric value may also be specified in 1/65536-ths if a trailing "f" is supplied (as "f" indicates the value is a fraction). For example, a value of "52429f" represents 52429/65536 or 0.8.

Note that specularity and diffusity should be considered together as it is possible, though physically incorrect, to define more reflected light than incident light. This is the case if the amount of specularly reflected light and diffusely reflected light add up to more than the amount of incident light.

[Example: The secondary light source is turned off so only the primary has an effect. Although the effect is subtle, the first cylinder has a sharper specular reflection on its edge:

<pre><o:extrusion on="true" specularity="1"
  lightposition="10000,-10000,10000"
  lightlevel2="0">
</o:extrusion></pre>
 
<pre>specularity="1"</pre>
 
<pre>specularity="0"</pre>
end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>type (Extrusion Type)</td><td>Specifies the way that the shape is extruded. Default is <code>parallel</code>. Allowed values are:

<table><tr><th>Value</th><th>Description</th></tr><tr><td><code>parallel</code></td><td>Extrusion is rendered so that the center of projection is infinitely far away; that is, the extrusion lines do not converge (unlike</td></tr></table></td></tr>
</table>

Attributes	Description

	<table><tr><td></td><td>perspective projections).</td></tr><tr><td>perspective</td><td>Extrusion is rendered to a center of projection, which is the same as the vanishing point for unrotated objects.</td></tr></table>
	[*Example:*
	
	`<o:extrusion on="true" type="parallel"`
	`  backdepth="100pt">`
	`</o:extrusion>`
	
	  `type="parallel"`
	  `type="perspective"`
	
	end example]
	
	The possible values for this attribute are defined by the ST_ExtrusionType simple type (§6.2.3.10).

viewpoint (Extrusion Viewpoint)	Specifies the viewpoint of the observer in EMUs. This is effectively the end of a vector extending from the viewpointorigin.
	
	The position "0,0,0" is at the center of the shape. Positive numbers are to the right, down and toward the viewer, respectively.
	
	[*Example:*
	
	`<o:extrusion on="true" type="perspective"`
	`  viewpoint="500000,-100000,100000">`
	`</o:extrusion>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

Attributes	Description

viewpointorigin (Extrusion Viewpoint Origin)	Specifies the origin of the viewpoint vector for perspective extrusions. This is the origin of the vector whose opposite end is given by the viewpoint attribute. This origin is always within the bounding box of the shape. Default is "0.5,-0.5".
	
	The viewpoint is specified in terms of the x and y values of the original shape. The x and y values range from 0.5 to -0.5 (50% to -50% of the shape's coordinate origin). Larger numbers move the viewpoint outside the bounding box.
	
	[*Example:*
	
	`<o:extrusion on="true" type="perspective"`
	`viewpoint="500000,-100000,100000"`
	`viewpointorigin="0,1">`
	`</o:extrusion>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Extrusion">
    <attributeGroup ref="v:AG_Ext"/>
    <attribute name="on" type="ST_TrueFalse" use="optional"/>
    <attribute name="type" type="ST_ExtrusionType" default="parallel" use="optional"/>
    <attribute name="render" type="ST_ExtrusionRender" default="solid" use="optional"/>
    <attribute name="viewpointorigin" type="xsd:string" use="optional"/>
    <attribute name="viewpoint" type="xsd:string" use="optional"/>
    <attribute name="plane" type="ST_ExtrusionPlane" default="XY" use="optional"/>
    <attribute name="skewangle" type="xsd:float" use="optional"/>
    <attribute name="skewamt" type="xsd:string" use="optional"/>
    <attribute name="foredepth" type="xsd:string" use="optional"/>
    <attribute name="backdepth" type="xsd:string" use="optional"/>
    <attribute name="orientation" type="xsd:string" use="optional"/>
    <attribute name="orientationangle" type="xsd:float" use="optional"/>
    <attribute name="lockrotationcenter" type="ST_TrueFalse" use="optional"/>
    <attribute name="autorotationcenter" type="ST_TrueFalse" use="optional"/>
    <attribute name="rotationcenter" type="xsd:string" use="optional"/>
    <attribute name="rotationangle" type="xsd:string" use="optional"/>
    <attribute name="colormode" type="ST_ColorMode" use="optional"/>
    <attribute name="color" type="ST_ColorType" use="optional"/>
    <attribute name="shininess" type="xsd:float" use="optional"/>
    <attribute name="specularity" type="xsd:string" use="optional"/>
    <attribute name="diffusity" type="xsd:string" use="optional"/>
    <attribute name="metal" type="ST_TrueFalse" use="optional"/>
    <attribute name="edge" type="xsd:string" use="optional"/>
    <attribute name="facet" type="xsd:string" use="optional"/>
    <attribute name="lightface" type="ST_TrueFalse" use="optional"/>
    <attribute name="brightness" type="xsd:string" use="optional"/>
    <attribute name="lightposition" type="xsd:string" use="optional"/>
    <attribute name="lightlevel" type="xsd:string" use="optional"/>
    <attribute name="lightharsh" type="ST_TrueFalse" use="optional"/>
    <attribute name="lightposition2" type="xsd:string" use="optional"/>
    <attribute name="lightlevel2" type="xsd:string" use="optional"/>
    <attribute name="lightharsh2" type="ST_TrueFalse" use="optional"/>
</complexType>
```

6.2.2.11 FieldCodes (WordprocessingML Field Switches)

This element specifies the WordprocessingML field switches which shall be stored with an embedded object, using the set of field switches defined by the LINK field, as specified in §2.16. This element shall only be used within a WordprocessingML document, and shall specify the exact field switches for the field which represents the object..

[*Rationale*: Legacy word processors used fields to represent embedded objects – this element stores the field switches not explicitly defined using individual Office VML Drawing elements for embeddings so as not to use the fidelity of their contents. *end rationale*]

[*Example*: The following example inserts an embedded object and specifies additional properties as defined by the LINK field.

```
<o:OLEObject …>
  <o:FieldCodes>\f 0</o:FieldCodes>
</o:OLEObject>
```

This embedded object specifies additional LINK field code values of `\f 0`, which specifies that the embedded object shall retain its source formatting (as defined in §2.16).

end example]

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
OLEObject (§6.2.2.19)

6.2.2.12 fill (Shape Fill Extended Properties)

This element specifies additional properties for fills. It is used to identify additional types of gradient fills beyond those specified in the fill element (§6.1.2.5).

Parent Elements
background (§2.2.1); fill (§6.1.2.5); hdrShapeDefaults (§2.15.1.50); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23); shapeDefaults (§2.15.1.79)

Attributes	Description
ext (VML Extension Handling Behavior) Namespace: urn:schemas-microsoft-com:vml	Specifies an optional value that indicates how applications that implement VML should interpret extensions not defined as part of the original specification of core VML. [*Rationale*: This part of the original VML specification is included to assist applications that leverage existing VML support in implementing the Office Open XML Format. *end rationale*] The possible values for this attribute are defined by the ST_Ext simple type (§6.1.3.3).

type (Fill Type)	Specifies the type of fill. If specified, this overrides the value of the type attribute in the parent fill element.
	
	[*Example:* The `gradientCenter` value overrides `gradientRadial`:
	
	`<v:fill color2="black" focus="100%" type="gradientRadial">`
	`  <o:fill v:ext="view" type="gradientCenter"/>`
	`</v:fill>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_FillType simple type (§6.2.3.11).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Fill">
   <attributeGroup ref="v:AG_Ext"/>
   <attribute name="type" type="ST_FillType"/>
</complexType>
```

6.2.2.13 idmap (Shape ID Map)

This element specifies how shape IDs in the document have been generated. This is an optional element included to allow applications a mechanism for storing information they need to persist related to generating shape IDs.

Parent Elements
shapelayout (§6.2.2.28)

Attributes	Description

data (Shape IDs)	Specifies the data the application uses to generate shape IDs.
	
	[*Example:* An application might choose to reserve blocks of shape ID numbers for each part in the package. Each block of 1024 shape IDs could be referred to by index and this index stored in the data attribute. The data value for a given part might then be:
	
	`<o:idmap v:ext="edit" data="1"/>`
	
	indicating that all the IDs in block 1 are reserved by this part (meaning shape IDs from 1 to 1024 cannot be used). The application's internal constraint would be that each part reserve a different set of IDs. Another part, that contains more shapes, might use:
	
	`<o:idmap v:ext="edit" data="2,3"/>`
	
	In this case, shape IDs from 1025 to 3072 [3 x 1024] cannot be used).
	
	Another implementation might choose to store more verbose information in this attribute. Yet another implementation might ignore this element completely.
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

ext (VML Extension Handling Behavior) Namespace: urn:schemas-microsoft-com:vml	Specifies an optional value that indicates how applications that implement VML should interpret extensions not defined as part of the original specification of core VML. [*Rationale*: This part of the original VML specification is included to assist applications that leverage existing VML support in implementing the Office Open XML Format. *end rationale*] The possible values for this attribute are defined by the ST_Ext simple type (§6.1.3.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_IdMap">
   <attributeGroup ref="v:AG_Ext"/>
   <attribute name="data" type="xsd:string" use="optional"/>
</complexType>
```

6.2.2.14 ink (Ink)

This element specifies the presence of an ink object. An ink object is a VML object which allows applications to store data for ink annotations in an application-defined format.

[*Example:*

```
<v:shape ... >
  <o:ink i="AMgFHQSWC+YFASAAaAwAAAAAAMA..." annotation="t"/>
</v:shape>
```

end example]

Parent Elements
background (§2.2.1); hdrShapeDefaults (§2.15.1.50); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23); polyline (§6.1.2.15); shape (§6.1.2.19); shapeDefaults (§2.15.1.79)

Attributes	Description

annotation (Annotation Flag)	Specifies whether the ink object was created as an annotation rather than through pen input. Default is `false`. [*Rationale* This allows an application to treat annotation ink objects as any other annotation. For example, if annotations are hidden, the application can hide the ink object. An ink object that represents primary user input through a pen can be left visible. *end rationale*]
	
	[*Example:*
	
	`<o:ink ... annotation="true">`
	`</o:ink>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

i (Ink Data)	Specifies additional ink object information which shall be associated with the parent VML shape. The VML shape specifies the information necessary to render the ink, and this attribute may be used to store additional application-specific data about the VML shape(s) representing ink. This attribute's contents are optional and may be ignored if not recognized. [*Example:*

Attributes	Description
	<o:ink ... i="AMgFHQSWC+YFASAAaAwAAAAAAMA..."> </o:ink> *end example*] The possible values for this attribute are defined by the XML Schema base64Binary datatype.

The following XML Schema fragment defines the contents of this element:

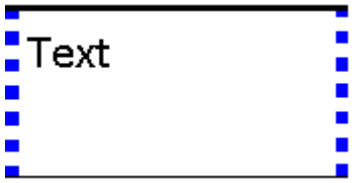
```
<complexType name="CT_Ink">
   <sequence/>
   <attribute name="i" type="xsd:base64Binary"/>
   <attribute name="annotation" type="ST_TrueFalse"/>
</complexType>
```

6.2.2.15 left (Text Box Left Stroke)

This element specifies the stroke properties for the left border of a text box. It entirely supercedes its parent stroke element if its on attribute is true. Thus the default value of an unspecified attribute overrides a value specified in the parent. If the on attribute is `false` or not specified, the border is not shown.

[*Example:* The text box borders are set independently. Note that the bottom border does not inherit the weight from the parent stroke element.

```
<v:stroke weight="2.25pt">
  <o:left v:ext="view" dashstyle="1 1" color="blue" weight="5pt" on="t"/>
  <o:top v:ext="view" color="black" weight="2.25pt" on="t"/>
  <o:right v:ext="view" dashstyle="1 1" color="blue" weight="5pt" on="t"/>
  <o:bottom v:ext="view" color="black" on="t"/>
  <o:column v:ext="view" color="#f60" on="t"/>
</v:stroke>
```

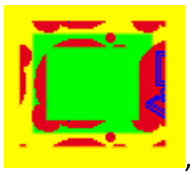


end example]

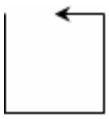
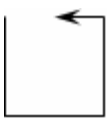
Parent Elements
background (§2.2.1); hdrShapeDefaults (§2.15.1.50); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23); shapeDefaults (§2.15.1.79); stroke (§6.1.2.21)

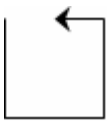
Attributes	Description
althref (Alternate Image Reference)	Specifies an alternate reference for an image in Macintosh PICT format. [*Example:* `<v:stroke ... althref="myimage.pcz" ... ></v:stroke>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
color (Stroke Color)	Specifies the stroke color. Overrides the strokecolor attribute of a shape. Default is `black`. See the fillcolor attribute for a list of supported named colors. [*Example:* The shape stroke is blue: `<v:shape ... strokecolor="red" ... ><v:stroke color="blue"/></v:shape>` *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.2.3.5).

color2 (Stroke Alternate Pattern Color)	Specifies a second color for strokes, used when filltype is `pattern`. Default is no value.
	
	When a pattern fill is used for the stroke, the stroke color is used in colored parts of the source image. The color2 defines an alternate color to use in place of black in the source image.
	
	[*Example:* This unusual example is intended to demonstrate how the image and colors interact to create a patterned stroke. The yellow background shows transparency. The non-square shape and square image create an effective offset. The heavy stroke weight shows more of the image. The green shape fill shows how the stroke is overlaid on the shape.
	
	`<v:background fillcolor="yellow"/>`
	`<v:shape style="width:60;height:50"`
	`strokecolor="red" fillcolor="lime"`
	`path="m 0,0 l 0,1000 1000,1000 1000,0 x e">`
	`<v:stroke filltype="pattern" weight="10pt"`
	`src="myimage.gif" color2="blue"/>`
	`</v:shape>`

Attributes	Description
	 , where myimage.gif is:  *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.2.3.5).

dashstyle (Stroke Dash Pattern)	Specifies the dot and dash pattern for a stroke. Default is `solid`. Pre-defined values are:
	
	• `solid`
	• `shortdash`
	• `shortdot`
	• `shortdashdot`
	• `shortdashdotdot`
	• `dot`
	• `dash`
	• `longdash`
	• `dashdot`
	• `longdashdot`
	• `longdashdotdot`
	
	A custom-defined dash pattern may also be specified using a series of numbers. These define the length of the dash (the drawn part of the stroke) and the length of the space between the dashes. The lengths are relative to the line width: a length of 1 is equal to the line width. The endcap style is applied to each dash but the arrow style is not. The string defines the length of the dash then the length of the space. This may be repeated to form complex dash styles. The string should always contain a pair of numbers; if it contains an odd number of numbers the last is disregarded. 0 implies a dot that is fourfold symmetrical (with round end caps, this is a circle).
	
	[*Example:*
	
	`<v:stroke dashstyle="0 2" weight="3pt" endcap="round"></v:stroke>`
	
	 
	
	`<v:stroke dashstyle="longdashdotdot" weight="2pt"></v:stroke>`

Attributes	Description
	 *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
endarrow (Line End Arrowhead)	Specifies an arrowhead for the end of a line. Default is none. Note that the path must not be closed with the x command for the arrowhead to show. Allowed values are: • none • block • classic • diamond • oval • open [*Example:* `<v:stroke endarrow="classic"/>`  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowType simple type (§6.1.3.9).
endarrowlength (Line End Arrowhead Length)	Specifies the length of the arrowhead at the end of a line. Default is medium. Allowed values are: • short • medium • long [*Example:* `<v:stroke ... endarrowlength="long" ... />`  *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_StrokeArrowLength simple type (§6.1.3.8).
endarrowwidth (Line End Arrowhead Width)	Specifies the width of the arrowhead at the end of a line. Default is `medium`. Allowed values are: • `narrow` • `medium` • `wide` [*Example:* `<v:stroke ... endarrowwidth="wide" ... />`  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowWidth simple type (§6.1.3.10).

endcap (Line End Cap)	Specifies the cap style for the end of a stroke. Default is `flat`. Allowed values are:
	• `flat`
	• `square`
	• `round`
	[*Example:*
	`<v:stroke ... endcap="round" weight="10pt" ... />`
	 
	`endcap="flat"`
	`endcap="square"`
	`endcap="round"`
	end example]

Attributes	Description
	The possible values for this attribute are defined by the ST_StrokeEndCap simple type (§6.1.3.11).
ext (VML Extension Handling Behavior) Namespace: urn:schemas-microsoft-com:vml	Specifies an optional value that indicates how applications that implement VML should interpret extensions not defined as part of the original specification of core VML. [*Rationale*: This part of the original VML specification is included to assist applications that leverage existing VML support in implementing the Office Open XML Format. *end rationale*] The possible values for this attribute are defined by the ST_Ext simple type (§6.1.3.3).

filltype (Stroke Image Style)	Specifies the type of fill used for the background of a stroke. Default is `solid`. Allowed values are:
	
	• `solid` - The fill pattern is solid.
	• `tile` - The fill image is tiled.
	• `pattern` - The fill image is stretched to form a pattern.
	• `frame` - The fill image becomes a border for the shape.
	
	[*Example:*
	
	`<v:shape style="width:50;height:50"`
	`  strokecolor="red"`
	`  path="m 0,0 l 0,1000 1000,1000 1000,0 x e">`
	`  <v:stroke filltype="frame" weight="10pt"`
	`  src="border.gif"/>`
	`</v:shape>`
	
	  , where border.gif is: 
	
	end example]
	
	The possible values for this attribute are defined by the ST_FillType simple type (§6.1.3.5).

forcedash (Force Dashed Outline)	Specifies whether a dashed outline is used to draw a shape when a shape has no line or fill. Default is `false`.
	
	Used by PresentationML placeholders to draw a dashed outline when there is no line and no fill for a shape.
	
	[*Example:*
	
	`<v:shape ... o:forcedash="true" ... >`
	`</v:shape>`

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
href (Original Image Reference)	Specifies the URL to the original image file. Used only if the picture has been linked and embedded. Default is no value. [*Example:* `<v:fill ... o:href="myimage.gif" ... ></v:fill>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
imagealignshape (Stoke Image Alignment)	Specifies the alignment of the stroke image. If `true`, the image is aligned with the shape. Otherwise, it is aligned with the containing scope. Default is `true`. [*Example:* The top position offset shifts the image alignment relative to the containing window: `<v:shape fillcolor="silver" style="top:20;width:50;height:50" path="m 0,0 l 0,1000 1000,1000 1000,0 x e"><v:stroke imagealignshape="false" weight="20pt" filltype="tile" src="myimage.gif"/></v:shape>`  `imagealignshape="false"`  `imagealignshape="false"` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
imageaspect	Specifies how the stroke image aspect ratio is preserved. Default is `ignore`. Allowed

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>(Stroke Image Aspect Ratio)</td><td>values are:

<table>
<tr><td>Value</td><td>Description</td></tr>
<tr><td><code>ignore</code></td><td>Ignore aspect issues.</td></tr>
<tr><td><code>atleast</code></td><td>Image is at least as big as imagesize.</td></tr>
<tr><td><code>atmost</code></td><td>Image is no bigger than imagesize.</td></tr>
</table>

[Example:

<pre>
<v:stroke filltype="frame" weight="10pt"
  src="border.gif" imagealignshape="true"
  imageaspect="atleast">
</v:stroke>
</pre>

<code>imagealignshape="ignore"</code>


<code>imagealignshape="atleast"</code>


<code>imagealignshape="atmost"</code>

end example]

The possible values for this attribute are defined by the ST_ImageAspect simple type (§6.1.3.6).</td></tr>
<tr><td>imagesize (Stroke Image Size)</td><td>Specifies the size of the image for the stroke. Default is the size of the image.

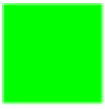
[Example:

<pre>
<v:stroke ... imagesize="10pt,10pt" ... />
</pre>
end example]

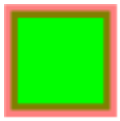
The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
<tr><td>insetpen (Inset Border From Path)</td><td>Specifies that the border shall be displayed inside of the path defining the shape, rather than along the path (the default border placement), or outside of the path as might be done with an image.

[Example:</td></tr>
</table>

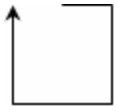
Attributes	Description
	<v:shape ... insetpen="true" ... > </v:shape> *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
joinstyle (Line End Join Style))	Specifies the join style for line ends. Default is round. • round • bevel • miter [*Example:* <v:polyline strokeweight="10pt" strokecolor="navy" points="10pt,10pt,50pt,50pt,90pt,10pt"> <v:stroke joinstyle="bevel"/> </v:polyline>  joinstyle="round" joinstyle="bevel" joinstyle="miter" *end example*] The possible values for this attribute are defined by the ST_StrokeJoinStyle simple type (§6.1.3.12).
linestyle (Stroke Line Style)	Specifies the line style of the stroke. Default is single. • single • thinThin • thinThick • thickThin • thickBetweenThin [*Example:*

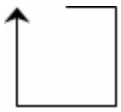
Attributes	Description
	<v:stroke linestyle="thickThin" weight="5pt"> </v:stroke>  *end example*] The possible values for this attribute are defined by the ST_StrokeLineStyle simple type (§6.1.3.13).
miterlimit (Miter Joint Limit)	Specifies the smoothness of the miter joint, or the maximum distance between the inner point and outer point of a joint. This number is a multiple of the thickness of the line. Default is 8. [*Example:* <v:stroke joinstyle="miter" weight="10pt" miterlimit="2"> </v:stroke>  *end example*] The possible values for this attribute are defined by the XML Schema decimal datatype.
on (Stroke Toggle)	Specifies whether the stroke is displayed. Default is true. This attribute overrides the shape's stroke attribute. [*Example:* <v:rect style="width:50;height:50" stroked="true" fillcolor="lime" strokecolor="red"> <v:stroke on="false" weight="5pt"/> </v:rect>  *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

Attributes	Description

opacity (Stroke Opacity)	Specifies the amount of transparency of a stroke. Default is 1.0.
	
	[*Example:*
	
	`<v:rect style="width:50;height:50"`
	`  fillcolor="lime" strokecolor="red">`
	`  <v:stroke weight="5pt" opacity="50%"/>`
	`</v:rect>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

src (Stroke Image Location)	Specifies the source image to load for a stroke fill. Default is no value.
	
	[*Example:*
	
	`<v:stroke ... src="myimage.gif" ... >`
	`</v:stroke>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

startarrow (Line Start Arrowhead)	Specifies an arrowhead for the start of a line. Default is none. Note that the path must not be closed with the x command for the arrowhead to show. Allowed values are: • none • block • classic • diamond • oval • open [*Example:* `<v:stroke startarrow="classic"/>`  *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_StrokeArrowType simple type (§6.1.3.9).
startarrowlength (Line Start Arrowhead Length)	Specifies the length of the arrowhead at the start of a line. Default is `medium`. Allowed values are: • `short` • `medium` • `long` [*Example:* `<v:stroke ... startarrowlength="long" ... />`  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowLength simple type (§6.1.3.8).
startarrowwidth (Line Start Arrowhead Width)	Specifies the width of the arrowhead at the start of a line. Default is `medium`. Allowed values are: • `narrow` • `medium` • `wide` [*Example:* `<v:stroke ... startarrowwidth="wide" ... />`  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowWidth simple type (§6.1.3.10).
title (Stroke Title)	Specifies the title of an embedded stroke image. This is typically set to the comment property of the image, which is often blank. [*Example:* `<v:fill ... o:title="alt text" ... >`

Attributes	Description
	</v:fill> *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
weight (Stroke Weight)	Specifies the thickness of a stroke. Default is 1. This attribute overrides the shape's strokeweight attribute. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_StrokeChild">
   <attributeGroup ref="v:AG_Ext"/>
   <attribute name="on" type="ST_TrueFalse" use="optional"/>
   <attribute name="weight" type="xsd:string" use="optional"/>
   <attribute name="color" type="ST_ColorType" use="optional"/>
   <attribute name="color2" type="ST_ColorType" use="optional"/>
   <attribute name="opacity" type="xsd:string" use="optional"/>
   <attribute name="linestyle" type="v:ST_StrokeLineStyle" use="optional"/>
   <attribute name="miterlimit" type="xsd:decimal" use="optional"/>
   <attribute name="joinstyle" type="v:ST_StrokeJoinStyle" use="optional"/>
   <attribute name="endcap" type="v:ST_StrokeEndCap" use="optional"/>
   <attribute name="dashstyle" type="xsd:string" use="optional"/>
   <attribute name="insetpen" type="ST_TrueFalse" use="optional"/>
   <attribute name="filltype" type="v:ST_FillType" use="optional"/>
   <attribute name="src" type="xsd:string" use="optional"/>
   <attribute name="imageaspect" type="v:ST_ImageAspect" use="optional"/>
   <attribute name="imagesize" type="xsd:string" use="optional"/>
   <attribute name="imagealignshape" type="ST_TrueFalse" use="optional"/>
   <attribute name="startarrow" type="v:ST_StrokeArrowType" use="optional"/>
   <attribute name="startarrowwidth" type="v:ST_StrokeArrowWidth" use="optional"/>
   <attribute name="startarrowlength" type="v:ST_StrokeArrowLength" use="optional"/>
   <attribute name="endarrow" type="v:ST_StrokeArrowType" use="optional"/>
   <attribute name="endarrowwidth" type="v:ST_StrokeArrowWidth" use="optional"/>
   <attribute name="endarrowlength" type="v:ST_StrokeArrowLength" use="optional"/>
   <attribute ref="href"/>
   <attribute ref="althref"/>
   <attribute ref="title"/>
   <attribute ref="forcedash"/>
</complexType>
```

6.2.2.16 LinkType (Embedded Object Alternate Image Request)

This element specifies the type of image which shall be requested from an embedded object's host application when the contents of a linked image are updated within a document. When linked images are stored in documents, the only items stored in the document are an image representation and a link to the source. This element specifies the type of image which shall be requested from the source on update.

[*Note*: The formats available may vary based on the type of embedded object - this information is typically queried from the embedded object's application before it is stored. This setting may be omitted, and is usually stored for performance reasons, so it is not queried on each update of the linked object. *end note*]

The possible values for this element are defined by the ST_OLELinkType simple type (§6.2.3.17).

Parent Elements
OLEObject (§6.2.2.19)

6.2.2.17 lock (Shape Protections)

This element specifies locks against actions that can be effected in the UI of an authoring application or programmatically through an object model.

[*Example:* The following snippet locks the shape's aspect ratio and text from user edits.

```
<v:shape ... >
  <o:lock v:ext="edit" aspectratio="t" text="t"/>
</v:shape>
```

end example]

Parent Elements
arc (§6.1.2.1); background (§2.2.1); curve (§6.1.2.3); group (§6.1.2.7); hdrShapeDefaults (§2.15.1.50); image (§6.1.2.10); line (§6.1.2.12); object (§2.3.3.19); oval (§6.1.2.13); pict (§2.3.3.21); pict (§2.9.23); polyline (§6.1.2.15); rect (§6.1.2.16); roundrect (§6.1.2.17); shape (§6.1.2.19); shapedefaults (§6.2.2.27); shapeDefaults (§2.15.1.79); shapetype (§6.1.2.20)

Attributes	Description
adjusthandles (Handles Lock)	Specifies whether the handles of a shape are locked from being edited. Default is `false`. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
aspectratio (Aspect Ratio Lock)	Specifies whether the aspect ratio of a shape is locked from being edited. Default is `false`. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
cropping (Cropping Lock)	Specifies whether cropping of a shape is locked from being edited. Default is `false`. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
ext (VML Extension Handling Behavior)	Specifies an optional value that indicates how applications that implement VML should interpret extensions not defined as part of the original specification of core VML.

Attributes	Description
Namespace: urn:schemas-microsoft-com:vml	[*Rationale*: This part of the original VML specification is included to assist applications that leverage existing VML support in implementing the Office Open XML Format. *end rationale*] The possible values for this attribute are defined by the ST_Ext simple type (§6.1.3.3).
grouping (Grouping Lock)	Specifies whether a shape is locked from being grouped. Default is `false`. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
position (Position Lock)	Specifies whether the position of a shape is locked from being edited. Default is `false`. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
rotation (Rotation Lock)	Specifies whether the rotation of a shape is locked from being edited. Default is `false`. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
selection (Selection Lock)	Specifies whether the shape is locked from being selectable in an editor. Default is `false`. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
shapetype (AutoShape Type Lock)	Specifies whether the AutoShape type is locked from being edited. Default is `false`. If `true`, the type of an AutoShape cannot be changed in a graphical editor. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
text (Text Lock)	Specifies whether the text attached to a shape is locked from being edited. Default is `false`. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
ungrouping (Ungrouping Lock)	Specifies whether a grouped shape is locked from being ungrouped. Default is `false`. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
verticies (Vertices Lock)	Specifies whether the vertices of a path are locked from being edited. Default is `false`. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Lock">
   <attributeGroup ref="v:AG_Ext"/>
   <attribute name="position" type="ST_TrueFalse" use="optional"/>
   <attribute name="selection" type="ST_TrueFalse" use="optional"/>
   <attribute name="grouping" type="ST_TrueFalse" use="optional"/>
   <attribute name="ungrouping" type="ST_TrueFalse" use="optional"/>
   <attribute name="rotation" type="ST_TrueFalse" use="optional"/>
   <attribute name="cropping" type="ST_TrueFalse" use="optional"/>
   <attribute name="verticies" type="ST_TrueFalse" use="optional"/>
   <attribute name="adjusthandles" type="ST_TrueFalse" use="optional"/>
   <attribute name="text" type="ST_TrueFalse" use="optional"/>
   <attribute name="aspectratio" type="ST_TrueFalse" use="optional"/>
   <attribute name="shapetype" type="ST_TrueFalse" use="optional"/>
</complexType>
```

6.2.2.18 LockedField (Embedded Object Cannot Be Refreshed)

This element specifies that the embedded object's appearance is locked - that is, that the object's current representation shall be locked to prevent any user interaction or automatic application behavior from modifying its contents.

This element shall contain no content - its presence indicates that the embedded object is locked, and its omission allows the field to be updated.

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.2.3.24).

Parent Elements
OLEObject (§6.2.2.19)

6.2.2.19 OLEObject (Embedded OLE Object)

This element specifies an OLE object.

[*Example:* The following demonstrates a video file embedded in a WordprocessingML document:

```
<w:object ... >
  <v:shape id="_x0000_i1025" type="#_x0000_t75"
  style="width:1in;height:24pt" o:ole="">
  <v:imagedata r:id="rId4" o:title=""/>
  </v:shape>
  <o:OLEObject Type="Embed" ProgID="AVIFile" ShapeID="_x0000_i1025"
  DrawAspect="Content" ObjectID="_1219561732" r:id="rId5"/>
</w:object>
```

end example]

Parent Elements
background (§2.2.1); hdrShapeDefaults (§2.15.1.50); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23);

Parent Elements
shapeDefaults (§2.15.1.79)

Child Elements	Subclause
FieldCodes (WordprocessingML Field Switches)	§6.2.2.11
LinkType (Embedded Object Alternate Image Request)	§6.2.2.16
LockedField (Embedded Object Cannot Be Refreshed)	§6.2.2.18

Attributes	Description

DrawAspect (OLE Object Representation)	Specifies how the object is represented visually in the application.
	
	[*Example:*
	
	`<o:OLEObject ... DrawAspect="Content">`
	`</o:OLEObject>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_OLEDrawAspect simple type (§6.2.3.16).

id (Relationship)
 	
Namespace: .../officeDocument/2006/relationships
	Specifies the actual OLE object using a standard part relationship lookup.
	
	[*Example:*
	
	`<o:OLEObject ... r:id="rId5">`
	`</o:OLEObject>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_RelationshipId simple type (§7.8.2.1).

ObjectID (OLE Object Unique ID)	Specifies a unique ID identifying the OLE object.
	
	[*Example:*
	
	`<o:OLEObject ... ObjectID="_1219561732">`
	`</o:OLEObject>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

ProgID (OLE Object	Specifies the OLE server application associated with the OLE object.

Attributes	Description

Application)	[*Example:*
	
	`<o:OLEObject ... ProgID="AVIFile">`
	`</o:OLEObject>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

ShapeID (OLE Object Shape)	Specifies the shape with which the OLE object is associated. A VML shape provides the visual placeholder for an OLE object and this attribute is set to the id of the placeholder shape.
	
	[*Example:*
	
	`<o:OLEObject ... ShapeID="_x0000_i1025">`
	`</o:OLEObject>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

Type (OLE Object Type)	Specifies the type of the OLE connection.
	
	[*Example:*
	
	`<o:OLEObject ... Type="Embed">`
	`</o:OLEObject>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_OLEType simple type (§6.2.3.18).

UpdateMode (OLE Update Mode)	Specifies how the object is updated with new data if the Type is `Link` - automatically or on-demand by the user.
	
	[*Example:*
	
	`<o:OLEObject ... UpdateMode="Always">`
	`</o:OLEObject>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_OLEUpdateMode simple type (§6.2.3.19).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OLEObject">
   <sequence>
      <element name="LinkType" type="ST_OLELinkType" minOccurs="0"/>
      <element name="LockedField" type="ST_TrueFalseBlank" minOccurs="0"/>
      <element name="FieldCodes" type="xsd:string" minOccurs="0"/>
   </sequence>
   <attribute name="Type" type="ST_OLEType" use="optional"/>
   <attribute name="ProgID" type="xsd:string" use="optional"/>
   <attribute name="ShapeID" type="xsd:string" use="optional"/>
   <attribute name="DrawAspect" type="ST_OLEDrawAspect" use="optional"/>
   <attribute name="ObjectID" type="xsd:string" use="optional"/>
   <attribute ref="r:id" use="optional"/>
   <attribute name="UpdateMode" type="ST_OLEUpdateMode" use="optional"/>
</complexType>
```

6.2.2.20 proxy (Shape Reference)

This element specifies an entry in a r element rule that contains a reference to one or more shapes that are participating in the rule.

[*Example:* The following rule defines a connection between two shapes. The shape with id _s1036 connects shape _s1033 to _s1032:

```
<o:shapelayout v:ext="edit">
  <o:rules v:ext="edit">
    <o:r id="V:Rule1" type="connector" idref="#_s1036">
      <o:proxy start="" idref="#_s1033" connectloc="0"/>
      <o:proxy end="" idref="#_s1032" connectloc="2"/>
    </o:r>
  </o:rules>
</o:shapelayout>
```

end example]

Parent Elements
r (§6.2.2.21)

Attributes	Description
connectloc (Connection Location)	Specifies the location on the shape where the connector is attached. The value is an index into the list of connection points defined in the shape - see the connectlocs attribute. Default is 0. Only used in a connector rule. The possible values for this attribute are defined by the XML Schema int datatype.
end (End Point Connection Flag)	Specifies whether the connector's end point is connected to the shape. Default is `false`. Only used in a connector rule.

Attributes	Description
	The possible values for this attribute are defined by the ST_TrueFalseBlank simple type (§6.2.3.24).
idref (Proxy Shape Reference)	Specifies a reference to a shape in the current document. Default is no value. A shape name is used as the reference mechanism; this is not a relationship ID. This attribute indicates that the referenced shape is part of this rule. Two or more proxy elements are used for an alignment rule. A connector rule uses one or two, indicating which shapes the connector is attached to. The possible values for this attribute are defined by the XML Schema string datatype.
start (Start Point Connection Flag)	Specifies whether the connector's start point is connected to the shape. Default is `false`. Only used in a connector rule. If both start and end are specified the later one takes precedence. The possible values for this attribute are defined by the ST_TrueFalseBlank simple type (§6.2.3.24).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Proxy">
   <attribute name="start" type="ST_TrueFalseBlank" use="optional" default="false"/>
   <attribute name="end" type="ST_TrueFalseBlank" use="optional" default="false"/>
   <attribute name="idref" type="xsd:string" use="optional"/>
   <attribute name="connectloc" type="xsd:int" use="optional"/>
</complexType>
```

6.2.2.21 r (Rule)

This element specifies a rule entry in a rules element rule set that describes how a certain shape or set of shapes behaves during editing.

[*Example:* The following rule defines a connection between two shapes. The shape with id _s1036 connects shape _s1033 to _s1032:

```
<o:shapelayout v:ext="edit">
  <o:rules v:ext="edit">
    <o:r id="V:Rule1" type="connector" idref="#_s1036">
      <o:proxy start="" idref="#_s1033" connectloc="0"/>
      <o:proxy end="" idref="#_s1032" connectloc="2"/>
    </o:r>
  </o:rules>
</o:shapelayout>
```

end example]

Parent Elements
rules (§6.2.2.26)

Child Elements	Subclause
proxy (Shape Reference)	§6.2.2.20

Attributes	Description
how (Alignment Rule Type)	Specifies the type of alignment for an alignment rule. Default is no value. Allowed values are: • top • middle • bottom • left • center • right The possible values for this attribute are defined by the ST_How simple type (§6.2.3.13).
id (Rule ID)	Specifies an identifier for the rule. Default is no value. The possible values for this attribute are defined by the XML Schema string datatype.
idref (Rule Shape Reference)	Specifies a reference to a shape in the current document that is the primary shape in the rule. For example, for a connector rule, the connector. Default is no value. A shape name is used as the reference mechanism; this is not a relationship ID. The possible values for this attribute are defined by the XML Schema string datatype.
type (Rule Type)	Specifies the type of the rule. Default is no value. Allowed values are: • arc • callout • connector • align The possible values for this attribute are defined by the ST_RType simple type (§6.2.3.21).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_R">
   <sequence>
      <element name="proxy" type="CT_Proxy" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="id" type="xsd:string" use="required"/>
   <attribute name="type" type="ST_RType" use="optional"/>
   <attribute name="how" type="ST_How" use="optional"/>
   <attribute name="idref" type="xsd:string" use="optional"/>
</complexType>
```

6.2.2.22 regrouptable (Shape Grouping History)

This element specifies a list of entries which describe how shapes were previously grouped so they can be regrouped. The regroupid attribute of shapes indicates which shapes belong together when a regroup is performed. The regrouptable tracks the previous regroupid that should be assigned to all shapes with the given current regroupid.

[*Example:* Consider a document containing two rectangles and a circle. The rectangles are grouped together, then that group is grouped with the circle. This new group is then ungrouped, leaving the circle and grouped rectangles. The document might contain the following snippets:

```
<v:oval ... o:regroupid="1"/>
<v:group ... o:regroupid="1"/>
  <v:rect ... />
  <v:rect ... />
</v:group>

<o:regrouptable v:ext="edit">
  <o:entry new="1" old="0"/>
</o:regrouptable>
```

The regroupid attribute indicates that the shapes with regroupid 1 were previously grouped together. The entry indicates that if those shapes are regrouped, the new group formed should not have a regroupid value as it was not previously ungrouped.

If the two rectangles are ungrouped, the document reflects that the rectangles were previously grouped and that their old group was previously grouped:

```
<v:oval ... o:regroupid="1"/>
<v:rect ... o:regroupid="2"/>
<v:rect ... o:regroupid="2"/>

<o:regrouptable v:ext="edit">
  <o:entry new="1" old="0"/>
  <o:entry new="2" old="1"/>
</o:regrouptable>
```

end example]

Parent Elements
shapelayout (§6.2.2.28)

Child Elements	Subclause
entry (Regroup Entry)	§6.2.2.9

Attributes	Description
ext (VML Extension Handling Behavior) Namespace: urn:schemas-microsoft-com:vml	Specifies an optional value that indicates how applications that implement VML should interpret extensions not defined as part of the original specification of core VML. [*Rationale*: This part of the original VML specification is included to assist applications that leverage existing VML support in implementing the Office Open XML Format. *end rationale*] The possible values for this attribute are defined by the ST_Ext simple type (§6.1.3.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RegroupTable">
   <sequence>
      <element name="entry" type="CT_Entry" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attributeGroup ref="v:AG_Ext"/>
</complexType>
```

6.2.2.23 rel (Diagram Relationship)

This element specifies a relationship between two diagram nodes. An optional third node that exists between the primary two may also be included. The relationship has an implicit order since it describes the source and destination nodes.

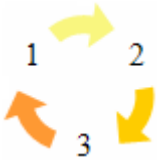
[*Example:* In the cycle diagram below, shape 1036 (the shape that is the text box for the text "2") is the first node. A relationship exists between shape 1036 and shape 1044 (the text box containing "1"). In between those shapes is shape 1038 (the yellow arrow).

```
<o:relationtable v:ext="edit">
  <o:rel v:ext="edit" idsrc="#_s1036" iddest="#_s1036"/>
  <o:rel v:ext="edit" idsrc="#_s1042" iddest="#_s1036" idcntr="#_s1043"/>
  <o:rel v:ext="edit" idsrc="#_s1044" iddest="#_s1042" idcntr="#_s1045"/>
  <o:rel v:ext="edit" idsrc="#_s1036" iddest="#_s1044" idcntr="#_s1038"/>
</o:relationtable>
```

```
<v:rect id="_s1036" ... >
  <v:textbox ... ><...>2</...></v:textbox>
</v:rect>

<v:rect id="_s1044" ... >
  <v:textbox ... ><...>1</...></v:textbox>
</v:rect>

<v:shape id="_s1038" ... />
```



end example]

Parent Elements
relationtable (§6.2.2.24)

Attributes	Description
ext (VML Extension Handling Behavior) Namespace: urn:schemas-microsoft-com:vml	Specifies an optional value that indicates how applications that implement VML should interpret extensions not defined as part of the original specification of core VML. [*Rationale*: This part of the original VML specification is included to assist applications that leverage existing VML support in implementing the Office Open XML Format. *end rationale*] The possible values for this attribute are defined by the ST_Ext simple type (§6.1.3.3).

idcntr (Diagram Relationship Center Shape)	Specifies the optional identifier of the shape that exists between the source and destination shapes. This is omitted if the relationship does not have a shape between the source and destination shapes.
	
	[*Example:*
	
	`<o:rel ... idcntr="#s_1038">`
	`</o:rel>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

iddest (Diagram Relationship	Specifies the identifier of the shape at the destination of the relationship.

Attributes	Description

Destination Shape)	[*Example:*
	
	`<o:rel ... iddest="#s_1044">`
	`</o:rel>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

idsrc (Diagram Relationship Source Shape)	Specifies the identifier of the shape at the source of the relationship.
	
	[*Example:*
	
	`<o:rel ... idsrc="#s_1036">`
	`</o:rel>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Relation">
   <attributeGroup ref="v:AG_Ext"/>
   <attribute name="idsrc" type="xsd:string" use="optional"/>
   <attribute name="iddest" type="xsd:string" use="optional"/>
   <attribute name="idcntr" type="xsd:string" use="optional"/>
</complexType>
```

6.2.2.24 relationtable (Diagram Relationship Table)

This element specifies a list that describes the relationships among diagram nodes.

[*Example:* The following table describes the parent-child relationships for shapes in an organization chart. The first entry describes the top-level shape in the diagram. The next two rows describe that the shapes are subordinates to the first shape. Shape 1029 is a subordinate of shape 1028. Shape 1032, a connector in this case, is in between the two.

```
<o:relationtable v:ext="edit">
  <o:rel v:ext="edit" idsrc="#_s1028" iddest="#_s1028"/>
  <o:rel v:ext="edit" idsrc="#_s1029" iddest="#_s1028" idcntr="#_s1032"/>
  <o:rel v:ext="edit" idsrc="#_s1030" iddest="#_s1028" idcntr="#_s1033"/>
</o:relationtable>
```

end example]

Parent Elements
diagram (§6.2.2.8)

Child Elements	Subclause
rel (Diagram Relationship)	§6.2.2.23

Attributes	Description
ext (VML Extension Handling Behavior) Namespace: urn:schemas-microsoft-com:vml	Specifies an optional value that indicates how applications that implement VML should interpret extensions not defined as part of the original specification of core VML. [*Rationale*: This part of the original VML specification is included to assist applications that leverage existing VML support in implementing the Office Open XML Format. *end rationale*] The possible values for this attribute are defined by the ST_Ext simple type (§6.1.3.3).

The following XML Schema fragment defines the contents of this element:

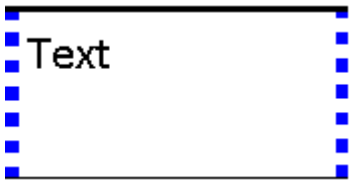
```
<complexType name="CT_RelationTable">
   <sequence>
      <element name="rel" type="CT_Relation" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attributeGroup ref="v:AG_Ext"/>
</complexType>
```

6.2.2.25 right (Text Box Right Stroke)

This element specifies the stroke properties for the right border of a text box. It entirely supercedes its parent stroke element if its on attribute is true. Thus the default value of an unspecified attribute overrides a value specified in the parent. If the on attribute is `false` or not specified, the border is not shown.

[*Example:* The text box borders are set independently. Note that the bottom border does not inherit the weight from the parent stroke element.

```
<v:stroke weight="2.25pt">
  <o:left v:ext="view" dashstyle="1 1" color="blue" weight="5pt" on="t"/>
  <o:top v:ext="view" color="black" weight="2.25pt" on="t"/>
  <o:right v:ext="view" dashstyle="1 1" color="blue" weight="5pt" on="t"/>
  <o:bottom v:ext="view" color="black" on="t"/>
  <o:column v:ext="view" color="#f60" on="t"/>
</v:stroke>
```

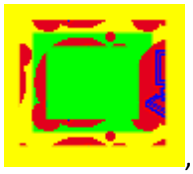


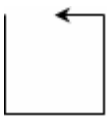
end example]

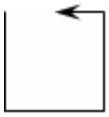
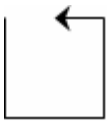
Parent Elements
background (§2.2.1); hdrShapeDefaults (§2.15.1.50); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23); shapeDefaults (§2.15.1.79); stroke (§6.1.2.21)

Attributes	Description
althref (Alternate Image Reference)	Specifies an alternate reference for an image in Macintosh PICT format. [*Example:* `<v:stroke ... althref="myimage.pcz" ... ></v:stroke>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
color (Stroke Color)	Specifies the stroke color. Overrides the strokecolor attribute of a shape. Default is `black`. See the fillcolor attribute for a list of supported named colors. [*Example:* The shape stroke is blue: `<v:shape ... strokecolor="red" ... ><v:stroke color="blue"/></v:shape>` *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.2.3.5).

color2 (Stroke Alternate Pattern Color)	Specifies a second color for strokes, used when filltype is `pattern`. Default is no value.
	
	When a pattern fill is used for the stroke, the stroke color is used in colored parts of the source image. The color2 defines an alternate color to use in place of black in the source image.
	
	[*Example:* This unusual example is intended to demonstrate how the image and colors interact to create a patterned stroke. The yellow background shows transparency. The non-square shape and square image create an effective offset. The heavy stroke weight shows more of the image. The green shape fill shows how the stroke is overlaid on the shape.
	
	`<v:background fillcolor="yellow"/>`
	`<v:shape style="width:60;height:50"`
	`strokecolor="red" fillcolor="lime"`

Attributes	Description
	path="m 0,0 l 0,1000 1000,1000 1000,0 x e"> <v:stroke filltype="pattern" weight="10pt" src="myimage.gif" color2="blue"/> </v:shape>  , where myimage.gif is:  *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.2.3.5).
dashstyle (Stroke Dash Pattern)	Specifies the dot and dash pattern for a stroke. Default is `solid`. Pre-defined values are: • `solid` • `shortdash` • `shortdot` • `shortdashdot` • `shortdashdotdot` • `dot` • `dash` • `longdash` • `dashdot` • `longdashdot` • `longdashdotdot` A custom-defined dash pattern may also be specified using a series of numbers. These define the length of the dash (the drawn part of the stroke) and the length of the space between the dashes. The lengths are relative to the line width: a length of 1 is equal to the line width. The endcap style is applied to each dash but the arrow style is not. The string defines the length of the dash then the length of the space. This may be repeated to form complex dash styles. The string should always contain a pair of numbers; if it contains an odd number of numbers the last is disregarded. 0 implies a dot that is fourfold symmetrical (with round end caps, this is a circle). [*Example:* <v:stroke dashstyle="0 2" weight="3pt" endcap="round"> </v:stroke> 

Attributes	Description
	<v:stroke dashstyle="longdashdotdot" weight="2pt"> </v:stroke>  *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
endarrow (Line End Arrowhead)	Specifies an arrowhead for the end of a line. Default is none. Note that the path must not be closed with the x command for the arrowhead to show. Allowed values are: • none • block • classic • diamond • oval • open [*Example:* <v:stroke endarrow="classic"/>  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowType simple type (§6.1.3.9).
endarrowlength (Line End Arrowhead Length)	Specifies the length of the arrowhead at the end of a line. Default is medium. Allowed values are: • short • medium • long [*Example:* <v:stroke ... endarrowlength="long" ... />

Attributes	Description
	 *end example*] The possible values for this attribute are defined by the ST_StrokeArrowLength simple type (§6.1.3.8).
endarrowwidth (Line End Arrowhead Width)	Specifies the width of the arrowhead at the end of a line. Default is `medium`. Allowed values are: • `narrow` • `medium` • `wide` [*Example:* `<v:stroke ... endarrowwidth="wide" ... />`  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowWidth simple type (§6.1.3.10).
endcap (Line End Cap)	Specifies the cap style for the end of a stroke. Default is `flat`. Allowed values are: • `flat` • `square` • `round` [*Example:* `<v:stroke ... endcap="round" weight="10pt" ... />`  `endcap="flat"`  `endcap="square"`

Attributes	Description
	 `endcap="round"` *end example*] The possible values for this attribute are defined by the ST_StrokeEndCap simple type (§6.1.3.11).
ext (VML Extension Handling Behavior) Namespace: urn:schemas-microsoft-com:vml	Specifies an optional value that indicates how applications that implement VML should interpret extensions not defined as part of the original specification of core VML. [*Rationale*: This part of the original VML specification is included to assist applications that leverage existing VML support in implementing the Office Open XML Format. *end rationale*] The possible values for this attribute are defined by the ST_Ext simple type (§6.1.3.3).

filltype (Stroke Image Style)	Specifies the type of fill used for the background of a stroke. Default is `solid`. Allowed values are:
	
	• `solid` - The fill pattern is solid.
	• `tile` - The fill image is tiled.
	• `pattern` - The fill image is stretched to form a pattern.
	• `frame` - The fill image becomes a border for the shape.
	
	[*Example:*
	
	`<v:shape style="width:50;height:50"`
	`  strokecolor="red"`
	`  path="m 0,0 l 0,1000 1000,1000 1000,0 x e">`
	`  <v:stroke filltype="frame" weight="10pt"`
	`  src="border.gif"/>`
	`</v:shape>`
	
	  , where border.gif is: 
	
	end example]
	
	The possible values for this attribute are defined by the ST_FillType simple type (§6.1.3.5).

forcedash (Force Dashed Outline)	Specifies whether a dashed outline is used to draw a shape when a shape has no line or fill. Default is `false`.

Attributes	Description
	Used by PresentationML placeholders to draw a dashed outline when there is no line and no fill for a shape. [*Example:* `<v:shape ... o:forcedash="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
href (Original Image Reference)	Specifies the URL to the original image file. Used only if the picture has been linked and embedded. Default is no value. [*Example:* `<v:fill ... o:href="myimage.gif" ... ></v:fill>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
imagealignshape (Stoke Image Alignment)	Specifies the alignment of the stroke image. If `true`, the image is aligned with the shape. Otherwise, it is aligned with the containing scope. Default is `true`. [*Example:* The top position offset shifts the image alignment relative to the containing window: `<v:shape fillcolor="silver" style="top:20;width:50;height:50" path="m 0,0 l 0,1000 1000,1000 1000,0 x e"> <v:stroke imagealignshape="false" weight="20pt" filltype="tile" src="myimage.gif"/> </v:shape>`  `imagealignshape="false"`  `imagealignshape="false"`

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>end example]

The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).</td></tr>
<tr><td>imageaspect (Stroke Image Aspect Ratio)</td><td>Specifies how the stroke image aspect ratio is preserved. Default is <code>ignore</code>. Allowed values are:

<table>
<tr><td>Value</td><td>Description</td></tr>
<tr><td><code>ignore</code></td><td>Ignore aspect issues.</td></tr>
<tr><td><code>atleast</code></td><td>Image is at least as big as imagesize.</td></tr>
<tr><td><code>atmost</code></td><td>Image is no bigger than imagesize.</td></tr>
</table>

[Example:

<pre><v:stroke filltype="frame" weight="10pt"
  src="border.gif" imagealignshape="true"
  imageaspect="atleast">
</v:stroke></pre>
  <code>imagealignshape="ignore"</code>
  <code>imagealignshape="atleast"</code>
  <code>imagealignshape="atmost"</code>

end example]

The possible values for this attribute are defined by the ST_ImageAspect simple type (§6.1.3.6).</td></tr>
<tr><td>imagesize (Stroke Image Size)</td><td>Specifies the size of the image for the stroke. Default is the size of the image.

[Example:

<pre><v:stroke ... imagesize="10pt,10pt" ... /></pre>
end example]

The possible values for this attribute are defined by the XML Schema string datatype.</td></tr>
</table>

Attributes	Description
insetpen (Inset Border From Path)	Specifies that the border shall be displayed inside of the path defining the shape, rather than along the path (the default border placement), or outside of the path as might be done with an image. [*Example:* `<v:shape ... insetpen="true" ... ></v:shape>` *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

joinstyle (Line End Join Style))	Specifies the join style for line ends. Default is `round`.
	
	• `round`
	• `bevel`
	• `miter`
	
	[*Example:*
	
	`<v:polyline strokeweight="10pt" strokecolor="navy" points="10pt,10pt,50pt,50pt,90pt,10pt"><v:stroke joinstyle="bevel"/></v:polyline>`
	
	 
	`joinstyle="round"`
	`joinstyle="bevel"`
	`joinstyle="miter"`
	
	end example]
	
	The possible values for this attribute are defined by the ST_StrokeJoinStyle simple type (§6.1.3.12).

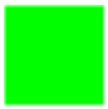
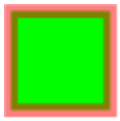
linestyle (Stroke Line Style)	Specifies the line style of the stroke. Default is `single`. • `single`

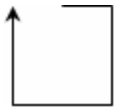
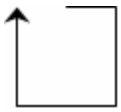
Attributes	Description

	• thinThin
	• thinThick
	• thickThin
	• thickBetweenThin
	
	[*Example:*
	
	`<v:stroke linestyle="thickThin" weight="5pt">`
	`</v:stroke>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_StrokeLineStyle simple type (§6.1.3.13).

miterlimit (Miter Joint Limit)	Specifies the smoothness of the miter joint, or the maximum distance between the inner point and outer point of a joint. This number is a multiple of the thickness of the line. Default is 8.
	
	[*Example:*
	
	`<v:stroke joinstyle="miter" weight="10pt"`
	`  miterlimit="2">`
	`</v:stroke>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema decimal datatype.

on (Stroke Toggle)	Specifies whether the stroke is displayed. Default is `true`. This attribute overrides the shape's stroke attribute.
	
	[*Example:*
	
	`<v:rect style="width:50;height:50" stroked="true"`
	`  fillcolor="lime" strokecolor="red">`
	`  <v:stroke on="false" weight="5pt"/>`
	`</v:rect>`

Attributes	Description
	 *end example*] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
opacity (Stroke Opacity)	Specifies the amount of transparency of a stroke. Default is 1.0. [*Example:* `<v:rect style="width:50;height:50" fillcolor="lime" strokecolor="red"> <v:stroke weight="5pt" opacity="50%"/> </v:rect>`  *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
src (Stroke Image Location)	Specifies the source image to load for a stroke fill. Default is no value. [*Example:* `<v:stroke ... src="myimage.gif" ... > </v:stroke>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
startarrow (Line Start Arrowhead)	Specifies an arrowhead for the start of a line. Default is none. Note that the path must not be closed with the x command for the arrowhead to show. Allowed values are: • none • block • classic • diamond • oval • open [*Example:*

Attributes	Description
	<v:stroke startarrow="classic"/>  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowType simple type (§6.1.3.9).
startarrowlength (Line Start Arrowhead Length)	Specifies the length of the arrowhead at the start of a line. Default is `medium`. Allowed values are: • `short` • `medium` • `long` [*Example:* <v:stroke ... startarrowlength="long" ... />  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowLength simple type (§6.1.3.8).
startarrowwidth (Line Start Arrowhead Width)	Specifies the width of the arrowhead at the start of a line. Default is `medium`. Allowed values are: • `narrow` • `medium` • `wide` [*Example:* <v:stroke ... startarrowwidth="wide" ... />  *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_StrokeArrowWidth simple type (§6.1.3.10).
title (Stroke Title)	Specifies the title of an embedded stroke image. This is typically set to the comment property of the image, which is often blank. [*Example:* `<v:fill ... o:title="alt text" ... ></v:fill>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
weight (Stroke Weight)	Specifies the thickness of a stroke. Default is 1. This attribute overrides the shape's strokeweight attribute. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_StrokeChild">
   <attributeGroup ref="v:AG_Ext"/>
   <attribute name="on" type="ST_TrueFalse" use="optional"/>
   <attribute name="weight" type="xsd:string" use="optional"/>
   <attribute name="color" type="ST_ColorType" use="optional"/>
   <attribute name="color2" type="ST_ColorType" use="optional"/>
   <attribute name="opacity" type="xsd:string" use="optional"/>
   <attribute name="linestyle" type="v:ST_StrokeLineStyle" use="optional"/>
   <attribute name="miterlimit" type="xsd:decimal" use="optional"/>
   <attribute name="joinstyle" type="v:ST_StrokeJoinStyle" use="optional"/>
   <attribute name="endcap" type="v:ST_StrokeEndCap" use="optional"/>
   <attribute name="dashstyle" type="xsd:string" use="optional"/>
   <attribute name="insetpen" type="ST_TrueFalse" use="optional"/>
   <attribute name="filltype" type="v:ST_FillType" use="optional"/>
   <attribute name="src" type="xsd:string" use="optional"/>
   <attribute name="imageaspect" type="v:ST_ImageAspect" use="optional"/>
   <attribute name="imagesize" type="xsd:string" use="optional"/>
   <attribute name="imagealignshape" type="ST_TrueFalse" use="optional"/>
   <attribute name="startarrow" type="v:ST_StrokeArrowType" use="optional"/>
   <attribute name="startarrowwidth" type="v:ST_StrokeArrowWidth" use="optional"/>
   <attribute name="startarrowlength" type="v:ST_StrokeArrowLength" use="optional"/>
   <attribute name="endarrow" type="v:ST_StrokeArrowType" use="optional"/>
   <attribute name="endarrowwidth" type="v:ST_StrokeArrowWidth" use="optional"/>
   <attribute name="endarrowlength" type="v:ST_StrokeArrowLength" use="optional"/>
   <attribute ref="href"/>
   <attribute ref="althref"/>
   <attribute ref="title"/>
   <attribute ref="forcedash"/>
</complexType>
```

6.2.2.26 rules (Rule Set)

This element specifies a list of rule entries which describe how a certain shape or sets of shapes should behave during editing.

[*Example:* The following rule defines a connection between two shapes. The shape with id _s1036 connects shape _s1033 to _s1032:

```
<o:shapelayout v:ext="edit">
  <o:rules v:ext="edit">
    <o:r id="V:Rule1" type="connector" idref="#_s1036">
      <o:proxy start="" idref="#_s1033" connectloc="0"/>
      <o:proxy end="" idref="#_s1032" connectloc="2"/>
    </o:r>
  </o:rules>
</o:shapelayout>
```

end example]

Parent Elements
shapelayout (§6.2.2.28)

Child Elements	Subclause
r (Rule)	§6.2.2.21

Attributes	Description
ext (VML Extension Handling Behavior) Namespace: urn:schemas-microsoft-com:vml	Specifies an optional value that indicates how applications that implement VML should interpret extensions not defined as part of the original specification of core VML. [*Rationale*: This part of the original VML specification is included to assist applications that leverage existing VML support in implementing the Office Open XML Format. *end rationale*] The possible values for this attribute are defined by the ST_Ext simple type (§6.1.3.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Rules">
   <sequence>
      <element name="r" type="CT_R" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attributeGroup ref="v:AG_Ext"/>
</complexType>
```

6.2.2.27 shapedefaults (New Shape Defaults)

This element specifies the defaults that are used when creating new shapes. These defaults are stored once per document.

[*Example:* Consider a case in which an application chooses to store the highest shape ID it has used in the document thus far. This could be used to support the generation of new shape IDs:

```
<o:shapedefaults v:ext="edit" spidmax="1029"/>
```

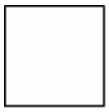
end example]

Parent Elements
background (§2.2.1); hdrShapeDefaults (§2.15.1.50); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23); shapeDefaults (§2.15.1.79)

Child Elements	Subclause
callout (Callout)	§6.2.2.2
colormenu (UI Default Colors)	§6.2.2.4
colormru (Most Recently Used Colors)	§6.2.2.5
extrusion (3D Extrusion)	§6.2.2.10
fill (Shape Fill Properties)	§6.1.2.5
lock (Shape Protections)	§6.2.2.17
shadow (Shadow Effect)	§6.1.2.18
skew (Skew Transform)	§6.2.2.30
stroke (Line Stroke Settings)	§6.1.2.21
textbox (Text Box)	§6.1.2.22

Attributes	Description
allowincell (Allow in Table Cell)	Specifies whether the shape is allowed to be placed in a table cell. Default is `false`. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
ext (VML Extension Handling Behavior) Namespace: urn:schemas-microsoft-com:vml	Specifies an optional value that indicates how applications that implement VML should interpret extensions not defined as part of the original specification of core VML. [*Rationale*: This part of the original VML specification is included to assist applications that leverage existing VML support in implementing the Office Open XML Format. *end rationale*] The possible values for this attribute are defined by the ST_Ext simple type (§6.1.3.3).

Attributes	Description

fill (Shape Fill Toggle)	Specifies whether the closed path will be filled. Default is `true`. This attribute is overridden by the fill on attribute.
	
	[*Example:*
	
	`<v:shape ... fill="f" fillcolor="red" ...>`
	`</v:shape>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

fillcolor (Default Fill Color)	Specifies the default shape fill color. Default is no value. Colors are typically specified as either a named color, such as `red`, or six hexadecimal digits representing the red, green and blue values of the color, such as `#00FF30`. Full details are specified in the simple type description. The possible values for this attribute are defined by the ST_ColorType simple type (§6.2.3.5).
spidmax (Shape ID Optional Storage)	Specifies an optional value that allows applications a mechanism for storing information they need to persist related to shape IDs. Default is 0. The possible values for this attribute are defined by the XML Schema integer datatype.

stroke (Shape Stroke Toggle)	Specifies whether the path defining the shape is stroked with a solid line. The stroke element (§6.1.2.21) defines other strokes. The on attribute of the stroke element overrides this attribute. Default is `true`.
	
	[*Example:*
	
	`<v:shape ... fillcolor="red"`
	`  stroke="false" strokecolor="blue"...>`
	`</v:shape>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>strokecolor (Shape Stroke Color)</td><td>Specifies the primary color of the brush to use to stroke the path of the shape. Default is <code>black</code>. The color attribute of the stroke element (§6.1.2.21) overrides this. Colors are typically specified as either a named color, such as <code>red</code>, or six hexadecimal digits representing the red, green and blue values of the color, such as <code>#00FF30</code>. Full details are specified in the simple type description.

[Example:

<pre><v:shape ... strokecolor="red" ...>
</v:shape></pre>
 

end example]

The possible values for this attribute are defined by the ST_ColorType simple type (§6.2.3.5).</td></tr>
<tr><td>style (Shape Styling Properties)</td><td>Specifies the CSS2 styling properties of the shape. This uses the syntax described in the "Visual formatting model" of the Cascading Style Sheets, Level 2 specification, a Recommendation of the World Wide Web Consortium available here: http://www.w3.org/TR/REC-CSS2. Full descriptions of each property are not repeated here, but the VML treatment of each property is defined. Allowed properties include:

<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td><code>flip</code></td><td>Specifies that the orientation of a shape is flipped. Default is no value. Allowed values are:

• x - Flip along the y-axis, reversing the x-coordinates.
• y - Flip along the x-axis, reversing the y-coordinates.
• xy - Flip along both the y- and x-axis.
• yx - Flip along both the x- and y-axis.</td></tr>
<tr><td><code>height</code></td><td>Specifies the height of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are:

• <code>auto</code> - Default position of an element in the flow of the page.
• <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed.
• <percentage>- Value expressed as a percentage of the parent object's height.</td></tr>
</table>
</td></tr>
</table>

Attributes	Description	
	`left`	Specifies the position of the left of the containing block of the shape relative to the element left of it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	`margin-bottom`	Specifies the position of the bottom of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	`margin-left`	Specifies the position of the left of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	`margin-right`	Specifies the position of the right of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • `auto` - Default position of an element in the flow of the page.

Attributes	Description	
		• <units> - A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's width.
	margin-top	Specifies the position of the top of the containing block of the shape relative to the shape anchor. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • auto - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	mso-position-horizontal	Specifies the horizontal positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute • left • center • right • inside • outside
	mso-position-horizontal-relative	Specifies relative horizontal position data for objects in WordprocessingML documents. This modifies the mso-position-horizontal property. Default is text. Allowed values are: • margin • page • text • char
	mso-position-vertical	Specifies the vertical positioning data for objects in WordprocessingML documents. Default is absolute. Allowed values are: • absolute • top • center • bottom • inside

Attributes	Description
	<table><tr><td></td><td>• outside</td></tr><tr><td>mso-position-vertical-relative</td><td>Specifies relative vertical position data for objects in WordprocessingML documents. This modifies the mso-position-vertical property. Default is text. Allowed values are: • margin • page • text • line</td></tr><tr><td>mso-wrap-distance-bottom</td><td>Specifies the distance from the bottom of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.</td></tr><tr><td>mso-wrap-distance-left</td><td>Specifies the distance from the left side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.</td></tr><tr><td>mso-wrap-distance-right</td><td>Specifies the distance from the right side of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.</td></tr><tr><td>mso-wrap-distance-top</td><td>Specifies the distance from the top of the shape to the text that wraps around it. Default is 0 pt. Note that this property is different from the CSS margin property, which changes the origin of the shape to include the margin areas. This property does not change the origin.</td></tr><tr><td>mso-wrap-edited</td><td>Specifies whether the wrap coordinates were customized by the user. If the wrap coordinates are generated by an editor, this property is true; otherwise they were customized by a user. Default is false.</td></tr><tr><td>mso-wrap-style</td><td>Specifies the wrapping mode for text in shapes in WordprocessingML documents. Default is square. Allowed values are: • square - Wraps text inside the shape in a square. • none - Text does not wrap.</td></tr><tr><td>position</td><td>Specifies the type of positioning used to place an element. Default is static. When the element is contained inside a group, this property must be absolute. Allowed values are:</td></tr></table>

Attributes	Description	
		• `static` - The element is positioned according to the normal flow of the page. The `top` and `left` properties are ignored. If the object is anchored inline, this value is used. • `absolute` - The element is positioned relative to the parent, using the `top` and `left` properties. • `relative` - The element is positioned according to the normal flow of the page, but the `top` and `left` properties are used. The overlap of overlapping elements is governed by the `z-index` property.
	`rotation`	Specifies the angle that a shape is rotated, in degrees. Default is 0. Positive angles are clockwise.
	`top`	Specifies the position of the top of the containing block of the shape relative to the element above it in the flow of the page. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. This property shall not be used for shapes anchored inline. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed. • <percentage>- Value expressed as a percentage of the parent object's height.
	`visibility`	Specifies whether a shape is displayed. Only `inherit` and `hidden` are used; any other values are mapped to `inherit`. Default is `inherit`. Allowed values are: • `hidden` - The shape is not visible, but is still part of the flow of the objects in the browser. Mouse events are not processed. • `inherit` - The visibility state is inherited from the parent of the shape.
	`width`	Specifies the width of the containing block of the shape. Default is 0. It is specified in CSS units or, for elements in a group, in the coordinate system of the parent element. Allowed values are: • `auto` - Default position of an element in the flow of the page. • <units>- A number with an absolute units designator (cm, mm, in, pt, pc, or px) or a relative units designator (em or ex). If no units are given, pixels (px) is assumed.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>• <percentage>- Value expressed as a percentage of the parent object's width.</td></tr>
<tr><td>z-index</td><td>Specifies the display order of overlapping shapes. Default is 0. This property shall not be used for shapes anchored inline. Allowed values are:

• auto - Uses the order that the shapes appear in the page, bottom to top.
• <order>- A number that represents the stacking precedence. Shapes with higher numbers are placed on top of those with lower numbers. Negative numbers are allowed.</td></tr>
</table>
The following properties are only used by the textbox element (§6.1.2.22):
<table>
<tr><th>Property</th><th>Description</th></tr>
<tr><td>direction</td><td>Specifies the direction of the text in the textbox. Default is ltr. This property is superceded by the mso-direction-alt property if that is specified. Allowed values are:

• ltr - Test is displayed left-to-right.
• rtl - Test is displayed right-to-left.</td></tr>
<tr><td>layout-flow</td><td>Determines the flow of the text layout in a textbox. Default is horizontal. Allowed values are:

• horizontal - Text is displayed horizontally.
• vertical - Text is displayed vertically.
• vertical-ideographic - Ideographic text is displayed vertically.
• horizontal-ideographic - Ideographic text is displayed horizontally.</td></tr>
<tr><td>mso-direction-alt</td><td>Specifies an alternate direction for text in textboxes. Overrides the direction property. The only allowed value is context.</td></tr>
<tr><td>mso-fit-shape-to-text</td><td>Specifies whether the shape stretches to fit the text in the textbox. Default is false.</td></tr>
<tr><td>mso-fit-text-to-shape</td><td>Specifies whether the text stretches to fit the textbox. Default is false.</td></tr>
<tr><td>mso-layout-flow-alt</td><td>Specifies the alternate layout flow for text in textboxes. This property is used instead of layout-flow when the layout flow is from bottom to top for non-ideographic languages. Its only value</td></tr>
</table>
</td></tr>
</table>

Attributes	Description

	is bottom-to-top.
mso-next-textbox	Specifies the ID of the next textbox in a series. Used to keep track of a set of linked textboxes. Default is no value.
mso-rotate	Specifies a specific rotation value for text in a textbox. Default is 0. Allowed values are: • 0 • 90 • 180 • -90
mso-text-scale	Specifies the scaling factor for fitting text to shapes. Default is 0. This property is only used if mso-fit-text-to-shape is true.
v-text-anchor	Specifies the vertical anchoring of text in a textbox. Default is top. The alignment of a text anchor only becomes evident if mso-fit-text-to-shape is false. This property is different from the vertical-align CSS property, which is used for ideographic languages. Allowed values are: • top • middle • bottom • top-center • middle-center • bottom-center • top-baseline • bottom-baseline • top-center-baseline • bottom-center-baseline

The following properties are only used by the textpath element (§6.1.2.23):

Property	Description
font	Specifies a compound value of font settings. Default is no value. The values are the same as those of the CSS font property. The order of definitions in the string is: font-style, font-variant, font-weight, font-size, line-height, font-family.
font-family	Specifies the family of the font. Default is no value. The values are the same as those of the CSS font-family property.
font-size	Specifies the size of the font. Default is no value. The font size is defined in points. The values are the same as those of the CSS font-size property.
font-style	Specifies the amount of slant for a font. Default is normal. The values are the same as those of the CSS font-style property.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>Allowed values are:

• normal
• italic
• oblique - Treated the same as italic.</td></tr>
<tr><td>font-variant</td><td>Specifies the variant style of a font. Default is normal. The values are the same as those of the CSS font-variant property. Allowed values are:

• normal
• small-caps</td></tr>
<tr><td>font-weight</td><td>Specifies the thickness of the letters of the font. Default is normal. The values are the same as those of the CSS font-weight property. Allowed values are:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>normal
lighter
100
200
300
400</td><td>Treated as non-bold.</td></tr>
<tr><td>bold
bolder
500
600
700
800
900</td><td>Treated as bold.</td></tr>
</table>
</td></tr>
<tr><td>mso-text-shadow</td><td>Specifies whether a shadow is applied to the text on a text path. Default is false.</td></tr>
<tr><td>text-decoration</td><td>Specifies the style of text decoration. Default is none. The values are the same as those of the CSS text-decoration property. Allowed values are:

• none
• underline
• overline
• line-through
• blink</td></tr>
<tr><td>v-rotate-letters</td><td>Specifies whether the letters of the text are rotated counterclockwise by 90 degrees. Default is false.</td></tr>
<tr><td>v-same-letter-heights</td><td>Specifies whether all letters will be the same height regardless of initial case. If true, the lowercase letters are stretched to the</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>height of the uppercase letters. Default is <code>false</code>.</td></tr>
<tr><td><code>v-text-align</code></td><td>Specifies the alignment of text. Default is <code>left</code>. Allowed values are:
<code>left</code>
<code>right</code>
<code>center</code>
<code>justify</code>
<code>letter-justify</code> - Distributes the extra space between the letters.
<code>stretch-justify</code> - Stretches the letters to fill in the space.</td></tr>
<tr><td><code>v-text-kern</code></td><td>Specifies whether kerning is turned on. Default is <code>false</code>.</td></tr>
<tr><td><code>v-text-reverse</code></td><td>Specifies whether the layout order of rows is reversed. Default is <code>false</code>. This is used for vertical text layout.</td></tr>
<tr><td><code>v-text-spacing-mode</code></td><td>Specifies the mode for letter spacing. Default is <code>tightening</code>. This property determines whether space will be removed between each letter (<code>tightening</code>) or added between each letter (<code>tracking</code>). The amount of letter spacing change is defined by the <code>v-text-spacing</code> property. Allowed values are:
<code>tightening</code>
<code>tracking</code></td></tr>
<tr><td><code>v-text-spacing</code></td><td>Specifies the amount of spacing for text in 100ths of single line spacing. Default is 100.</td></tr>
</table>
The line (§6.1.2.12), polyline (§6.1.2.15) and curve (§6.1.2.3) elements ignore the following properties:
<code>top</code>
<code>left</code>
<code>width</code>
<code>height</code>
The following properties are not inherited by an element that references a shapetype element (§6.1.2.20) via the id attribute:
<code>flip</code>
<code>height</code>
<code>left</code>
<code>margin-left</code>
<code>margin-top</code>
<code>position</code>
<code>rotation</code>
<code>top</code>
</td></tr>
</table>

Attributes	Description
	• visibility • width • z-index The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShapeDefaults">
   <all minOccurs="0">
      <element ref="v:fill" minOccurs="0"/>
      <element ref="v:stroke" minOccurs="0"/>
      <element ref="v:textbox" minOccurs="0"/>
      <element ref="v:shadow" minOccurs="0"/>
      <element ref="skew" minOccurs="0"/>
      <element ref="extrusion" minOccurs="0"/>
      <element ref="callout" minOccurs="0"/>
      <element ref="lock" minOccurs="0"/>
      <element name="colormru" minOccurs="0" type="CT_ColorMru"/>
      <element name="colormenu" minOccurs="0" type="CT_ColorMenu"/>
   </all>
   <attributeGroup ref="v:AG_Ext"/>
   <attribute name="spidmax" type="xsd:integer" use="optional"/>
   <attribute name="style" type="xsd:string" use="optional"/>
   <attribute name="fill" type="ST_TrueFalse" use="optional"/>
   <attribute name="fillcolor" type="ST_ColorType" use="optional"/>
   <attribute name="stroke" type="ST_TrueFalse" use="optional"/>
   <attribute name="strokecolor" type="ST_ColorType"/>
   <attribute name="allowincell" form="qualified" type="ST_TrueFalse"/>
</complexType>
```

6.2.2.28 shapelayout (Shape Layout Properties)

This element contains child elements that store information used in the editing and layout of shapes.

Parent Elements
background (§2.2.1); hdrShapeDefaults (§2.15.1.50); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23); shapeDefaults (§2.15.1.79)

Child Elements	Subclause
idmap (Shape ID Map)	§6.2.2.13
regrouptable (Shape Grouping History)	§6.2.2.22
rules (Rule Set)	§6.2.2.26

Attributes	Description

Attributes	Description
ext (VML Extension Handling Behavior) Namespace: urn:schemas-microsoft-com:vml	Specifies an optional value that indicates how applications that implement VML should interpret extensions not defined as part of the original specification of core VML. [*Rationale*: This part of the original VML specification is included to assist applications that leverage existing VML support in implementing the Office Open XML Format. *end rationale*] The possible values for this attribute are defined by the ST_Ext simple type (§6.1.3.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ShapeLayout">
   <all>
      <element name="idmap" type="CT_IdMap" minOccurs="0"/>
      <element name="regrouptable" type="CT_RegroupTable" minOccurs="0"/>
      <element name="rules" type="CT_Rules" minOccurs="0"/>
   </all>
   <attributeGroup ref="v:AG_Ext"/>
</complexType>
```

6.2.2.29 signatureline (Digital Signature Line)

This element specifies a signature line in a document. A signature line provides a visual representation of a signature in a document that is digitally signed. The signature line element indicates that the VML shape in which it appears acts as that visual representation. Typically, the VML shape is an image.

[*Example:*

```
<v:shape ... >
  <v:imagedata ... />
  <o:signatureline v:ext="edit" id="{11979195-DE54-414B-ABD6-5F63607C648B}"
  provid="{00000000-0000-0000-0000-000000000000}" o:suggestedsigner="John Doe"
  o:suggestedsigner2="Manager" o:suggestedsigneremail=johndoe@example.com
  allowcomments="t" issignatureline="t"/>
</v:shape>
```

The signature line in the document might look like this:

end example]

Parent Elements
arc (§6.1.2.1); background (§2.2.1); curve (§6.1.2.3); group (§6.1.2.7); hdrShapeDefaults (§2.15.1.50); image (§6.1.2.10); line (§6.1.2.12); object (§2.3.3.19); oval (§6.1.2.13); pict (§2.3.3.21); pict (§2.9.23); polyline (§6.1.2.15); rect (§6.1.2.16); roundrect (§6.1.2.17); shape (§6.1.2.19); shapeDefaults (§2.15.1.79); shapetype (§6.1.2.20)

Attributes	Description

addlxml (Additional Signature Information)	Specifies an optional string that is used to store additional information about the digital signature. Default is no value. [*Rationale*: Some digital signature software stores, for example, server and region information with the signature. *end rationale*]
	
	[Example:
	
	`<o:signatureline ... o:addlxml="...">`
	`</o:signatureline>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

allowcomments (User-specified Comments Flag)	Specifies whether the user can attach comments to the signature line at signing time. Default is `false`.
	
	[Example:
	
	`<o:signatureline ... allowcomments="true">`
	`</o:signatureline>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

ext (VML Extension Handling Behavior) Namespace: urn:schemas-microsoft-com:vml	Specifies an optional value that indicates how applications that implement VML should interpret extensions not defined as part of the original specification of core VML. [*Rationale*: This part of the original VML specification is included to assist applications that leverage existing VML support in implementing the Office Open XML Format. *end rationale*] The possible values for this attribute are defined by the ST_Ext simple type (§6.1.3.3).

id (Unique ID)	Specifies a unique ID for the signature line. Default is no value.
	
	[Example:
	
	`<o:signatureline ...`
	`id="{11979195-DE54-414B-ABD6-5F63607C648B}">`

Attributes	Description
	</o:signatureline> end example] The possible values for this attribute are defined by the ST_Guid simple type (§6.2.3.12).
issignatureline (Signature Line Flag)	Specifies whether the image is a signature line. Default is true. [Example: <o:signatureline ... issignatureline="true"> </o:signatureline> end example] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
provid (Signature Provider ID)	Specifies a unique ID identifying which signature provider created the signature line. Default is no value. [*Guidance* The GUID is typically the CLSID of the provider COM add-in. *end guidance*] [Example: <o:signatureline ... provid="{00000000-0000-0000-0000-000000000000}"> </o:signatureline> end example] The possible values for this attribute are defined by the ST_Guid simple type (§6.2.3.12).
showsigndate (Show Signed Date Flag)	Specifies whether the signed signature line image generated should include the date of signing. Default is true. [Example: <o:signatureline ... showsigndate="false"> </o:signatureline> end example] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
signinginstructions (Instructions for Signing)	Specifies text shown to the user at signing time. Default is no value. [Example:

Attributes	Description
	<o:signatureline ... o:signinginstructions="Sign here"> </o:signatureline> end example] The possible values for this attribute are defined by the XML Schema string datatype.
signinginstructionsset (Use Signing Instructions Flag)	Specifies whether there is data set in the signinginstructions attribute. Default is false. [Example: <o:signatureline ... signinginstructionsset="true"> </o:signatureline> end example] The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
sigprovurl (Signature Provider Download URL)	Specifies the URL for downloading the signature provider. Default is no value. [Example: <o:signatureline ... o:sigprovurl="http://www.example.com"> </o:signatureline> end example] The possible values for this attribute are defined by the XML Schema string datatype.
suggestedsigner (Suggested Signer Line 1)	Specifies the first line of information of who should sign the signature line. Default is no value. [Example: <o:signatureline ... o:suggestedsigner="John Doe"> </o:signatureline> end example] The possible values for this attribute are defined by the XML Schema string datatype.
suggestedsigner2 (Suggested Signer Line 2)	Specifies the second line of information of who should sign the signature line. Default is no value. [Example: <o:signatureline ... o:suggestedsigner2="Title"> </o:signatureline>

Attributes	Description
	end example] The possible values for this attribute are defined by the XML Schema string datatype.
suggestedsigneremail (Suggested Signer E-mail Address)	Specifies the e-mail address of who should sign the signature line. Default is no value. [Example: `<o:signatureline ... o:suggestedsigneremail="johndoe@example.com"></o:signatureline>` end example] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SignatureLine">
   <attributeGroup ref="v:AG_Ext"/>
   <attribute name="issignatureline" type="ST_TrueFalse"/>
   <attribute name="id" type="ST_Guid"/>
   <attribute name="provid" type="ST_Guid"/>
   <attribute name="signinginstructionsset" type="ST_TrueFalse"/>
   <attribute name="allowcomments" type="ST_TrueFalse"/>
   <attribute name="showsigndate" type="ST_TrueFalse"/>
   <attribute name="suggestedsigner" type="xsd:string" form="qualified"/>
   <attribute name="suggestedsigner2" type="xsd:string" form="qualified"/>
   <attribute name="suggestedsigneremail" type="xsd:string" form="qualified"/>
   <attribute name="signinginstructions" type="xsd:string"/>
   <attribute name="addlxml" type="xsd:string"/>
   <attribute name="sigprovurl" type="xsd:string"/>
</complexType>
```

6.2.2.30 skew (Skew Transform)

This element specifies a perspective skew effect on a shape. The skew is applied to vector graphics, not image data on the shape in picture fills or image elements. The on attribute must be true and a valid value assigned to the matrix attribute.

Parent Elements
arc (§6.1.2.1); background (§2.2.1); curve (§6.1.2.3); group (§6.1.2.7); hdrShapeDefaults (§2.15.1.50); image (§6.1.2.10); line (§6.1.2.12); object (§2.3.3.19); oval (§6.1.2.13); pict (§2.3.3.21); pict (§2.9.23); polyline (§6.1.2.15); rect (§6.1.2.16); roundrect (§6.1.2.17); shape (§6.1.2.19); shapedefaults (§6.2.2.27); shapeDefaults (§2.15.1.79); shapetype (§6.1.2.20)

Attributes	Description
ext (VML Extension Handling Behavior) Namespace: urn:schemas-microsoft-com:vml	Specifies an optional value that indicates how applications that implement VML should interpret extensions not defined as part of the original specification of core VML. [*Rationale*: This part of the original VML specification is included to assist applications that leverage existing VML support in implementing the Office Open XML Format. *end rationale*] The possible values for this attribute are defined by the ST_Ext simple type (§6.1.3.3).
id (Skew ID)	Specifies a name that provides a unique identifier for a skew. Default is no value. The possible values for this attribute are defined by the XML Schema string datatype.
matrix (Skew Perspective Matrix)	Specifies a perspective transform of a skew. Default is "1,0,0,1,0,0". The matrix is given in the form "s_{xx}, s_{xy}, s_{yx}, s_{yy}, p_x, p_y" where s = scale and p = perspective. If the offset attribute is in absolute units then p_x, p_y are in 1/EMU units; otherwise they are an inverse fraction of the shape size. The possible values for this attribute are defined by the XML Schema string datatype.
offset (Skew Offset)	Specifies the amount of x,y offset from the shape's location. Default is "2pt,2pt". Positive values are measured from the upper left of the face of the shape. Values are specified as either an absolute measurement or a fractional value of the shape's dimensions (–0.5 to +0.5). The possible values for this attribute are defined by the XML Schema string datatype.
on (Skew Toggle)	Specifies whether a skew is displayed. Default is `false`. The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).
origin (Skew Origin)	Specifies the origin of the skew. Default is "0,0". Values are typically a percentage of the shape's size and range from -0.5 to +0.5. Larger values are allowed that give offsets as multiples of the shape's size. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

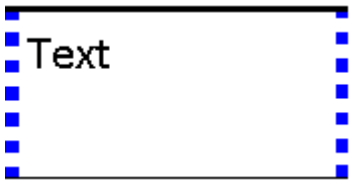
```
<complexType name="CT_Skew">
   <attributeGroup ref="v:AG_Ext"/>
   <attribute name="id" type="xsd:string" use="optional"/>
   <attribute name="on" type="ST_TrueFalse" use="optional"/>
   <attribute name="offset" type="xsd:string" use="optional"/>
   <attribute name="origin" type="xsd:string" use="optional"/>
   <attribute name="matrix" type="xsd:string" use="optional"/>
</complexType>
```

6.2.2.31 top (Text Box Top Stroke)

This element specifies the stroke properties for the top border of a text box. It entirely supercedes its parent stroke element if its on attribute is true. Thus the default value of an unspecified attribute overrides a value specified in the parent. If the on attribute is `false` or not specified, the border is not shown.

[*Example:* The text box borders are set independently. Note that the bottom border does not inherit the weight from the parent stroke element.

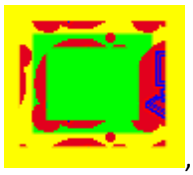
```
<v:stroke weight="2.25pt">
  <o:left v:ext="view" dashstyle="1 1" color="blue" weight="5pt" on="t"/>
  <o:top v:ext="view" color="black" weight="2.25pt" on="t"/>
  <o:right v:ext="view" dashstyle="1 1" color="blue" weight="5pt" on="t"/>
  <o:bottom v:ext="view" color="black" on="t"/>
  <o:column v:ext="view" color="#f60" on="t"/>
</v:stroke>
```



end example]

Parent Elements
background (§2.2.1); hdrShapeDefaults (§2.15.1.50); object (§2.3.3.19); pict (§2.3.3.21); pict (§2.9.23); shapeDefaults (§2.15.1.79); stroke (§6.1.2.21)

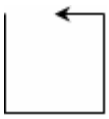
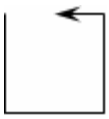
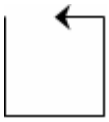
Attributes	Description
althref (Alternate Image Reference)	Specifies an alternate reference for an image in Macintosh PICT format. [*Example:* `<v:stroke ... althref="myimage.pcz" ... ></v:stroke>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
color (Stroke Color)	Specifies the stroke color. Overrides the strokecolor attribute of a shape. Default is `black`. See the fillcolor attribute for a list of supported named colors. [*Example:* The shape stroke is blue:

Attributes	Description
	<v:shape ... strokecolor="red" ... > <v:stroke color="blue"/> </v:shape> *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.2.3.5).
color2 (Stroke Alternate Pattern Color)	Specifies a second color for strokes, used when filltype is `pattern`. Default is no value. When a pattern fill is used for the stroke, the stroke color is used in colored parts of the source image. The color2 defines an alternate color to use in place of black in the source image. [*Example:* This unusual example is intended to demonstrate how the image and colors interact to create a patterned stroke. The yellow background shows transparency. The non-square shape and square image create an effective offset. The heavy stroke weight shows more of the image. The green shape fill shows how the stroke is overlaid on the shape. <v:background fillcolor="yellow"/> <v:shape style="width:60;height:50" strokecolor="red" fillcolor="lime" path="m 0,0 l 0,1000 1000,1000 1000,0 x e"> <v:stroke filltype="pattern" weight="10pt" src="myimage.gif" color2="blue"/> </v:shape>  , where myimage.gif is:  *end example*] The possible values for this attribute are defined by the ST_ColorType simple type (§6.2.3.5).
dashstyle (Stroke Dash Pattern)	Specifies the dot and dash pattern for a stroke. Default is `solid`. Pre-defined values are: • `solid` • `shortdash` • `shortdot` • `shortdashdot` • `shortdashdotdot` • `dot`

Attributes	Description

	• dash
	• longdash
	• dashdot
	• longdashdot
	• longdashdotdot
	
	A custom-defined dash pattern may also be specified using a series of numbers. These define the length of the dash (the drawn part of the stroke) and the length of the space between the dashes. The lengths are relative to the line width: a length of 1 is equal to the line width. The endcap style is applied to each dash but the arrow style is not. The string defines the length of the dash then the length of the space. This may be repeated to form complex dash styles. The string should always contain a pair of numbers; if it contains an odd number of numbers the last is disregarded. 0 implies a dot that is fourfold symmetrical (with round end caps, this is a circle).
	
	[*Example:*
	
	`<v:stroke dashstyle="0 2" weight="3pt" endcap="round"></v:stroke>`
	
	 
	
	`<v:stroke dashstyle="longdashdotdot" weight="2pt"></v:stroke>`
	
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

endarrow (Line End Arrowhead)	Specifies an arrowhead for the end of a line. Default is none. Note that the path must not be closed with the x command for the arrowhead to show. Allowed values are: • none • block • classic • diamond • oval • open [*Example:*

Attributes	Description
	<v:stroke endarrow="classic"/>  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowType simple type (§6.1.3.9).
endarrowlength (Line End Arrowhead Length)	Specifies the length of the arrowhead at the end of a line. Default is `medium`. Allowed values are: • `short` • `medium` • `long` [*Example:* <v:stroke ... endarrowlength="long" ... />  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowLength simple type (§6.1.3.8).
endarrowwidth (Line End Arrowhead Width)	Specifies the width of the arrowhead at the end of a line. Default is `medium`. Allowed values are: • `narrow` • `medium` • `wide` [*Example:* <v:stroke ... endarrowwidth="wide" ... /> 

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_StrokeArrowWidth simple type (§6.1.3.10).

endcap (Line End Cap)	Specifies the cap style for the end of a stroke. Default is `flat`. Allowed values are:
	
	• `flat`
	• `square`
	• `round`
	
	[*Example:*
	
	`<v:stroke ... endcap="round" weight="10pt" ... />`
	
	 
	`endcap="flat"`
	`endcap="square"`
	`endcap="round"`
	
	end example]
	
	The possible values for this attribute are defined by the ST_StrokeEndCap simple type (§6.1.3.11).

ext (VML Extension Handling Behavior) Namespace: urn:schemas-microsoft-com:vml	Specifies an optional value that indicates how applications that implement VML should interpret extensions not defined as part of the original specification of core VML. [*Rationale*: This part of the original VML specification is included to assist applications that leverage existing VML support in implementing the Office Open XML Format. *end rationale*] The possible values for this attribute are defined by the ST_Ext simple type (§6.1.3.3).
filltype (Stroke Image Style)	Specifies the type of fill used for the background of a stroke. Default is `solid`. Allowed values are: • `solid` - The fill pattern is solid. • `tile` - The fill image is tiled. • `pattern` - The fill image is stretched to form a pattern. • `frame` - The fill image becomes a border for the shape.

Attributes	Description

	[*Example:*
	
	`<v:shape style="width:50;height:50"`
	`  strokecolor="red"`
	`  path="m 0,0 l 0,1000 1000,1000 1000,0 x e">`
	`  <v:stroke filltype="frame" weight="10pt"`
	`  src="border.gif"/>`
	`</v:shape>`
	
	  , where border.gif is: 
	
	end example]
	
	The possible values for this attribute are defined by the ST_FillType simple type (§6.1.3.5).

forcedash (Force Dashed Outline)	Specifies whether a dashed outline is used to draw a shape when a shape has no line or fill. Default is `false`.
	
	Used by PresentationML placeholders to draw a dashed outline when there is no line and no fill for a shape.
	
	[*Example:*
	
	`<v:shape ... o:forcedash="true" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

href (Original Image Reference)	Specifies the URL to the original image file. Used only if the picture has been linked and embedded. Default is no value.
	
	[*Example:*
	
	`<v:fill ... o:href="myimage.gif" ... >`
	`</v:fill>`
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

imagealignshape	Specifies the alignment of the stroke image. If `true`, the image is aligned with the shape.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>(Stoke Image Alignment)</td><td>Otherwise, it is aligned with the containing scope. Default is <code>true</code>.

[Example: The top position offset shifts the image alignment relative to the containing window:

<pre>
<v:shape fillcolor="silver"
  style="top:20;width:50;height:50"
  path="m 0,0 l 0,1000 1000,1000 1000,0 x e">
  <v:stroke imagealignshape="false" weight="20pt"
    filltype="tile" src="myimage.gif"/>
</v:shape>
</pre>
 imagealignshape="false"

 imagealignshape="false"

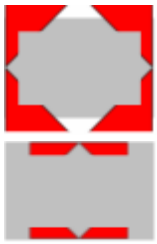
end example]

The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).</td></tr>
<tr><td>imageaspect (Stroke Image Aspect Ratio)</td><td>Specifies how the stroke image aspect ratio is preserved. Default is <code>ignore</code>. Allowed values are:

<table>
<tr><td>Value</td><td>Description</td></tr>
<tr><td><code>ignore</code></td><td>Ignore aspect issues.</td></tr>
<tr><td><code>atleast</code></td><td>Image is at least as big as imagesize.</td></tr>
<tr><td><code>atmost</code></td><td>Image is no bigger than imagesize.</td></tr>
</table>

[Example:

<pre>
<v:stroke filltype="frame" weight="10pt"
  src="border.gif" imagealignshape="true"
  imageaspect="atleast">
</v:stroke>
</pre>
 imagealignshape="ignore"</td></tr>
</table>

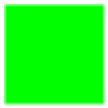
Attributes	Description
	 `imagealignshape="atleast"` `imagealignshape="atmost"` *end example*] The possible values for this attribute are defined by the ST_ImageAspect simple type (§6.1.3.6).
imagesize (Stroke Image Size)	Specifies the size of the image for the stroke. Default is the size of the image. [*Example:* `<v:stroke ... imagesize="10pt,10pt" ... />` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

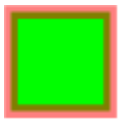
insetpen (Inset Border From Path)	Specifies that the border shall be displayed inside of the path defining the shape, rather than along the path (the default border placement), or outside of the path as might be done with an image.
	
	[*Example:*
	
	`<v:shape ... insetpen="true" ... >`
	`</v:shape>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

joinstyle (Line End Join Style))	Specifies the join style for line ends. Default is round.
	
	• round
	• bevel
	• miter
	
	[*Example:*
	
	`<v:polyline strokeweight="10pt" strokecolor="navy"`
	`points="10pt,10pt,50pt,50pt,90pt,10pt">`
	`<v:stroke joinstyle="bevel"/>`

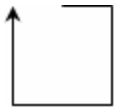
Attributes	Description
	</v:polyline>  joinstyle="round" joinstyle="bevel" joinstyle="miter" *end example*] The possible values for this attribute are defined by the ST_StrokeJoinStyle simple type (§6.1.3.12).
linestyle (Stroke Line Style)	Specifies the line style of the stroke. Default is single. • single • thinThin • thinThick • thickThin • thickBetweenThin [*Example:* <v:stroke linestyle="thickThin" weight="5pt"> </v:stroke>  *end example*] The possible values for this attribute are defined by the ST_StrokeLineStyle simple type (§6.1.3.13).
miterlimit (Miter Joint Limit)	Specifies the smoothness of the miter joint, or the maximum distance between the inner point and outer point of a joint. This number is a multiple of the thickness of the line. Default is 8. [*Example:* <v:stroke joinstyle="miter" weight="10pt"

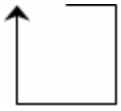
Attributes	Description
	<pre> miterlimit="2">
</v:stroke></pre>    *end example*] The possible values for this attribute are defined by the XML Schema decimal datatype.

on (Stroke Toggle)	Specifies whether the stroke is displayed. Default is `true`. This attribute overrides the shape's stroke attribute.
	
	[*Example:*
	
	<pre><v:rect style="width:50;height:50" stroked="true"
	  fillcolor="lime" strokecolor="red">
	  <v:stroke on="false" weight="5pt"/>
	</v:rect></pre>
	 
	
	end example]
	
	The possible values for this attribute are defined by the ST_TrueFalse simple type (§6.2.3.23).

opacity (Stroke Opacity)	Specifies the amount of transparency of a stroke. Default is 1.0.
	
	[*Example:*
	
	<pre><v:rect style="width:50;height:50"
	  fillcolor="lime" strokecolor="red">
	  <v:stroke weight="5pt" opacity="50%"/>
	</v:rect></pre>
	 
	
	end example]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

src (Stroke Image Location)	Specifies the source image to load for a stroke fill. Default is no value.

Attributes	Description
	[*Example:* `<v:stroke ... src="myimage.gif" ... ></v:stroke>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
startarrow (Line Start Arrowhead)	Specifies an arrowhead for the start of a line. Default is none. Note that the path must not be closed with the x command for the arrowhead to show. Allowed values are: • none • block • classic • diamond • oval • open [*Example:* `<v:stroke startarrow="classic"/>`  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowType simple type (§6.1.3.9).
startarrowlength (Line Start Arrowhead Length)	Specifies the length of the arrowhead at the start of a line. Default is medium. Allowed values are: • short • medium • long [*Example:* `<v:stroke ... startarrowlength="long" ... />`  *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_StrokeArrowLength simple type (§6.1.3.8).
startarrowwidth (Line Start Arrowhead Width)	Specifies the width of the arrowhead at the start of a line. Default is `medium`. Allowed values are: • `narrow` • `medium` • `wide` [*Example:* `<v:stroke ... startarrowwidth="wide" ... />`  *end example*] The possible values for this attribute are defined by the ST_StrokeArrowWidth simple type (§6.1.3.10).
title (Stroke Title)	Specifies the title of an embedded stroke image. This is typically set to the comment property of the image, which is often blank. [*Example:* `<v:fill ... o:title="alt text" ... ></v:fill>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.
weight (Stroke Weight)	Specifies the thickness of a stroke. Default is 1. This attribute overrides the shape's strokeweight attribute. The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_StrokeChild">
    <attributeGroup ref="v:AG_Ext"/>
    <attribute name="on" type="ST_TrueFalse" use="optional"/>
    <attribute name="weight" type="xsd:string" use="optional"/>
    <attribute name="color" type="ST_ColorType" use="optional"/>
    <attribute name="color2" type="ST_ColorType" use="optional"/>
    <attribute name="opacity" type="xsd:string" use="optional"/>
    <attribute name="linestyle" type="v:ST_StrokeLineStyle" use="optional"/>
    <attribute name="miterlimit" type="xsd:decimal" use="optional"/>
    <attribute name="joinstyle" type="v:ST_StrokeJoinStyle" use="optional"/>
    <attribute name="endcap" type="v:ST_StrokeEndCap" use="optional"/>
    <attribute name="dashstyle" type="xsd:string" use="optional"/>
    <attribute name="insetpen" type="ST_TrueFalse" use="optional"/>
    <attribute name="filltype" type="v:ST_FillType" use="optional"/>
    <attribute name="src" type="xsd:string" use="optional"/>
    <attribute name="imageaspect" type="v:ST_ImageAspect" use="optional"/>
    <attribute name="imagesize" type="xsd:string" use="optional"/>
    <attribute name="imagealignshape" type="ST_TrueFalse" use="optional"/>
    <attribute name="startarrow" type="v:ST_StrokeArrowType" use="optional"/>
    <attribute name="startarrowwidth" type="v:ST_StrokeArrowWidth" use="optional"/>
    <attribute name="startarrowlength" type="v:ST_StrokeArrowLength" use="optional"/>
    <attribute name="endarrow" type="v:ST_StrokeArrowType" use="optional"/>
    <attribute name="endarrowwidth" type="v:ST_StrokeArrowWidth" use="optional"/>
    <attribute name="endarrowlength" type="v:ST_StrokeArrowLength" use="optional"/>
    <attribute ref="href"/>
    <attribute ref="althref"/>
    <attribute ref="title"/>
    <attribute ref="forcedash"/>
</complexType>
```

6.2.3 Simple Types

This is the complete list of simple types in the urn:schemas-microsoft-com:office:office namespace.

6.2.3.1 ST_Angle (Callout Angles)

This simple type specifies values for the angle attribute of the callout element (§6.2.2.2).

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
30 (30 degrees)	30 degrees.
45 (45 degrees)	45 degrees.
60 (60 degrees)	60 degrees.
90 (90 degrees)	90 degrees.
any (Any Angle)	Unconstrained angle.

Enumeration Value	Description
auto (Automatic Angle)	The application chooses an appropriate angle.

Referenced By
callout@angle (§6.2.2.2)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Angle">
   <restriction base="xsd:string">
      <enumeration value="any"/>
      <enumeration value="30"/>
      <enumeration value="45"/>
      <enumeration value="60"/>
      <enumeration value="90"/>
      <enumeration value="auto"/>
   </restriction>
</simpleType>
```

6.2.3.2 ST_BWMode (Black And White Modes)

This simple type specifies the ways in which a shape renders in a black and white context.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
auto (Automatic)	Use the bwpure or bwnormal attributes based on the type of output being generated.
black (Black)	Use black only.
blackTextAndLines (Black Text And Lines)	Use shades of gray, except for text and lines, which are black.
color (Color)	Do not use grayscale or black and white.
grayOutline (Gray Outlines)	Use gray and white only.
grayScale (Grayscale)	Use shades of gray only.
hide (Hide Object When Displayed in Black and White)	Do not display the object when rendering in only black and white.
highContrast (Black And White)	Use black and white only, no grays.
inverseGray (Inverse Grayscale)	Use shades of gray only, but invert light and dark grays.
lightGrayscale (Light grayscale)	Use light shades of gray only.
undrawn (Do Not Show)	Do not show the object.

Enumeration Value	Description
white (White)	Use white only.

Referenced By
arc@bwmode (§6.1.2.1); arc@bwnormal (§6.1.2.1); arc@bwpure (§6.1.2.1); background@bwmode (§6.1.2.2); background@bwnormal (§6.1.2.2); background@bwpure (§6.1.2.2); curve@bwmode (§6.1.2.3); curve@bwnormal (§6.1.2.3); curve@bwpure (§6.1.2.3); image@bwmode (§6.1.2.10); image@bwnormal (§6.1.2.10); image@bwpure (§6.1.2.10); line@bwmode (§6.1.2.12); line@bwnormal (§6.1.2.12); line@bwpure (§6.1.2.12); oval@bwmode (§6.1.2.13); oval@bwnormal (§6.1.2.13); oval@bwpure (§6.1.2.13); polyline@bwmode (§6.1.2.15); polyline@bwnormal (§6.1.2.15); polyline@bwpure (§6.1.2.15); rect@bwmode (§6.1.2.16); rect@bwnormal (§6.1.2.16); rect@bwpure (§6.1.2.16); roundrect@bwmode (§6.1.2.17); roundrect@bwnormal (§6.1.2.17); roundrect@bwpure (§6.1.2.17); shape@bwmode (§6.1.2.19); shape@bwnormal (§6.1.2.19); shape@bwpure (§6.1.2.19); shapetype@bwmode (§6.1.2.20); shapetype@bwnormal (§6.1.2.20); shapetype@bwpure (§6.1.2.20)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_BWMode">
   <restriction base="xsd:string">
      <enumeration value="color"/>
      <enumeration value="auto"/>
      <enumeration value="grayScale"/>
      <enumeration value="lightGrayscale"/>
      <enumeration value="inverseGray"/>
      <enumeration value="grayOutline"/>
      <enumeration value="highContrast"/>
      <enumeration value="black"/>
      <enumeration value="white"/>
      <enumeration value="hide"/>
      <enumeration value="undrawn"/>
      <enumeration value="blackTextAndLines"/>
   </restriction>
</simpleType>
```

6.2.3.3 ST_CalloutDrop (Callout Drop Location)

This simple type specifies location values for the drop attribute of the callout element (§6.2.2.2).

This simple type's contents are a restriction of the XML Schema string datatype.

Referenced By
callout@drop (§6.2.2.2)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_CalloutDrop">
   <restriction base="xsd:string"/>
</simpleType>
```

6.2.3.4 ST_ColorMode (Extrusion Color Types)

This simple type specifies ways that the extrusion color is defined.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
auto (Use Shape Fill Color)	Specifies that the color of the extrusion is the same as the fill color of the shape.
custom (Use Custom Color)	Specifies that the extrusion is the color of the color attribute.

Referenced By
extrusion@colormode (§6.2.2.10)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ColorMode">
    <restriction base="xsd:string">
        <enumeration value="auto"/>
        <enumeration value="custom"/>
    </restriction>
</simpleType>
```

6.2.3.5 ST_ColorType (Color Type)

This simple type specifies a color. Colors are specified in one of three ways - named color, hexadecimal RGB or color palette entry. One of two optional descriptors may follow the color and a space: a number in square brackets or a recoloring instruction. [*Rationale:* An application might store the color's index in a system color palette using the numeric storage. An application might choose to make one color dependent on another using a recoloring instruction. *end rationale*] In cases where it is appropriate for an application to indicate that no color exists, the value none may be used.

A named color is specified using the name of the color. The following named colors are supported:

- Black (#000000)
- Silver (#C0C0C0)
- Gray (#808080)
- White (#FFFFFF)
- Maroon (#800000)
- Red (#FF0000)
- Purple (#800080)
- Fuchsia (#FF00FF)
- Green (#008000)
- Lime (#00FF00)

- Olive (#808000)
- Yellow (#FFFF00)
- Navy (#000080)
- Blue (#0000FF)
- Teal (#008080)
- Aqua (#00FFFF)

[*Example:*

```
<... color="red" ... >
```

end example]

Hexadecimal RGB is specified using a hash symbol (#) followed by six hexadecimal characters, where each pair represents the red, green and blue component of the color.

[*Example:*

```
<... color="#5f2726" ... >
```

end example]

Hexadecimal RGB is also optionally be specified using a hash symbol followed by three hexadecimal characters, where each character stands in for two characters in each of the red, green and blue components. For clarity, applications should use the full six character representation rather than this short form.

[*Example:* The following two representations are equivalent:

```
<... color="ff4400" ... >
<... color="f40" ... >
```

end example]

A color palette entry is specified using the name of the color in the palette.

[*Example:* This example also demonstrates the optional numeric storage:

```
<... color="buttonFace [67]" ... >
```

end example]

Recoloring instructions indicate that the given color is to be modified a particular amount. The instruction consists of a named instruction followed by the amount, a number in the range 0-255, in parenthesis. There are no required implementations for these recoloring instructions - applications are free to implement their own versions of each. A particular implementation should achieve the result implied by the instruction name. The instructions are:

- darken

- lighten
- add
- subtract
- reverseSubtract
- blackWhite
- invert
- invert128
- grayScale

[Example: The color is darker than the fill color by 50 units:

```
<... color="fill darken(50)">
```

end example]

This simple type's contents are a restriction of the XML Schema string datatype.

Referenced By
bottom@color (§6.2.2.1); bottom@color2 (§6.2.2.1); colormenu@extrusioncolor (§6.2.2.4); colormenu@fillcolor (§6.2.2.4); colormenu@shadowcolor (§6.2.2.4); colormenu@strokecolor (§6.2.2.4); column@color (§6.2.2.6); column@color2 (§6.2.2.6); extrusion@color (§6.2.2.10); left@color (§6.2.2.15); left@color2 (§6.2.2.15); right@color (§6.2.2.25); right@color2 (§6.2.2.25); shapedefaults@fillcolor (§6.2.2.27); shapedefaults@strokecolor (§6.2.2.27); top@color (§6.2.2.31); top@color2 (§6.2.2.31)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ColorType">
   <restriction base="xsd:string"/>
</simpleType>
```

6.2.3.6 ST_ConnectorType (Connector Type)

This simple type specifies types of connectors.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
curved (Curved Connector)	A curved connector.
elbow (Elbow Connector)	An elbow-shaped connector.
none (No Connector)	No connector.
straight (Straight Connector)	A straight connector.

Referenced By
arc@connectortype (§6.1.2.1); curve@connectortype (§6.1.2.3); image@connectortype (§6.1.2.10);

Referenced By
line@connectortype (§6.1.2.12); oval@connectortype (§6.1.2.13); polyline@connectortype (§6.1.2.15); rect@connectortype (§6.1.2.16); roundrect@connectortype (§6.1.2.17); shape@connectortype (§6.1.2.19); shapetype@connectortype (§6.1.2.20)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ConnectorType">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="straight"/>
      <enumeration value="elbow"/>
      <enumeration value="curved"/>
   </restriction>
</simpleType>
```

6.2.3.7 ST_ConnectType (Connection Locations Type)

This simple type specifies types of connection locations.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
custom (Custom Connections)	A custom array of connection locations.
none (No)	No connection locations.
rect (Four Connections)	Standard four connection points at midpoints of top, bottom, left, and right sides.
segments (Edit Point Connections)	The edit points of the shape are used. Edit points are the black dots in a graphical editor that are used to select parts of a shape.

Referenced By
path@connecttype (§6.1.2.14)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ConnectType">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="rect"/>
      <enumeration value="segments"/>
      <enumeration value="custom"/>
   </restriction>
</simpleType>
```

6.2.3.8 ST_ExtrusionPlane (Extrusion Planes)

This simple type specifies three axis-aligned planes.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
XY (XY Plane)	The xy plane.
YZ (YZ Plane)	The yz plane.
ZX (ZX Plane)	The zx plane.

Referenced By
extrusion@plane (§6.2.2.10)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ExtrusionPlane">
   <restriction base="xsd:string">
      <enumeration value="XY"/>
      <enumeration value="ZX"/>
      <enumeration value="YZ"/>
   </restriction>
</simpleType>
```

6.2.3.9 ST_ExtrusionRender (Extrusion Rendering Types)

This simple type specifies different rendering modes.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
boundingCube (Bounding Cube)	Rendering displays the bounding cube that contains the shape.
solid (Solid)	Rendering displays a solid shape.
wireFrame (Wireframe)	Rendering displays a wireframe shape.

Referenced By
extrusion@render (§6.2.2.10)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ExtrusionRender">
   <restriction base="xsd:string">
      <enumeration value="solid"/>
      <enumeration value="wireFrame"/>
      <enumeration value="boundingCube"/>
   </restriction>
</simpleType>
```

6.2.3.10 ST_ExtrusionType (Extrusion Type)

This simple type specifies types of extrusions.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
parallel (Parallel Projection)	Extrusion is rendered so that the center of projection is infinitely far away; the extrusion lines do not converge.
perspective (Perspective Projection)	Extrusion is rendered to a center of projection, which is the same as the vanishing point for unrotated objects.

Referenced By
extrusion@type (§6.2.2.10)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ExtrusionType">
   <restriction base="xsd:string">
      <enumeration value="perspective"/>
      <enumeration value="parallel"/>
   </restriction>
</simpleType>
```

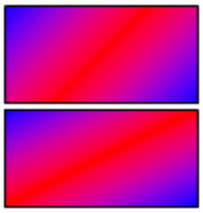
6.2.3.11 ST_FillType (Shape Fill Type)

This simple type specifies the types for fills applied to a shape.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
background (Use Background Fill)	Use the fill properties of the background of the object on which the shape exists, such as the page.

Enumeration Value	Description
frame (Stretch Image to Fit)	The image is stretched to fill the shape. 
gradient (Linear Gradient)	The fill colors blend together in a linear gradient from bottom to top. 
gradientCenter (Centered Radial Gradient)	This indicates that the gradient runs across the center of the shape for a gradient that is defined as `gradientRadial` in the parent fill element (§6.1.2.5) that is defined in the VML namespace.
gradientRadial (Radial Gradient)	The fill colors blend together in a radial gradient. 
gradientUnscaled (Unscaled Gradient)	The gradient angle is not scaled relative to the aspect ratio of the shape. For example, the shapes below are twice as wide as they are tall. The first shape uses an unscaled gradient and the second uses a regular scaled gradient: 
pattern (Image Pattern)	The image is used to create a pattern using the fill colors. 
solid (Solid Fill)	The fill pattern is a solid color. 
tile (Tiled Image)	The fill image is tiled. 

Referenced By
fill@type (§6.2.2.12)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FillType">
   <restriction base="xsd:string">
      <enumeration value="gradientCenter"/>
      <enumeration value="solid"/>
      <enumeration value="pattern"/>
      <enumeration value="tile"/>
      <enumeration value="frame"/>
      <enumeration value="gradientUnscaled"/>
      <enumeration value="gradientRadial"/>
      <enumeration value="gradient"/>
      <enumeration value="background"/>
   </restriction>
</simpleType>
```

6.2.3.12 ST_Guid (128-Bit GUID)

This simple type specifies a 128 bit GUID.

This simple type's contents are a restriction of the XML Schema token datatype.

This simple type also specifies the following restrictions:

- This simple type's contents must match the following regular expression pattern: \{[0-9A-F]{8}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{12}\}.

Referenced By
signatureline@id (§6.2.2.29); signatureline@provid (§6.2.2.29)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Guid">
   <restriction base="xsd:token">
      <pattern value="\{[0-9A-F]{8}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{12}\}"/>
   </restriction>
</simpleType>
```

6.2.3.13 ST_How (Alignment Type)

This simple type specifies types of alignment.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
bottom (Bottom Alignment)	Bottom vertical alignment.

Enumeration Value	Description
center (Center Alignment)	Center horizontal alignment.
left (Left Alignment)	Left horizontal alignment.
middle (Middle Alignment)	Middle vertical alignment.
right (Right Alignment)	Right horizontal alignment.
top (Top Alignment)	Top vertical alignment.

Referenced By
r@how (§6.2.2.21)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_How">
   <restriction base="xsd:string">
      <enumeration value="top"/>
      <enumeration value="middle"/>
      <enumeration value="bottom"/>
      <enumeration value="left"/>
      <enumeration value="center"/>
      <enumeration value="right"/>
   </restriction>
</simpleType>
```

6.2.3.14 ST_HrAlign (Alignment Type)

This simple type specifies alignments for horiztonal rules.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
center (Center Alignment)	Center aligned.
left (Left Alignment)	Left aligned.
right (Right Alignment)	Right aligned.

Referenced By
arc@hralign (§6.1.2.1); curve@hralign (§6.1.2.3); group@hralign (§6.1.2.7); image@hralign (§6.1.2.10); line@hralign (§6.1.2.12); oval@hralign (§6.1.2.13); polyline@hralign (§6.1.2.15); rect@hralign (§6.1.2.16); roundrect@hralign (§6.1.2.17); shape@hralign (§6.1.2.19); shapetype@hralign (§6.1.2.20)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_HrAlign">
   <restriction base="xsd:string">
      <enumeration value="left"/>
      <enumeration value="right"/>
      <enumeration value="center"/>
   </restriction>
</simpleType>
```

6.2.3.15 ST_InsetMode (Inset Margin Type)

This simple type specifies how inner text margins are obtained.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
auto (Automatic Margins)	Inner text margins are calculated by the application.
custom (Custom Margins)	Inner text margins are specified by the shape.

Referenced By
arc@insetmode (§6.1.2.1); curve@insetmode (§6.1.2.3); group@insetmode (§6.1.2.7); image@insetmode (§6.1.2.10); line@insetmode (§6.1.2.12); oval@insetmode (§6.1.2.13); polyline@insetmode (§6.1.2.15); rect@insetmode (§6.1.2.16); roundrect@insetmode (§6.1.2.17); shape@insetmode (§6.1.2.19); shapetype@insetmode (§6.1.2.20); textbox@insetmode (§6.1.2.22)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_InsetMode">
   <restriction base="xsd:string">
      <enumeration value="auto"/>
      <enumeration value="custom"/>
   </restriction>
</simpleType>
```

6.2.3.16 ST_OLEDrawAspect (OLE Object Representations)

This simple type specifies the ways in which OLE objects are displayed in the application.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
Content (Snapshot)	The object's presentation is a picture of the contained document (provided by the OLE server application).
Icon (Icon)	The object's presentation is an icon.

Referenced By
OLEObject@DrawAspect (§6.2.2.19)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_OLEDrawAspect">
   <restriction base="xsd:string">
      <enumeration value="Content"/>
      <enumeration value="Icon"/>
   </restriction>
</simpleType>
```

6.2.3.17 ST_OLELinkType (Embedded Object Alternate Image Request Types)

This simple type specifies the type of image that shall be requested from the application which hosts embedded object data for a linked object.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
Bitmap (Bitmap Image)	Specifies that a bitmap shall be requested.
EnhancedMetaFile (Enhanced Metafile Image)	Specifies that an enhanced metafile shall be requested.
Picture (Other Image)	Specifies that any image format may be requested.

Referenced By
LinkType (§6.2.2.16)

6.2.3.18 ST_OLEType (OLE Connection Type)

This simple type specifies whether an OLE object is included in the package (that is, embedded) or is stored outside the package (that is, linked).

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
Embed (Embedded Object)	Embedded object.
Link (Linked Object)	Linked object.

Referenced By
OLEObject@Type (§6.2.2.19)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_OLEType">
   <restriction base="xsd:string">
      <enumeration value="Embed"/>
      <enumeration value="Link"/>
   </restriction>
</simpleType>
```

6.2.3.19 ST_OLEUpdateMode (OLE Update Method Type)

This simple type specifies how an OLE object is updated.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
Always (Server Application Update)	The object is updated whenever the OLE server application indicates there is new data available.
OnCall (User Update)	The object is updated when the user chooses to update it.

Referenced By
OLEObject@UpdateMode (§6.2.2.19)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_OLEUpdateMode">
   <restriction base="xsd:string">
      <enumeration value="Always"/>
      <enumeration value="OnCall"/>
   </restriction>
</simpleType>
```

6.2.3.20 ST_RelationshipId (Explicit Relationship ID)

This simple type specifies the relationship ID in a part's relationship item which is the target of an explicit relationship from the parent XML element. This simple type is an exact analog of ST_RelationshipId defined in the http://schemas.openxmlformats.org/officeDocument/2006/relationships namespace

The type of relationship which shall be the target of the relationship specified shall be determined based on the context of the parent XML element.

[*Example*: Consider the following markup in an Office Open XML document:

```
<… o:relid="rId5" />
```

The relid attribute is of type ST_RelationshipID, and therefore the relationship with ID `rId5` shall be the target of an explicit relationship from the source part, based on the context of the parent XML element. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

Referenced By
fill@relid (§6.1.2.5); imagedata@relid (§6.1.2.11); stroke@relid (§6.1.2.21)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_RelationshipId">
   <restriction base="xsd:string"/>
</simpleType>
```

6.2.3.21 ST_RType (Rule Type)

This simple type specifies types of rules.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
align (Alignment Rule)	Alignment rule.
arc (Arc Rule)	Arc rule.
callout (Callout Rule)	Callout rule.
connector (Connector Rule)	Connector rule.

Referenced By
r@type (§6.2.2.21)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_RType">
   <restriction base="xsd:string">
      <enumeration value="arc"/>
      <enumeration value="callout"/>
      <enumeration value="connector"/>
      <enumeration value="align"/>
   </restriction>
</simpleType>
```

6.2.3.22 ST_ScreenSize (Screen Sizes Type)

This simple type specifies screen sizes.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
1024,768 (1024x768 pixels)	1024x768 pixels.
1152,862 (1152x862 pixels)	1152x862 pixels.
544,376 (544x376 pixels)	544x376 pixels.
640,480 (640x480 pixels)	640x480 pixels.
720,512 (720x512 pixels)	720x512 pixels.
800,600 (800x600 pixels)	800x600 pixels.

Referenced By
background@targetscreensize (§6.1.2.2)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ScreenSize">
   <restriction base="xsd:string">
      <enumeration value="544,376"/>
      <enumeration value="640,480"/>
      <enumeration value="720,512"/>
      <enumeration value="800,600"/>
      <enumeration value="1024,768"/>
      <enumeration value="1152,862"/>
   </restriction>
</simpleType>
```

6.2.3.23 ST_TrueFalse (Boolean Value)

This simple type specifies logical true and false.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
f (False)	Logical false.
false (False)	Logical false.
t (True)	Logical true.
true (True)	Logical true.

Referenced By

Referenced By
arc@allowincell (§6.1.2.1); arc@allowoverlap (§6.1.2.1); arc@bullet (§6.1.2.1); arc@button (§6.1.2.1); arc@clip (§6.1.2.1); arc@cliptowrap (§6.1.2.1); arc@doubleclicknotify (§6.1.2.1); arc@forcedash (§6.1.2.1); arc@hr (§6.1.2.1); arc@hrnoshade (§6.1.2.1); arc@hrstd (§6.1.2.1); arc@oleicon (§6.1.2.1); arc@oned (§6.1.2.1); arc@preferrelative (§6.1.2.1); arc@userdrawn (§6.1.2.1); arc@userhidden (§6.1.2.1); bottom@forcedash (§6.2.2.1); bottom@imagealignshape (§6.2.2.1); bottom@insetpen (§6.2.2.1); bottom@on (§6.2.2.1); callout@accentbar (§6.2.2.2); callout@dropauto (§6.2.2.2); callout@lengthspecified (§6.2.2.2); callout@minusx (§6.2.2.2); callout@minusy (§6.2.2.2); callout@on (§6.2.2.2); callout@textborder (§6.2.2.2); column@forcedash (§6.2.2.6); column@imagealignshape (§6.2.2.6); column@insetpen (§6.2.2.6); column@on (§6.2.2.6); curve@allowincell (§6.1.2.3); curve@allowoverlap (§6.1.2.3); curve@bullet (§6.1.2.3); curve@button (§6.1.2.3); curve@clip (§6.1.2.3); curve@cliptowrap (§6.1.2.3); curve@doubleclicknotify (§6.1.2.3); curve@forcedash (§6.1.2.3); curve@hr (§6.1.2.3); curve@hrnoshade (§6.1.2.3); curve@hrstd (§6.1.2.3); curve@oleicon (§6.1.2.3); curve@oned (§6.1.2.3); curve@preferrelative (§6.1.2.3); curve@userdrawn (§6.1.2.3); curve@userhidden (§6.1.2.3); diagram@autoformat (§6.2.2.8); diagram@autolayout (§6.2.2.8); diagram@reverse (§6.2.2.8); extrusion@autorotationcenter (§6.2.2.10); extrusion@lightface (§6.2.2.10); extrusion@lightharsh (§6.2.2.10); extrusion@lightharsh2 (§6.2.2.10); extrusion@lockrotationcenter (§6.2.2.10); extrusion@metal (§6.2.2.10); extrusion@on (§6.2.2.10); fill@detectmouseclick (§6.1.2.5); group@allowincell (§6.1.2.7); group@allowoverlap (§6.1.2.7); group@bullet (§6.1.2.7); group@button (§6.1.2.7); group@doubleclicknotify (§6.1.2.7); group@hr (§6.1.2.7); group@hrnoshade (§6.1.2.7); group@hrstd (§6.1.2.7); group@oned (§6.1.2.7); group@userdrawn (§6.1.2.7); group@userhidden (§6.1.2.7); image@allowincell (§6.1.2.10); image@allowoverlap (§6.1.2.10); image@bullet (§6.1.2.10); image@button (§6.1.2.10); image@clip (§6.1.2.10); image@cliptowrap (§6.1.2.10); image@doubleclicknotify (§6.1.2.10); image@forcedash (§6.1.2.10); image@hr (§6.1.2.10); image@hrnoshade (§6.1.2.10); image@hrstd (§6.1.2.10); image@oleicon (§6.1.2.10); image@oned (§6.1.2.10); image@preferrelative (§6.1.2.10); image@userdrawn (§6.1.2.10); image@userhidden (§6.1.2.10); imagedata@detectmouseclick (§6.1.2.11); ink@annotation (§6.2.2.14); left@forcedash (§6.2.2.15); left@imagealignshape (§6.2.2.15); left@insetpen (§6.2.2.15); left@on (§6.2.2.15); line@allowincell (§6.1.2.12); line@allowoverlap (§6.1.2.12); line@bullet (§6.1.2.12); line@button (§6.1.2.12); line@clip (§6.1.2.12); line@cliptowrap (§6.1.2.12); line@doubleclicknotify (§6.1.2.12); line@forcedash (§6.1.2.12); line@hr (§6.1.2.12); line@hrnoshade (§6.1.2.12); line@hrstd (§6.1.2.12); line@oleicon (§6.1.2.12); line@oned (§6.1.2.12); line@preferrelative (§6.1.2.12); line@userdrawn (§6.1.2.12); line@userhidden (§6.1.2.12); lock@adjusthandles (§6.2.2.17); lock@aspectratio (§6.2.2.17); lock@cropping (§6.2.2.17); lock@grouping (§6.2.2.17); lock@position (§6.2.2.17); lock@rotation (§6.2.2.17); lock@selection (§6.2.2.17); lock@shapetype (§6.2.2.17); lock@text (§6.2.2.17); lock@ungrouping (§6.2.2.17); lock@verticies (§6.2.2.17); oval@allowincell (§6.1.2.13); oval@allowoverlap (§6.1.2.13); oval@bullet (§6.1.2.13); oval@button (§6.1.2.13); oval@clip (§6.1.2.13); oval@cliptowrap (§6.1.2.13); oval@doubleclicknotify (§6.1.2.13); oval@forcedash (§6.1.2.13); oval@hr (§6.1.2.13); oval@hrnoshade (§6.1.2.13); oval@hrstd (§6.1.2.13); oval@oleicon (§6.1.2.13); oval@oned (§6.1.2.13); oval@preferrelative (§6.1.2.13); oval@userdrawn (§6.1.2.13); oval@userhidden (§6.1.2.13); path@extrusionok (§6.1.2.14); polyline@allowincell (§6.1.2.15); polyline@allowoverlap (§6.1.2.15); polyline@bullet (§6.1.2.15); polyline@button (§6.1.2.15); polyline@clip (§6.1.2.15); polyline@cliptowrap (§6.1.2.15); polyline@doubleclicknotify (§6.1.2.15); polyline@forcedash (§6.1.2.15); polyline@hr (§6.1.2.15); polyline@hrnoshade (§6.1.2.15); polyline@hrstd (§6.1.2.15); polyline@oleicon (§6.1.2.15); polyline@oned (§6.1.2.15); polyline@preferrelative (§6.1.2.15); polyline@userdrawn (§6.1.2.15); polyline@userhidden (§6.1.2.15); rect@allowincell (§6.1.2.16); rect@allowoverlap (§6.1.2.16); rect@bullet (§6.1.2.16); rect@button (§6.1.2.16); rect@clip (§6.1.2.16); rect@cliptowrap (§6.1.2.16); rect@doubleclicknotify (§6.1.2.16); rect@forcedash (§6.1.2.16); rect@hr (§6.1.2.16); rect@hrnoshade (§6.1.2.16); rect@hrstd

Referenced By
(§6.1.2.16); rect@oleicon (§6.1.2.16); rect@oned (§6.1.2.16); rect@preferrelative (§6.1.2.16); rect@userdrawn (§6.1.2.16); rect@userhidden (§6.1.2.16); right@forcedash (§6.2.2.25); right@imagealignshape (§6.2.2.25); right@insetpen (§6.2.2.25); right@on (§6.2.2.25); roundrect@allowincell (§6.1.2.17); roundrect@allowoverlap (§6.1.2.17); roundrect@bullet (§6.1.2.17); roundrect@button (§6.1.2.17); roundrect@clip (§6.1.2.17); roundrect@cliptowrap (§6.1.2.17); roundrect@doubleclicknotify (§6.1.2.17); roundrect@forcedash (§6.1.2.17); roundrect@hr (§6.1.2.17); roundrect@hrnoshade (§6.1.2.17); roundrect@hrstd (§6.1.2.17); roundrect@oleicon (§6.1.2.17); roundrect@oned (§6.1.2.17); roundrect@preferrelative (§6.1.2.17); roundrect@userdrawn (§6.1.2.17); roundrect@userhidden (§6.1.2.17); shape@allowincell (§6.1.2.19); shape@allowoverlap (§6.1.2.19); shape@bullet (§6.1.2.19); shape@button (§6.1.2.19); shape@clip (§6.1.2.19); shape@cliptowrap (§6.1.2.19); shape@doubleclicknotify (§6.1.2.19); shape@forcedash (§6.1.2.19); shape@hr (§6.1.2.19); shape@hrnoshade (§6.1.2.19); shape@hrstd (§6.1.2.19); shape@oleicon (§6.1.2.19); shape@oned (§6.1.2.19); shape@preferrelative (§6.1.2.19); shape@userdrawn (§6.1.2.19); shape@userhidden (§6.1.2.19); shapedefaults@allowincell (§6.2.2.27); shapedefaults@fill (§6.2.2.27); shapedefaults@stroke (§6.2.2.27); shapetype@allowincell (§6.1.2.20); shapetype@allowoverlap (§6.1.2.20); shapetype@bullet (§6.1.2.20); shapetype@button (§6.1.2.20); shapetype@clip (§6.1.2.20); shapetype@cliptowrap (§6.1.2.20); shapetype@doubleclicknotify (§6.1.2.20); shapetype@forcedash (§6.1.2.20); shapetype@hr (§6.1.2.20); shapetype@hrnoshade (§6.1.2.20); shapetype@hrstd (§6.1.2.20); shapetype@oleicon (§6.1.2.20); shapetype@oned (§6.1.2.20); shapetype@preferrelative (§6.1.2.20); shapetype@userdrawn (§6.1.2.20); shapetype@userhidden (§6.1.2.20); signatureline@allowcomments (§6.2.2.29); signatureline@issignatureline (§6.2.2.29); signatureline@showsigndate (§6.2.2.29); signatureline@signinginstructionsset (§6.2.2.29); skew@on (§6.2.2.30); stroke@forcedash (§6.1.2.21); textbox@singleclick (§6.1.2.22); top@forcedash (§6.2.2.31); top@imagealignshape (§6.2.2.31); top@insetpen (§6.2.2.31); top@on (§6.2.2.31)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TrueFalse">
   <restriction base="xsd:string">
      <enumeration value="t"/>
      <enumeration value="f"/>
      <enumeration value="true"/>
      <enumeration value="false"/>
   </restriction>
</simpleType>
```

6.2.3.24 ST_TrueFalseBlank (Boolean Value with Blank [False] State)

This simple type specifies a boolean value with a third state, using a blank attribute, which specifies that the value be `false`.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
(Blank – Logical False)	Logical false.

Enumeration Value	Description
f (Logical False)	Logical false.
false (Logical False)	Logical false.
t (Logical True)	Logical true.
true (Logical True)	Logical true.

Referenced By
arc@ole (§6.1.2.1); curve@ole (§6.1.2.3); image@ole (§6.1.2.10); line@ole (§6.1.2.12); LockedField (§6.2.2.18); oval@ole (§6.1.2.13); polyline@ole (§6.1.2.15); proxy@end (§6.2.2.20); proxy@start (§6.2.2.20); rect@ole (§6.1.2.16); roundrect@ole (§6.1.2.17); shape@ole (§6.1.2.19); shapetype@ole (§6.1.2.20)

6.3 VML - WordprocessingML Drawing

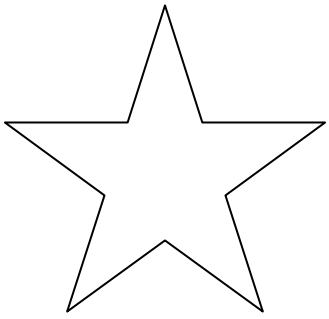
Within a WordprocessingML document, it is possible to include graphical VML objects. When these objects are present in a word processing document, it is necessary to include information about the object which is specific to their presence in a word processing document.

[*Note*: The VML format is a legacy format originally introduced with Office 2000 and is included and fully defined in this Standard for backwards compatibility reasons. The DrawingML format is a newer and richer format created with the goal of eventually replacing any uses of VML in the Office Open XML formats. VML should be considered a deprecated format included in Office Open XML for legacy reasons only and new applications that need a file format for drawings are strongly encouraged to use preferentially DrawingML .*end note*]

The VML WordprocessingML Drawing namespace acts in this capacity, specifying all information necessary to anchor and display VML objects within a word processing document.

All elements defined in this subclause shall only appear in a WordprocessingML document.

[*Example*: Consider a 5-point star added to a WordprocessingML document, for example:



This object allows surrounding text to wrap around its top and bottom, but not to either side, so this interaction with the surrounding document text (which is specific to a word processing document) is stored in the WordprocessingML Drawing namespace as follows:

```
<v:shape … >
  …
  <wd:wrap wd:type="topAndBottom" />
</v:shape>
```

The wrap element specifies how surrounding WordprocessingML document content shall wrap around the floating VML object - in this case, by wrapping to its top and bottom extents via the type attribute value of `topAndBottom`. *end example*]

6.3.1 Table of Contents

This subclause is informative.

End of informative text.

6.3.2 Elements

The following elements comprise the contents of the urn:schemas-microsoft-com:office:word namespace:

[*Note*: As the VML format is a format provided for backward compatibility, those VML elements defined in the same urn:schemas-microsoft-com:office:word namespace will remain in that namespace is it is already used by millions of documents already using VML. *end note*]

6.3.2.1 anchorlock (Anchor Location Is Locked)

This element specifies that the anchor location for this object shall not be modified at runtime when an application edits the contents of this document. [*Guidance*: An application might have automatic behaviors which reposition the anchor for a VML object based on user interaction - for example, moving it from one page to another as needed. This element shall tell applications not to perform any such behaviors. *end guidance*]

If this element is omitted, then the anchor shall not be locked for the parent VML object.

[*Example*: Consider a floating VML object which shall have its anchor locked at the current location. This setting would be specified as follows:

```
<wd:anchorLock/>
```

The anchorLock element's presence specifies that the VML object's current anchor location shall not be changed by applications editing this content. *end example*].

Parent Elements
arc (§6.1.2.1); curve (§6.1.2.3); group (§6.1.2.7); image (§6.1.2.10); line (§6.1.2.12); oval (§6.1.2.13); polyline (§6.1.2.15); rect (§6.1.2.16); roundrect (§6.1.2.17); shape (§6.1.2.19); shapetype (§6.1.2.20)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AnchorLock"/>
```

6.3.2.2 borderbottom (Bottom Border)

This element specifies the properties for the bottom border of a VML object.

Parent Elements
arc (§6.1.2.1); curve (§6.1.2.3); group (§6.1.2.7); image (§6.1.2.10); line (§6.1.2.12); oval (§6.1.2.13); polyline (§6.1.2.15); rect (§6.1.2.16); roundrect (§6.1.2.17); shape (§6.1.2.19); shapetype (§6.1.2.20)

Attributes	Description
shadow (Border shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the left and top borders, this is accomplished by moving the border down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following content: `<wd:bordertop wd:shadow="true" ... />` This element's shadow attribute is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_BorderShadow simple type (§6.3.3.1).
type (Border Style)	Specifies the style of border used on this object. See the simple type definition for a description of each border style.

Attributes	Description
	[*Example*: Consider a left border resulting in the following WordprocessingML: `<wd:borderleft wd:type="single" …/>` This border's type is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_BorderType simple type (§6.3.3.2).

width (Border Width)	Specifies the width of the current border.
	
	The width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value.
	
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<wd:bordertop wd:type="dashed" wd:width="24" …/>`
	`<wd:borderleft wd:type="dashed" wd:width="24" …/>`
	`<wd:borderbottom wd:type="dashed" wd:width="24" …/>`
	`<wd:borderright wd:type="dashed" wd:width="24" …/>`
	
	The width attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	
	The possible values for this attribute are defined by the XML Schema positiveInteger datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <attribute name="type" type="ST_BorderType" use="optional"/>
   <attribute name="width" type="xsd:positiveInteger" use="optional"/>
   <attribute name="shadow" type="ST_BorderShadow" use="optional"/>
</complexType>
```

6.3.2.3 borderleft (Left Border)

This element represents the properties for the left border of a VML object.

Parent Elements
arc (§6.1.2.1); curve (§6.1.2.3); group (§6.1.2.7); image (§6.1.2.10); line (§6.1.2.12); oval (§6.1.2.13); polyline (§6.1.2.15); rect (§6.1.2.16); roundrect (§6.1.2.17); shape (§6.1.2.19); shapetype (§6.1.2.20)

Attributes	Description

Attributes	Description
shadow (Border shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the left and top borders, this is accomplished by moving the border down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following content: `<wd:bordertop wd:shadow="true" ... />` This element's shadow attribute is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_BorderShadow simple type (§6.3.3.1).
type (Border Style)	Specifies the style of border used on this object. See the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<wd:borderleft wd:type="single" …/>` This border's type is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_BorderType simple type (§6.3.3.2).

width (Border Width)	Specifies the width of the current border.
	
	The width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value.
	
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<wd:bordertop wd:type="dashed" wd:width="24" …/>`
	`<wd:borderleft wd:type="dashed" wd:width="24" …/>`
	`<wd:borderbottom wd:type="dashed" wd:width="24" …/>`
	`<wd:borderright wd:type="dashed" wd:width="24" …/>`
	
	The width attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]

Attributes	Description
	The possible values for this attribute are defined by the XML Schema positiveInteger datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <attribute name="type" type="ST_BorderType" use="optional"/>
   <attribute name="width" type="xsd:positiveInteger" use="optional"/>
   <attribute name="shadow" type="ST_BorderShadow" use="optional"/>
</complexType>
```

6.3.2.4 borderright (Right Border)

This element specifies the properties for the right border of a VML object.

Parent Elements
arc (§6.1.2.1); curve (§6.1.2.3); group (§6.1.2.7); image (§6.1.2.10); line (§6.1.2.12); oval (§6.1.2.13); polyline (§6.1.2.15); rect (§6.1.2.16); roundrect (§6.1.2.17); shape (§6.1.2.19); shapetype (§6.1.2.20)

Attributes	Description
shadow (Border shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the left and top borders, this is accomplished by moving the border down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following content: `<wd:bordertop wd:shadow="true" ... />` This element's shadow attribute is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_BorderShadow simple type (§6.3.3.1).
type (Border Style)	Specifies the style of border used on this object. See the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML:

Attributes	Description
	<wd:borderleft wd:type="single" …/> This border's type is single, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_BorderType simple type (§6.3.3.2).
width (Border Width)	Specifies the width of the current border. The width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value. [*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup: <wd:bordertop wd:type="dashed" wd:width="24" …/> <wd:borderleft wd:type="dashed" wd:width="24" …/> <wd:borderbottom wd:type="dashed" wd:width="24" …/> <wd:borderright wd:type="dashed" wd:width="24" …/> The width attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*] The possible values for this attribute are defined by the XML Schema positiveInteger datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <attribute name="type" type="ST_BorderType" use="optional"/>
   <attribute name="width" type="xsd:positiveInteger" use="optional"/>
   <attribute name="shadow" type="ST_BorderShadow" use="optional"/>
</complexType>
```

6.3.2.5 bordertop (Top Border)

This element specifies the properties for the top border of a VML object.

Parent Elements
arc (§6.1.2.1); curve (§6.1.2.3); group (§6.1.2.7); image (§6.1.2.10); line (§6.1.2.12); oval (§6.1.2.13); polyline (§6.1.2.15); rect (§6.1.2.16); roundrect (§6.1.2.17); shape (§6.1.2.19); shapetype (§6.1.2.20)

Attributes	Description
shadow (Border shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below

Attributes	Description
	and right of the normal border location. For the left and top borders, this is accomplished by moving the border down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which shall appear with a shadow effect, resulting in the following content: `<wd:bordertop wd:shadow="true" ... />` This element's shadow attribute is `true`, indicating that the shadow effect shall be applied to the border. *end example*] The possible values for this attribute are defined by the ST_BorderShadow simple type (§6.3.3.1).
type (Border Style)	Specifies the style of border used on this object. See the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<wd:borderleft wd:type="single" …/>` This border's type is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_BorderType simple type (§6.3.3.2).

width (Border Width)	Specifies the width of the current border.
	
	The width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range may be reassigned to a more appropriate value.
	
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<wd:bordertop wd:type="dashed" wd:width="24" …/>`
	`<wd:borderleft wd:type="dashed" wd:width="24" …/>`
	`<wd:borderbottom wd:type="dashed" wd:width="24" …/>`
	`<wd:borderright wd:type="dashed" wd:width="24" …/>`
	
	The width attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	
	The possible values for this attribute are defined by the XML Schema positiveInteger datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Border">
   <attribute name="type" type="ST_BorderType" use="optional"/>
   <attribute name="width" type="xsd:positiveInteger" use="optional"/>
   <attribute name="shadow" type="ST_BorderShadow" use="optional"/>
</complexType>
```

6.3.2.6 wrap (Text Wrapping)

This element specifies the type of text wrapping which should be allowed around the contents of this VML object.

If this element is omitted, then no text wrapping shall be performed (i.e. the object shall be presented in line with text).

[*Example*: Consider the following VML object:

```
<v:shape … >
  …
  <wd:wrap wd:type="square" />
```

The wrap element specifies how surrounding WordprocessingML document content shall wrap around the floating VML object - in this case, by wrapping around its extents in a square via the type attribute value of `square`. *end example*].

Parent Elements
arc (§6.1.2.1); curve (§6.1.2.3); group (§6.1.2.7); image (§6.1.2.10); line (§6.1.2.12); oval (§6.1.2.13); polyline (§6.1.2.15); rect (§6.1.2.16); roundrect (§6.1.2.17); shape (§6.1.2.19); shapetype (§6.1.2.20)

Attributes	Description
anchorx (Horizontal Positioning Base)	Specifies the base object from which the horizontal positioning of the object should be calculated. A VML object may be horizontally positioned relative to: • The vertical edge of the page before any runs of text (the left edge for left-to-right paragraphs, the right edge for right-to-left paragraphs) • The vertical edge of the text margin before any runs of text (the left edge for left-to-right paragraphs, the right edge for right-to-left paragraphs) • The vertical edge of the text in the paragraph containing the VML object • The position of anchor for the floating VML object in the text. If this attribute is omitted, then its value shall be assumed to be page. [*Example*: Consider a VML object which should be positioned relative to the page edges, which would be specified as follows:

Attributes	Description
	<wd:wrap wd:anchorx="page" wd:anchory="page" /> The anchorx attribute specifies that horizontal anchoring is relative to the edge of the page. *end example*] The possible values for this attribute are defined by the ST_HorizontalAnchor simple type (§6.3.3.3).
anchory (Vertical Positioning Base)	Specifies the base object from which the vertical positioning of the object should be calculated. A VML object may be vertically positioned relative to: • The horizontal top edge of the page • The horizontal edge of the top text margin before any runs of text • The horizontal top edge of line containing the VML object • The horizontal top edge of the paragraph containing the text. If this attribute is omitted, then its value shall be assumed to be page. [*Example*: Consider a VML object which should be positioned relative to the page edges, which would be specified as follows: <wd:wrap wd:anchorx="page" wd:anchory="page" /> The anchory attribute specifies that horizontal anchoring is relative to the edge of the page. *end example*] The possible values for this attribute are defined by the ST_VerticalAnchor simple type (§6.3.3.4).
side (Wrapping side)	Specifies how text shall wrap around the object's left and right sides. [*Example*: Consider a floating DrawingML object which shall allow text to wrap around its left side only. This setting would be specified as follows: <wd:wrap side="left" … /> The side attribute value of `left` specifies that text shall only wrap around the left side of the object. *end example*] The possible values for this attribute are defined by the ST_WrapSide simple type (§6.3.3.5).
type (Wrapping type)	Specifies the type of wrapping - see the simple type definition for a description of each type. [*Example*: Consider the following VML object:

Attributes	Description
	<v:shape … > … <wd:wrap wd:type="topAndBottom" /> The wrap element specifies how surrounding WordprocessingML document content shall wrap around the floating VML object - in this case, by wrapping around its top and bottom extents via the type attribute value of topAndBottom. *end example*] The possible values for this attribute are defined by the ST_WrapType simple type (§6.3.3.6).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Wrap">
   <attribute name="type" type="ST_WrapType" use="optional"/>
   <attribute name="side" type="ST_WrapSide" use="optional"/>
   <attribute name="anchorx" type="ST_HorizontalAnchor" use="optional"/>
   <attribute name="anchory" type="ST_VerticalAnchor" use="optional"/>
</complexType>
```

6.3.3 Simple Types

This is the complete list of simple types in the urn:schemas-microsoft-com:office:word namespace.

6.3.3.1 ST_BorderShadow (Border Shadow Type)

This simple type specifies logical true and false values.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
f (False)	Logical false.
false (False)	Logical false.
t (True)	Logical true.
true (True)	Logical true.

Referenced By
borderbottom@shadow (§6.3.2.2); borderleft@shadow (§6.3.2.3); borderright@shadow (§6.3.2.4); bordertop@shadow (§6.3.2.5)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_BorderShadow">
   <restriction base="xsd:string">
      <enumeration value="t"/>
      <enumeration value="true"/>
      <enumeration value="f"/>
      <enumeration value="false"/>
   </restriction>
</simpleType>
```

6.3.3.2 ST_BorderType (Border Type)

This type defines which types of borders are supported.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
dash (pecifies a line border consisting of a dashed line around the parent object.)	Specifies a line border consisting of a dashed line around the parent object.
dashDotDot (Dash Dot Dot Border)	Specifies a line border consisting of a alternating dotted, dotted, dashed line around the parent object.
dashDotStroked (Stroked Dash Dot Border)	Specifies a line border consisting of a line with a series of alternating thin and thick strokes around the parent object.
dashedSmall (Small Dash Border)	Specifies a line border consisting of a dashed line with small gaps around the parent object.
dot (Dotted Border)	Specifies a line border consisting of a dotted line around the parent object.
dotDash (Dot Dash Border)	Specifies a line border consisting of a alternating dotted and dashed line around the parent object.
double (Double Line Border)	Specifies a line border consisting of a double line around the parent object.
doubleWave (Double Wavy Lines Border)	Specifies a line border consisting of a double wavy line around the parent object.
hairline (Hairline Border)	Specifies a line border consisting of a very thin line.
HTMLInset (Inset Border)	Specifies a line border consisting of an inset set of lines around the parent object. [*Example*: *end example*]

Enumeration Value	Description
HTMLOutset (Outset Border)	Specifies a line border consisting of an outset set of lines around the parent object. [*Example*: *end example*]
none (No Border)	Specifies that no border shall be applied to the current item.
single (Single Line Border)	Specifies a line border consisting of a single line around the parent object.
thick (Thick Line Border)	Specifies a line border consisting of a single line around the parent object.
thickBetweenThin (Thin-thick-thin Border)	Specifies a line border consisting of a thick line contained within a thin line with a medium sized intermediate gap around the parent object.
thickBetweenThinLarge (Large thin-thick-thin Border)	Specifies a line border consisting of a thin line contained within a thick line, contained within a thin line with a medium sized intermediate gap around the parent object.
thickBetweenThinSmall (Small thin-thick-thin Lines Border)	Specifies a line border consisting of a thin line contained within a thick line, contained within a thin line with a small intermediate gap around the parent object.
thickThin (Thick Thin Line Border)	Specifies a line border consisting of a thick line contained within a thin line with a medium sized intermediate gap around the parent object.
thickThinLarge (Thick Thin Large Gap Border)	Specifies a line border consisting of a thick line contained within a thin line with a large sized intermediate gap around the parent object.
thickThinSmall (Small thick-thin lines border)	Specifies a line border consisting of a thick line contained within a thin line with a small intermediate gap around the parent object.
thinThick (Thin Thick Line Border)	Specifies a line border consisting of a thin line contained within a thick line contained within a thick thin with a medium sized intermediate gap between each around the parent object.
thinThickLarge (Thin Thick Large Gap Border)	Specifies a line border consisting of a thin line contained within a thick line contained within a thick thin with a large sized intermediate gap between each around the parent object.

Enumeration Value	Description
thinThickSmall (Thin Thick Small Gap Border)	Specifies a line border consisting of a thin line contained within a thick line contained within a thick thin with a small intermediate gap between each around the parent object.
threeDEmboss (3D Embossed Border)	Specifies a line border consisting of three staged gradient lines around the parent object, getting darker towards the object. [*Example*:  *end example*]
threeDEngrave (3D Engraved Border)	Specifies a line border consisting of three staged gradient lines around the parent object, getting darker away from the object. [*Example*:  *end example*]
triple (Triple Line Border)	Specifies a line border consisting of a triple line around the parent object.
wave (Wavy Border)	Specifies a line border consisting of a wavy line around the parent object. [*Example*: *end example*]

Referenced By
borderbottom@type (§6.3.2.2); borderleft@type (§6.3.2.3); borderright@type (§6.3.2.4); bordertop@type (§6.3.2.5)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_BorderType">
   <restriction base="xsd:string">
      <enumeration value="none"/>
      <enumeration value="single"/>
      <enumeration value="thick"/>
      <enumeration value="double"/>
      <enumeration value="hairline"/>
      <enumeration value="dot"/>
      <enumeration value="dash"/>
      <enumeration value="dotDash"/>
      <enumeration value="dashDotDot"/>
      <enumeration value="triple"/>
      <enumeration value="thinThickSmall"/>
      <enumeration value="thickThinSmall"/>
      <enumeration value="thickBetweenThinSmall"/>
      <enumeration value="thinThick"/>
      <enumeration value="thickThin"/>
      <enumeration value="thickBetweenThin"/>
      <enumeration value="thinThickLarge"/>
      <enumeration value="thickThinLarge"/>
      <enumeration value="thickBetweenThinLarge"/>
      <enumeration value="wave"/>
      <enumeration value="doubleWave"/>
      <enumeration value="dashedSmall"/>
      <enumeration value="dashDotStroked"/>
      <enumeration value="threeDEmboss"/>
      <enumeration value="threeDEngrave"/>
      <enumeration value="HTMLOutset"/>
      <enumeration value="HTMLInset"/>
   </restriction>
</simpleType>
```

6.3.3.3 ST_HorizontalAnchor (Horizontal Anchor Type)

This simple type specifies the horizontal position to which the parent object has been anchored in the document. This anchor position shall be used as the base location to determine the final horizontal position of the object in the document.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
char (Character)	Specifies that the parent object shall be horizontally anchored based on the position of the anchor within the text flow.
margin (Margin)	Specifies that the parent object shall be horizontally anchored to the text margins.

Enumeration Value	Description
	This shall be used to specify that any horizontal positioning values shall be calculated with respect to the location of the text margin.
page (Page)	Specifies that the parent object shall be horizontally anchored to the page edge. This shall be used to specify that any horizontal positioning values shall be calculated with respect to the location of the edge of the page.
text (Text)	Specifies that the parent object shall be horizontally anchored to the text extents. This shall be used to specify that any horizontal positioning values shall be calculated with respect to the location of the edge of the text in the anchor paragraph (including text indentations on that paragraph within the text margins).

Referenced By
wrap@anchorx (§6.3.2.6)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_HorizontalAnchor">
   <restriction base="xsd:string">
      <enumeration value="margin"/>
      <enumeration value="page"/>
      <enumeration value="text"/>
      <enumeration value="char"/>
   </restriction>
</simpleType>
```

6.3.3.4 ST_VerticalAnchor (Vertical Anchor Type)

This simple type specifies the vertical position to which the parent object has been anchored in the document. This anchor position shall be used as the base location to determine the final vertical position of the object in the document.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
line (Line)	Specifies that the parent object shall be vertically anchored to the line on which its anchor appears.

Enumeration Value	Description
	This shall be used to specify that any vertical positioning values shall be calculated with respect to the location of the top edge of the anchor's line in the anchor paragraph.
margin (Margin)	Specifies that the parent object shall be vertically anchored to the text margins. This shall be used to specify that any vertical positioning values shall be calculated with respect to the location of the text margin.
page (Page)	Specifies that the parent object shall be vertically anchored to the page edge. This shall be used to specify that any vertical positioning values shall be calculated with respect to the location of the edge of the page.
text (Text)	Specifies that the parent object shall be vertically anchored to the text extents. This shall be used to specify that any vertical positioning values shall be calculated with respect to the location of the top edge of the text in the anchor paragraph.

Referenced By
wrap@anchory (§6.3.2.6)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_VerticalAnchor">
   <restriction base="xsd:string">
      <enumeration value="margin"/>
      <enumeration value="page"/>
      <enumeration value="text"/>
      <enumeration value="line"/>
   </restriction>
</simpleType>
```

6.3.3.5 ST_WrapSide (Text Wrapping Side)

This type defines which sides text can wrap around a VML object.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
both (Both sides)	Wrap text on both sides.
largest (Largest side)	Wrap text on largest side.
left (Left side)	Wrap text on left side.
right (Right side)	Wrap text on right side.

Referenced By
wrap@side (§6.3.2.6)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_WrapSide">
   <restriction base="xsd:string">
      <enumeration value="both"/>
      <enumeration value="left"/>
      <enumeration value="right"/>
      <enumeration value="largest"/>
   </restriction>
</simpleType>
```

6.3.3.6 ST_WrapType (Text Wrapping Type)

This simple type specifies the type of text wrapping which shall be allowed around a VML object within a document.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
none (No wrapping)	Specifies that text shall not be allowed to wrap around the remaining space on each lines around this VML object.
square (Square wrapping)	Specifies that text shall be allowed to wrap around the remaining space on each line around this text frame in the document using a rectangle touching each of the object's furthest edges.
through (Through wrapping)	Specifies that text shall be allowed to wrap around the remaining space on each line around this text frame in the document, including any holes in the object.
tight (Tight wrapping)	Specifies that text shall be allowed to tightly wrap around the remaining space on each line around this text frame in the document.
topAndBottom (Top and bottom wrapping)	Specifies that text shall not be allowed to wrap around the remaining space on each lines around the VML

Enumeration Value	Description
	object. Any text content shall therefore be placed on the next line following the object which does not intersect with the object's extents.

Referenced By
wrap@type (§6.3.2.6)

The following XML Schema fragment defines the contents of this simple type:

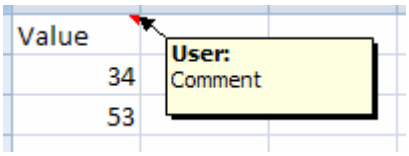
```
<simpleType name="ST_WrapType">
   <restriction base="xsd:string">
      <enumeration value="topAndBottom"/>
      <enumeration value="square"/>
      <enumeration value="none"/>
      <enumeration value="tight"/>
      <enumeration value="through"/>
   </restriction>
</simpleType>
```

6.4 VML - SpreadsheetML Drawing

It is possible to attach user interface controls, such as comments, combo boxes (dropdowns) and embedded controls, to a SpreadsheetML document. VML is used to define certain aspects of the control, such as size and visual appearance. Additional information describing the control must also be included. The VML SpreadsheetML Drawing namespace provides the additional information necessary to define the type, settings and behavior of the control.

[*Note*: The VML format is a legacy format originally introduced with Office 2000 and is included and fully defined in this Standard for backwards compatibility reasons. The DrawingML format is a newer and richer format created with the goal of eventually replacing any uses of VML in the Office Open XML formats. VML should be considered a deprecated format included in Office Open XML for legacy reasons only and new applications that need a file format for drawings are strongly encouraged to use preferentially DrawingML .*end note*]

[*Example:* Assume the comment below exists on a spreadsheet:



The following defines the additional information necessary to describe the comment. The ObjectType attribute describes the object as a comment. The Anchor element defines that its edges are anchored to the first and

fourth rows and the second and fourth columns. The Row and Column elements indicate that it points to the cell in the first row, first column.

```
<x:ClientData ObjectType="Note">
  <x:MoveWithCells/>
  <x:SizeWithCells/>
  <x:Anchor>1, 13, 0, 12, 2, 52, 2, 10</x:Anchor>
  <x:AutoFill>False</x:AutoFill>
  <x:Row>0</x:Row>
  <x:Column>0</x:Column>
  <x:Visible/>
</x:ClientData>
```

This additional comment data exists inside the VML shape that defines the comment object:

```
<v:shape id="_x0000_s1025" type="#_x0000_t202" style='position:absolute;margin
  left:57.75pt;margin-top:9pt;width:77.25pt;height:28.5pt;z-index:1;mso-wrap-
  style:tight' fillcolor="#ffffe1" o:insetmode="auto">
  <v:fill color2="#ffffe1"/>
  <v:shadow on="t" color="black" obscured="t"/>
  <v:path o:connecttype="none"/>
  <v:textbox style='mso-direction-alt:auto'>
    <div style='text-align:left'></div>
  </v:textbox>
  <x:ClientData ObjectType="Note"> … </x:ClientData>
</v:shape>
```

end example]

6.4.1 Table of Contents

This subclause is informative.

End of informative text.

6.4.2 Elements

The following elements comprise the contents of the urn:schemas-microsoft-com:office:excel namespace:

[*Note*: As the VML format is a format provided for backward compatibility, those VML elements defined in the same urn:schemas-microsoft-com:office:excel namespace will remain in that namespace is it is already used by millions of documents already using VML. *end note*]

6.4.2.1 Accel (Primary Keyboard Accelerator)

This element specifies the primary keyboard accelerator for an object. The value is the ASCII decimal number corresponding to the accelerator key. This element is used for buttons, checkboxes, radio buttons and group boxes.

[*Example:* The primary accelerator key is 'A' (65 is the ASCII decimal value for 'A'):

```
<x:ClientData … >
  <x:Accel>65</x:Accel>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema integer datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.2 Accel2 (Secondary Keyboard Accelerator)

This element specifies the secondary keyboard accelerator for an object. The value is the ASCII decimal number corresponding to the accelerator key. This element is used for buttons, checkboxes, radio buttons and group boxes.

[*Example:* The secondary accelerator key is 'A' (65 is the ASCII decimal value for 'A'):

```
<x:ClientData … >
  <x:Accel2>65</x:Accel2>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema integer datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.3 Anchor (Anchor)

This element specifies the anchor location for the object. This is a general-use element.

The value is a comma-separated list of data written out as: LeftColumn, LeftOffset, TopRow, TopOffset, RightColumn, RightOffset, BottomRow, BottomOffset.

Value	Description
LeftColumn	The left anchor column of the object (left-most column is 0). [*Example*: An object whose left anchor was off of the third column would have a LeftColumn value of 2. *end example*]
LeftOffset	The offset of the object's left edge from the left edge of the left anchor column. This value is measured in pixels.
TopRow	The top anchor row of the object (top-most column is 0). [*Example*: An object whose top anchor was off of the fifth row would have a TopRow value of 4. *end example*]
TopOffset	The offset of the object's top edge from the top edge of the top anchor row. This value is measured in pixels.
RightColumn	The right anchor column of the object (left-most column is 0). [*Example*: An object whose right anchor was off of the tenth column would have a RightColumn value of 9. *end example*]
RightOffset	The offset of the object's right edge from the left edge of the right anchor column. This value is measured in pixels.
BottomRow	The bottom anchor row of the object (top-most column is 0). [*Example*: An object whose bottom anchor was off of the tenth row would have a BottomRow value of 9. *end example*]
BottomOffset	The offset of the object's bottom edge from the bottom edge of the bottom anchor row. This value is measured in pixels.

[*Example:* The left side of the object is 15 pixels to the right of the left edge of the second column. The top edge is 2 pixels below the upper edge of the first row. The right side is 15 pixels to the right of the left edge of the fourth column. The bottom edge is 16 pixels below the top of the fourth row.

```
<x:ClientData … >
  <x:Anchor>1, 15, 0, 2, 3, 15, 3, 16</x:Anchor>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.4 AutoFill (AutoFill)

This element specifies that the object is an AutoFill object. If this element is specified without a value, it is assumed to be `true`. This is a general-use element.

[*Example:*

```
<x:ClientData … > …
  <x:AutoFill>False</x:AutoFill>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.5 AutoLine (AutoLine)

This element specifies that the object is an AutoLine object. If this element is specified without a value, it is assumed to be `true`. This is a general-use element.

[*Example:*

```
<x:ClientData … > …
  <x:AutoLine>False</x:AutoLine>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.6 AutoPict (Automatically Size)

This element specifies whether the object's size is formatted automatically by the application. If this element is specified without a value, it is assumed to be `true`. This is a general-use element for objects that use an image representation, such as OLE objects, Embedded controls, cameras and signature lines.

[*Example:*

```
<x:ClientData … > …
  <x:AutoPict>True</x:AutoPict>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.7 AutoScale (Font AutoScale)

This element specifies whether the object's font is automatically scaled by the application when the object is resized. If this element is specified without a value, it is assumed to be `true`. This element is used for attached text.

[*Example:*

```
<x:ClientData … > …
  <x:AutoScale>True</x:AutoScale>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.8 Camera (Camera Tool)

This element specifies that the object is a camera object. A camera object shows a live view of another part of the spreadsheet. If this element is specified without a value, it is assumed to be `true`. This element is used for cameras.

[*Example:*

```
<x:ClientData … > …
  <x:Camera>True</x:Camera>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.9 Cancel (Cancel Button)

This element specifies that the object is a cancel button. If this element is specified without a value, it is assumed to be `true`. This element is used for buttons.

[*Example:*

```
<x:ClientData … > …
  <x:Cancel/>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.10 CF (Clipboard Format)

This element specifies the clipboard format used to render the object. This is a general-use element for objects that use an image representation, such as OLE objects, embedded controls, cameras and signature lines.

[*Example:*

```
<x:ClientData … > …
  <x:CF>Pict</x:CF>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_CF simple type (§6.4.3.1).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.11 Checked (Checked)

This element specifies that the checkbox is checked or the radio button is selected. This element is used for checkboxes and radio buttons. Valid values are:

Value	Description
0	Unchecked / unselected
1	Checked / selected
2	Mixed selection

[*Example:*

```
<x:ClientData … > …
  <x:Checked>2</x:Checked>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema integer datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.12 ClientData (Attached Object Data)

This element specifies data associated with objects attached to a spreadsheet. While this element may contain any of the child elements below, only certain combinations are meaningful. The ObjectType attribute determines the type of object the element represents and which subset of child elements is appropriate. Relevant groups are identified for each child element.

[*Example:* The following defines additional information for a comment. Its edges are anchored to the first and fourth rows and the second and fourth columns. It points to the cell in the first row, first column.

```
<x:ClientData ObjectType="Note">
  <x:MoveWithCells/>
  <x:SizeWithCells/>
  <x:Anchor>1, 15, 0, 2, 3, 15, 3, 16</x:Anchor>
  <x:AutoFill>False</x:AutoFill>
  <x:Row>0</x:Row>
  <x:Column>0</x:Column>
  <x:Visible/>
</x:ClientData>
```

end example]

[*Example:* The following defines additional information for a radio button. It is the first in a series of radio buttons and selected by default. The accelerator key is 'A' (65 is the ASCII decimal value for 'A') and it is linked to the cell at column A, row 1 of the first sheet.

```
<x:ClientData ObjectType=3D"Radio">
  <x:SizeWithCells/>
  <x:AutoFill>False</x:AutoFill>
  <x:AutoLine>False</x:AutoLine>
  <x:TextVAlign>Center</x:TextVAlign>
  <x:Checked>1</x:Checked>
  <x:Accel>65</x:Accel>
  <x:FmlaLink>Sheet1!$A$1</x:FmlaLink>
  <x:FirstButton/>
</x:ClientData>
```

end example]

Parent Elements
arc (§6.1.2.1); curve (§6.1.2.3); group (§6.1.2.7); image (§6.1.2.10); line (§6.1.2.12); oval (§6.1.2.13); polyline (§6.1.2.15); rect (§6.1.2.16); roundrect (§6.1.2.17); shape (§6.1.2.19); shapetype (§6.1.2.20)

Child Elements	Subclause
Accel (Primary Keyboard Accelerator)	§6.4.2.1
Accel2 (Secondary Keyboard Accelerator)	§6.4.2.2
Anchor (Anchor)	§6.4.2.3
AutoFill (AutoFill)	§6.4.2.4
AutoLine (AutoLine)	§6.4.2.5
AutoPict (Automatically Size)	§6.4.2.6
AutoScale (Font AutoScale)	§6.4.2.7
Camera (Camera Tool)	§6.4.2.8
Cancel (Cancel Button)	§6.4.2.9
CF (Clipboard Format)	§6.4.2.10
Checked (Checked)	§6.4.2.11
ColHidden (Comment's Column is Hidden)	§6.4.2.13
Colored (Dropdown Color Toggle)	§6.4.2.14
Column (Comment Column Target)	§6.4.2.15
DDE (Dynamic Data Exchange)	§6.4.2.16
Default (Default Button)	§6.4.2.17
DefaultSize (Default Size Toggle)	§6.4.2.18

Child Elements	Subclause
Disabled (Macro Disable Toggle)	§6.4.2.19
Dismiss (Dismiss Button)	§6.4.2.20
DropLines (Dropdown Maximum Lines)	§6.4.2.21
DropStyle (Dropdown Style)	§6.4.2.22
Dx (Scroll Bar Width)	§6.4.2.23
FirstButton (First Radio Button)	§6.4.2.24
FmlaGroup (Linked Formula - Group Box)	§6.4.2.25
FmlaLink (Linked Formula)	§6.4.2.26
FmlaMacro (Reference to Custom Function)	§6.4.2.27
FmlaPict (Camera Source Range)	§6.4.2.28
FmlaRange (List Items Source Range)	§6.4.2.29
FmlaTxbx (Text Formula)	§6.4.2.30
Help (Help Button)	§6.4.2.31
Horiz (Scroll Bar Orientation)	§6.4.2.32
Inc (Scroll Bar Increment)	§6.4.2.33
JustLastX (Far East Alignment Toggle)	§6.4.2.34
LCT (Callback Type)	§6.4.2.35
ListItem (Non-linked List Item)	§6.4.2.36
Locked (Lock Toggle)	§6.4.2.37
LockText (Text Lock)	§6.4.2.38
MapOCX (Embedded Control)	§6.4.2.39
Max (Scroll Bar Maximum)	§6.4.2.40
Min (Scroll Bar Minimum)	§6.4.2.41
MoveWithCells (Move with Cells)	§6.4.2.42
MultiLine (Multi-line)	§6.4.2.43
MultiSel (Multiple Selections)	§6.4.2.44
NoThreeD (Disable 3D)	§6.4.2.45
NoThreeD2 (Disable 3D)	§6.4.2.46
Page (Scroll Bar Page Increment)	§6.4.2.47
PrintObject (Print Toggle)	§6.4.2.48
RecalcAlways (Recalculation Toggle)	§6.4.2.49
Row (Comment Row Target)	§6.4.2.50
RowHidden (Comment's Row is Hidden)	§6.4.2.51
ScriptExtended (HTML Script Attributes)	§6.4.2.52

Child Elements	Subclause
ScriptLanguage (HTML Script Language)	§6.4.2.53
ScriptLocation (HTML Script Location)	§6.4.2.54
ScriptText (HTML Script Text)	§6.4.2.55
SecretEdit (Password Edit)	§6.4.2.56
Sel (Selected Entry)	§6.4.2.57
SelType (Selection Type)	§6.4.2.58
SizeWithCells (Resize with Cells)	§6.4.2.59
TextHAlign (Horizontal Text Alignment)	§6.4.2.60
TextVAlign (Vertical Text Alignment)	§6.4.2.61
UIObj (UI Object Toggle)	§6.4.2.62
Val (Scroll bar position)	§6.4.2.63
ValidIds (Valid ID)	§6.4.2.64
Visible (Comment Visibility Toggle)	§6.4.2.65
VScroll (Vertical Scroll)	§6.4.2.66
VTEdit (Validation Type)	§6.4.2.67
WidthMin (Minimum Width)	§6.4.2.68

Attributes	Description
ObjectType (Object type)	Specifies the type of the object. Different sets of child elements are appropriate for different types of objects. The possible values for this attribute are defined by the ST_ObjectType simple type (§6.4.3.2).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ClientData">
   <choice minOccurs="0" maxOccurs="unbounded">
      <element name="MoveWithCells" type="ST_TrueFalseBlank"/>
      <element name="SizeWithCells" type="ST_TrueFalseBlank"/>
      <element name="Anchor" type="xsd:string"/>
      <element name="Locked" type="ST_TrueFalseBlank"/>
      <element name="DefaultSize" type="ST_TrueFalseBlank"/>
      <element name="PrintObject" type="ST_TrueFalseBlank"/>
      <element name="Disabled" type="ST_TrueFalseBlank"/>
      <element name="AutoFill" type="ST_TrueFalseBlank"/>
      <element name="AutoLine" type="ST_TrueFalseBlank"/>
      <element name="AutoPict" type="ST_TrueFalseBlank"/>
      <element name="FmlaMacro" type="xsd:string"/>
      <element name="TextHAlign" type="xsd:string"/>
      <element name="TextVAlign" type="xsd:string"/>
      <element name="LockText" type="ST_TrueFalseBlank"/>
      <element name="JustLastX" type="ST_TrueFalseBlank"/>
      <element name="SecretEdit" type="ST_TrueFalseBlank"/>
      <element name="Default" type="ST_TrueFalseBlank"/>
      <element name="Help" type="ST_TrueFalseBlank"/>
      <element name="Cancel" type="ST_TrueFalseBlank"/>
      <element name="Dismiss" type="ST_TrueFalseBlank"/>
      <element name="Accel" type="xsd:integer"/>
      <element name="Accel2" type="xsd:integer"/>
      <element name="Row" type="xsd:integer"/>
      <element name="Column" type="xsd:integer"/>
      <element name="Visible" type="ST_TrueFalseBlank"/>
      <element name="RowHidden" type="ST_TrueFalseBlank"/>
      <element name="ColHidden" type="ST_TrueFalseBlank"/>
      <element name="VTEdit" type="xsd:integer"/>
      <element name="MultiLine" type="ST_TrueFalseBlank"/>
      <element name="VScroll" type="ST_TrueFalseBlank"/>
      <element name="ValidIds" type="ST_TrueFalseBlank"/>
      <element name="FmlaRange" type="xsd:string"/>
      <element name="WidthMin" type="xsd:integer"/>
      <element name="Sel" type="xsd:integer"/>
      <element name="NoThreeD2" type="ST_TrueFalseBlank"/>
      <element name="SelType" type="xsd:string"/>
      <element name="MultiSel" type="xsd:string"/>
      <element name="LCT" type="xsd:string"/>
      <element name="ListItem" type="xsd:string"/>
      <element name="DropStyle" type="xsd:string"/>
      <element name="Colored" type="ST_TrueFalseBlank"/>
      <element name="DropLines" type="xsd:integer"/>
      <element name="Checked" type="xsd:integer"/>
      <element name="FmlaLink" type="xsd:string"/>
      <element name="FmlaPict" type="xsd:string"/>
      <element name="NoThreeD" type="ST_TrueFalseBlank"/>
      <element name="FirstButton" type="ST_TrueFalseBlank"/>
      <element name="FmlaGroup" type="xsd:string"/>
      <element name="Val" type="xsd:integer"/>
      <element name="Min" type="xsd:integer"/>
```

```
        <element name="Max" type="xsd:integer"/>
        <element name="Inc" type="xsd:integer"/>
        <element name="Page" type="xsd:integer"/>
        <element name="Horiz" type="ST_TrueFalseBlank"/>
        <element name="Dx" type="xsd:integer"/>
        <element name="MapOCX" type="ST_TrueFalseBlank"/>
        <element name="CF" type="ST_CF"/>
        <element name="Camera" type="ST_TrueFalseBlank"/>
        <element name="RecalcAlways" type="ST_TrueFalseBlank"/>
        <element name="AutoScale" type="ST_TrueFalseBlank"/>
        <element name="DDE" type="ST_TrueFalseBlank"/>
        <element name="UIObj" type="ST_TrueFalseBlank"/>
        <element name="ScriptText" type="xsd:string"/>
        <element name="ScriptExtended" type="xsd:string"/>
        <element name="ScriptLanguage" type="xsd:nonNegativeInteger"/>
        <element name="ScriptLocation" type="xsd:nonNegativeInteger"/>
        <element name="FmlaTxbx" type="xsd:string"/>
    </choice>
    <attribute name="ObjectType" type="ST_ObjectType" use="required"/>
</complexType>
```

6.4.2.13 ColHidden (Comment's Column is Hidden)

This element specifies that the column of the cell to which this comment points is hidden. If this element is specified without a value, it is assumed to be `true`. This element is used for comments.

[*Example:*

```
<x:ClientData … > …
  <x:ColHidden>True</x:ColHidden>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.14 Colored (Dropdown Color Toggle)

This element specifies that the dropdown is colored. If this element is specified without a value, it is assumed to be `true`. This element is used for dropdowns.

[*Example:*

```
<x:ClientData … > …
  <x:Colored>True</x:Colored>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.15 Column (Comment Column Target)

This element specifies the column a comment points to. The column index is 0-based. This element is used for comments.

[*Example:*

```
<x:ClientData … > …
  <x:Column>0</x:Column>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema integer datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.16 DDE (Dynamic Data Exchange)

This element specifies that the object is a DDE (Dynamic Data Exchange) link. If this element is specified without a value, it is assumed to be `true`. This is a general-use element.

[*Example:*

```
<x:ClientData … > …
  <x:DDE>True</x:DDE>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.17 Default (Default Button)

This element specifies that the object is a default (OK) button . If this element is specified without a value, it is assumed to be `true`. This element is used for buttons.

[*Example:*

```
<x:ClientData … > …
  <x:Default>True</x:Default>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.18 DefaultSize (Default Size Toggle)

This element specifies that the object is at its default size. If this element is specified without a value, it is assumed to be `true`. This is a general-use element.

[*Example:*

```
<x:ClientData … > …
  <x:DefaultSize>True</x:DefaultSize>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.19 Disabled (Macro Disable Toggle)

This element specifies that the object cannot run an attached macro. If this element is specified without a value, it is assumed to be `true`. This is a general-use element.

[*Example:*

```
<x:ClientData … > …
  <x:Disabled>True</x:Disabled>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.20 Dismiss (Dismiss Button)

This element specifies that the object is a dismiss button. If this element is specified without a value, it is assumed to be `true`. This element is used for buttons.

[*Example:*

```
<x:ClientData … > …
  <x:Dismiss>True</x:Dismiss>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.21 DropLines (Dropdown Maximum Lines)

This element specifies the maximum number of lines in the dropdown before scrollbars are added. This element is used for dropdowns.

If this element is omitted, one line is shown.

[*Example:*

```
<x:ClientData … > …
  <x:DropLines>8</x:DropLines>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema integer datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.22 DropStyle (Dropdown Style)

This element specifies the style of the dropdown. Allowed values are:

Value	Description
`Combo`	Standard combo box
`ComboEdit`	Editable combo box
`Simple`	Standard combo box with only the dropdown button visible when the box is not expanded

This element is used for dropdowns.

[*Example:*

```
<x:ClientData … > …
  <x:DropStyle>Combo</x:DropStyle>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.23 Dx (Scroll Bar Width)

This element specifies the width of the scroll bar in screen pixels. This element is used for scroll bars and spinners. [*Note*: It is possible for other controls, such as combo boxes and list boxes, to use scroll bars and this element is valid for those controls. *end note*]

[*Example:*

```
<x:ClientData … > …
  <x:Dx>16</x:Dx>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema integer datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.24 FirstButton (First Radio Button)

This element specifies that the object is the first radio button in a set of radio buttons. If this element is specified without a value, it is assumed to be `true`. This element is used for radio buttons.

[*Example:*

```
<x:ClientData … > …
  <x:FirstButton>True</x:FirstButton>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.25 FmlaGroup (Linked Formula - Group Box)

This element specifies the cell the object is linked to, using standard cell reference syntax. This element is used for group boxes. This overrides the FmlaLink for any radio buttons enclosed in the group box. The value in the linked cell and the index of the selected radio button are linked together. The formula syntax is described in §3.17 of the SpreadsheetML reference.

[*Example:*

```
<x:ClientData … > …
  <x:FmlaGroup>$A$1</x:FmlaGroup>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.26 FmlaLink (Linked Formula)

This element specifies the cell the object is linked to, using standard cell reference syntax. This element is used for checkboxes, radio buttons, scroll bars, spinners, dropdowns and list boxes. The value in the linked cell and the index of the selected item in the object are linked together. This link is ignored if the control allows multiple selections. The formula syntax is described in §3.17 of the SpreadsheetML reference.

[*Example:*

```
<x:ClientData … > …
  <x:FmlaLink>$A$4</x:FmlaLink>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.27 FmlaMacro (Reference to Custom Function)

This element specifies the custom function associated with the object. [*Example*: A macro script, add-in function, and so on. *end example*]

The format of this string shall be application-defined, and should be ignored if not understood.

[*Example*:

```
<x:ClientData … > …
  <x:FmlaMacro>Button1_Click()</x:FmlaMacro>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.28 FmlaPict (Camera Source Range)

This element specifies the range of source data cells visible in the camera. This element is used for cameras. The formula syntax is described in §3.17 of the SpreadsheetML reference.

[*Example:*

```
<x:ClientData … > …
  <x:FmlaPict>$A$2:$B$4</x:FmlaPict>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.29 FmlaRange (List Items Source Range)

This element specifies the range of source data cells used to populate the list box, using standard cell reference syntax. This element is used for list boxes. The formula syntax is described in §3.17 of the SpreadsheetML reference.

[*Example:*

```
<x:ClientData … > …
  <x:FmlaRange>$A$1:$A$15</x:FmlaRange>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.30 FmlaTxbx (Text Formula)

This element defines the formula associated with the object's text. This element is used for attached text.

[*Example:*

```
<x:ClientData … > …
  <x:FmlaTxbx>$D$9</x:FmlaTxbx>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.31 Help (Help Button)

This element specifies that the object is a help button. If this element is specified without a value, it is assumed to be `true`. This element is used for buttons.

[*Example:*

```
<x:ClientData … > …
  <x:Help>True</x:Help>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.32 Horiz (Scroll Bar Orientation)

This element specifies that the scroll bar is horizontal. If omitted, the scroll bar is vertical. If this element is specified without a value, it is assumed to be `true`. This element is used for scroll bars and spinners.

[*Example:*

```
<x:ClientData … > …
  <x:Horiz>True</x:Horiz>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.33 Inc (Scroll Bar Increment)

This element specifies the number of lines to move the scroll bar on an increment click. If omitted, the increment is 0. This element is used for scroll bars and spinners.

[*Example:*

```
<x:ClientData … > …
  <x:Inc>1</x:Inc>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema integer datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.34 JustLastX (Far East Alignment Toggle)

This element specifies that Far East alignment is set for the last line in the text. Typically, justified text in Far East environments leaves the last line unjustified. Specifying this element also justifies the last line. If this element is specified without a value, it is assumed to be `true`. This element is used for attached text.

[*Example:*

```
<x:ClientData … > …
  <x:JustLastX>True</x:JustLastX>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.35 LCT (Callback Type)

This element specifies the list box callback type. The application should use the callback to determine how to handle user actions on the list box. The only allowed value is `Normal`. This element is used for list boxes.

[*Example:*

```
<x:ClientData … > …
  <x:LCT>Normal</x:LCT>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.36 ListItem (Non-linked List Item)

This element specifies a non-linked list item that must be persisted with the list. This element is used for list boxes. [*Rationale*: This is a place for applications to store optional information associated with the list box. For example, an item to be shown in the list box that is not linked from another set of data. *end rationale*]

[*Example:*

```
<x:ClientData … > …
  <x:ListItem>TheItem</x:ListItem>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.37 Locked (Lock Toggle)

This element specifies that the object is locked when the sheet is protected. If omitted, the object is assumed to be locked. If this element is specified without a value, it is assumed to be `true`. This is a general-use element.

[*Example:*

```
<x:ClientData … > …
  <x:Locked>False</x:Locked>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.38 LockText (Text Lock)

This element specifies that the object's text is locked. If omitted, the object's text is assumed to be locked. If this element is specified without a value, it is assumed to be `true`. This element is used for attached text.

[*Example:*

```
<x:ClientData … > …
  <x:LockText>False</x:LockText>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.39 MapOCX (Embedded Control)

This element specifies that the object is an embedded control. If this element is specified without a value, it is assumed to be `true`. This element is used for all embedded controls.

[*Example:*

```
<x:ClientData … >…
  <x:MapOCX>True</x:MapOCX>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.40 Max (Scroll Bar Maximum)

This element specifies the maximum scroll bar position as the index of the list item just above the item at the top of the view when the control is scrolled all the way down. The list indexes are 1-based. If omitted, the value is assumed to be that which allows the last item to be viewed when the control is scrolled all the way down. This element is used for scroll bars and spinners.

[*Example:* Item 21 is the first item visible in the list when the object is scrolled all the way down.

```
<x:ClientData … > …
  <x:Max>20</x:Max>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema integer datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.41 Min (Scroll Bar Minimum)

This element specifies the minimum scroll bar position as the index of the list item just above the item at the top of the view when the control is scrolled all the way up, typically 0. The list indexes are 1-based. If omitted, the value is assumed to be 0. This element is used for scroll bars and spinners.

[*Example:* The first item in the list is visible when the object is scrolled all the way up:

```
<x:ClientData … > …
  <x:Min>0</x:Min>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema integer datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.42 MoveWithCells (Move with Cells)

This element specifies that the object moves with its underlying cells. If this element is specified without a value, it is assumed to be `true`. This is a general-use element.

[*Example:*

```
<x:ClientData … > …
  <x:MoveWithCells>True</x:MoveWithCells>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements

Parent Elements
ClientData (§6.4.2.12)

6.4.2.43 MultiLine (Multi-line)

This element specifies that the control is multiline. If this element is specified without a value, it is assumed to be `true`. This element is used for edit controls.

[*Example:*

```
<x:ClientData … > …
  <x:Multiline>True</x:Multiline>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.44 MultiSel (Multiple Selections)

This element specifies a comma-delimited list of selected items. This element overrides the Sel element (§6.4.2.57). This element is used for list boxes that allow multiple selections. See also the SelType element (§6.4.2.58).

[*Example:*

```
<x:ClientData … > …
  <x:MultiSel>3, 5, 6</x:MultiSel>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.45 NoThreeD (Disable 3D)

This element specifies that 3D effects are disabled. If this element is specified without a value, it is assumed to be `true`. This element is used for checkboxes, radio buttons, group boxes and scroll bars.

[*Example:*

```
<x:ClientData … > …
  <x:NoThreeD>True</x:NoThreeD>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.46 NoThreeD2 (Disable 3D)

This element specifies that 3D effects are disabled. If this element is specified without a value, it is assumed to be `true`. This element is used for dropdowns and list boxes.

[*Example:*

```
<x:ClientData … > …
  <x:NoThreeD2>True</x:NoThreeD2>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.47 Page (Scroll Bar Page Increment)

This element specifies the number of lines to move the scroll bar on a page click. This element is used for scroll bars and spinners.

[*Example:*

```
<x:ClientData … > …
  <x:Page>9</x:Page>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema integer datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.48 PrintObject (Print Toggle)

This element specifies that the object is printed when the document is printed. If omitted, it is assumed the object will print when the document is printed. If this element is specified without a value, it is assumed to be `true`. This is a general-use element.

[*Example:*

```
<x:ClientData … > …
  <x:PrintObject>False</x:PrintObject>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.49 RecalcAlways (Recalculation Toggle)

This element defines whether the object is always included in recalculation. If this element is specified without a value, it is assumed to be `true`. This is used by controls that reference cells in the spreadsheet to update themselves when the spreadsheet changes.

[*Example:*

```
<x:ClientData … > …
  <x:RecalcAlways>True</x:RecalcAlways>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.50 Row (Comment Row Target)

This element specifies the row a comment points to. The row index is 0-based. This element is used for comments.

[*Example:*

```
<x:ClientData … > …
  <x:Row>0</x:Row>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema integer datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.51 RowHidden (Comment's Row is Hidden)

This element specifies that the row of the cell to which this comment points is hidden. If this element is specified without a value, it is assumed to be `true`. This element is used for comments.

[*Example:*

```
<x:ClientData … > …
  <x:RowHidden>True</x:RowHidden>
</x:ClientData>
```

end example

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.52 ScriptExtended (HTML Script Attributes)

This element specifies custom extended attributes associated with the HTML script tag. The language and id are not included in the extended attributes. If the document contains no HTML script, this element should be ignored.

[*Example:* The extended script attribute is " src="file.js"":

```
<x:ClientData … > …
  <x:ScriptExtended>src=&amp;quot;file.js&amp;quot;</x:ScriptExtended>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.53 ScriptLanguage (HTML Script Language)

This element specifies the language of the custom function. If the document contains no HTML script, this element should be ignored. Allowed values are:

Value	Description
1	Java
2	Visual Basic
3	ASP
4	Other

[*Example:*

```
<x:ClientData … > …
  <x:ScriptLanguage>1</x:ScriptLanguage>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema nonNegativeInteger datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.54 ScriptLocation (HTML Script Location)

This element specifies the location of the custom function. If the document contains no HTML script, this element should be ignored. Allowed values are:

Value	Description
1	Head
2	Body

[*Example:*

```
<x:ClientData … > …
  <x:ScriptLocation>2</x:ScriptLocation>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema nonNegativeInteger datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.55 ScriptText (HTML Script Text)

This element specifies the script text (comment) associated with a block of HTML script in the document. If the document contains no HTML script, this element should be ignored.

[*Example:* The script text reads: "<!-- Comment -->":

```
<x:ClientData … > …
  <x:ScriptText>&lt;!&#45;- Comment &#45;-&gt;</x:ScriptText>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.56 SecretEdit (Password Edit)

This element specifies that the object represents a password edit field. If this element is specified without a value, it is assumed to be `true`. This element is used for attached text.

[*Example:*

```
<x:ClientData … > …
  <x:SecretEdit>True</x:SecretEdit>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.57 Sel (Selected Entry)

This element specifies the index of the selected item. The list indexes are 1-based. If omitted or set to a value of 0, no items are selected. This element is used for list boxes.

[*Example:*

```
<x:ClientData … >…
  <x:Sel>1</x:Sel>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema integer datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.58 SelType (Selection Type)

This element specifies the selection type for the list box. If omitted, the control is assumed to be `Single`. Allowed values are:

Value	Description
`Single`	The listbox may only have one selected item.
`Multi`	The listbox may have multiple items selected by clicking on each item.
`Extend`	The listbox may have multiple items selected by holding a control key and clicking on each item.

This element is used for list boxes.

[*Example:*

```
<x:ClientData … > …
  <x:SelType>Single</x:SelType>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.59 SizeWithCells (Resize with Cells)

This element specifies that the object resizes with its underlying cells. If this element is specified without a value, it is assumed to be `true`. This is a general-use element.

[*Example:*

```
<x:ClientData … > …
  <x:SizeWithCells>True</x:SizeWithCells>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.60 TextHAlign (Horizontal Text Alignment)

This element specifies the horizontal text alignment for the object. Valid values are Left, Justify, Center, Right and Distributed. If omitted, the alignment is assumed to be `Left`. This element is used for attached text.

[*Example:*

```
<x:ClientData … > …
  <x:TextHAlign>Right</x:TextHAlign>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.61 TextVAlign (Vertical Text Alignment)

This element specifies the horizontal text alignment for the object. Valid values are Top, Justify, Center, Bottom and Distributed. If omitted, the alignment is assumed to be `Top`. This element is used for attached text.

[*Example:*

```
<x:ClientData … > …
  <x:TextVAlign>Center</x:TextVAlign>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.62 UIObj (UI Object Toggle)

This element defines whether the object is a UI object. If this element is specified without a value, it is assumed to be `true`. This is a general-use element.

[*Example:*

```
<x:ClientData … > …
  <x:UIObj>True</x:UIObj>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.63 Val (Scroll bar position)

This element specifies the scroll bar position as the index of the list item just above the item at the top of the view, given the current scroll position. The list indexes are 1-based. If omitted, the value is assumed to be 0. This element is used for scroll bars and spinners.

[*Example:* The first list item (item 1) is just off the top of the view. The second list item is at the top of the view.

```
<x:ClientData … > …
  <x:Val>1</x:Val>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema integer datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.64 ValidIds (Valid ID)

This element specifies that the ID of a linked object is valid. This is a general-use element.

[*Example:*

```
<x:ClientData … > …
  <x:ValidIds>True</x:ValidIds>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.65 Visible (Comment Visibility Toggle)

This element specifies that a comment is visible. If omitted, the comment is assumed to be invisible. If this element is specified without a value, it is assumed to be `true`. This element is used for comments.

[*Example:*

```
<x:ClientData … > …
  <x:Visible>True</x:Visible>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.66 VScroll (Vertical Scroll)

This element specifies that the object has a vertical scroll. If omitted, a vertical scroll is not used. If this element is specified without a value, it is assumed to be `true`. This element is used for edit controls.

[*Example:*

```
<x:ClientData … > …
  <x:VScroll>True</x:VScroll>
</x:ClientData>
```

end example]

The possible values for this element are defined by the ST_TrueFalseBlank simple type (§6.4.3.3).

Parent Elements
ClientData (§6.4.2.12)

6.4.2.67 VTEdit (Validation Type)

This element specifies the type of validation to use for data input to the control. If omitted, the value is assumed to be `Text`. Valid values are:

Value	Description
0	Text
1	Integer
2	Number
3	Reference
4	Formula

This element is used for edit controls.

[*Example:*

```
<x:ClientData … > …
  <x:VTEdit>True</x:VTEdit>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema integer datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.2.68 WidthMin (Minimum Width)

This element specifies the smallest width allowed for the dropdown window in screen pixels. This element is used for list boxes and dropdowns.

[*Example:*

```
<x:ClientData … > …
  <x:WidthMin>78</x:WidthMin>
</x:ClientData>
```

end example]

The possible values for this element are defined by the XML Schema integer datatype.

Parent Elements
ClientData (§6.4.2.12)

6.4.3 Simple Types

This is the complete list of simple types in the urn:schemas-microsoft-com:office:excel namespace.

6.4.3.1 ST_CF (Clipboard Format Type)

This simple type specifies the allowed clipboard formats.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
Bitmap (Bitmap)	Bitmap.
Pict (EMF)	Enhanced metafile.
PictOld (WMF)	Windows metafile.
PictPrint (Printer Picture)	An image rendered using the default printer's settings. This is typically of higher resolution and scaled

Enumeration Value	Description
	differently compared to a picture created for on-screen rendering.
PictScreen (Screen Picture EMF)	An image rendered using screen settings. This is typically lower resolution than an image created for printing.

Referenced By
CF (§6.4.2.10)

6.4.3.2 ST_ObjectType (Object Type)

This simple type specifies the types of objects that a ClientData element can represent.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
Button (Pushbutton)	A pushbutton control.
Checkbox (Checkbox)	A checkbox control.
Dialog (Dialog)	A dialog.
Drop (Dropdown Box)	A dropdown (combo box) control.
Edit (Editable Text Field)	An editable text field control.
GBox (Group Box)	A group box control.
Group (Group)	A group of objects, such as a group of checkboxes.
Label (Label)	A label control.
LineA (Auditing Line)	A formula auditing arrow.
List (List Box)	A list control.
Movie (Movie)	A movie object in Mac format.
Note (Comment)	A comment.
Pict (Image)	A placeholder image.
Radio (Radio Button)	A radio button control.
Rect (Plain Rectangle)	A rectangle shape that is not a control.
RectA (Auditing Rectangle)	A formula auditing rectangle.
Scroll (Scroll Bar)	A scroll bar.
Shape (Plain Shape)	A general shape that is not a control.
Spin (Spin Button)	A spin button (spinner) control.

Referenced By
ClientData@ObjectType (§6.4.2.12)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ObjectType">
   <restriction base="xsd:string">
      <enumeration value="Button"/>
      <enumeration value="Checkbox"/>
      <enumeration value="Dialog"/>
      <enumeration value="Drop"/>
      <enumeration value="Edit"/>
      <enumeration value="GBox"/>
      <enumeration value="Label"/>
      <enumeration value="LineA"/>
      <enumeration value="List"/>
      <enumeration value="Movie"/>
      <enumeration value="Note"/>
      <enumeration value="Pict"/>
      <enumeration value="Radio"/>
      <enumeration value="RectA"/>
      <enumeration value="Scroll"/>
      <enumeration value="Spin"/>
      <enumeration value="Shape"/>
      <enumeration value="Group"/>
      <enumeration value="Rect"/>
   </restriction>
</simpleType>
```

6.4.3.3 ST_TrueFalseBlank (Boolean Value with Blank State)

This element specifies logical true and false. Any string that does not evaluate to true is considered false.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
(Blank - Default Value)	Use the default true or false value as defined by the element. If an element does not define a specific value for this state, it is assumed to be `true`.
f (Logical False)	Logical false.
False (Logical False)	Logical false.
t (Logical True)	Logical true.
True (Logical True)	Logical true.

Referenced By

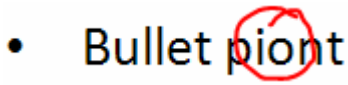
Referenced By
AutoFill (§6.4.2.4); AutoLine (§6.4.2.5); AutoPict (§6.4.2.6); AutoScale (§6.4.2.7); Camera (§6.4.2.8); Cancel (§6.4.2.9); ColHidden (§6.4.2.13); Colored (§6.4.2.14); DDE (§6.4.2.16); Default (§6.4.2.17); DefaultSize (§6.4.2.18); Disabled (§6.4.2.19); Dismiss (§6.4.2.20); FirstButton (§6.4.2.24); Help (§6.4.2.31); Horiz (§6.4.2.32); JustLastX (§6.4.2.34); Locked (§6.4.2.37); LockText (§6.4.2.38); MapOCX (§6.4.2.39); MoveWithCells (§6.4.2.42); MultiLine (§6.4.2.43); NoThreeD (§6.4.2.45); NoThreeD2 (§6.4.2.46); PrintObject (§6.4.2.48); RecalcAlways (§6.4.2.49); RowHidden (§6.4.2.51); SecretEdit (§6.4.2.56); SizeWithCells (§6.4.2.59); UIObj (§6.4.2.62); ValidIds (§6.4.2.64); Visible (§6.4.2.65); VScroll (§6.4.2.66)

6.5 VML - PresentationML Drawing

This section describes additional information attached to VML shapes that is specific to usage with PresentationML.

[*Note*: The VML format is a legacy format originally introduced with Office 2000 and is included and fully defined in this Standard for backwards compatibility reasons. The DrawingML format is a newer and richer format created with the goal of eventually replacing any uses of VML in the Office Open XML formats. VML should be considered a deprecated format included in Office Open XML for legacy reasons only and new applications that need a file format for drawings are strongly encouraged to use preferentially DrawingML .*end note*]

[*Example:* Assume the following annotation was drawn on a slide during a presentation and saved into the presentation:



The red circle annotation is stored as a VML shape that is an ink annotation. For brevity, the specific path and ink data are omitted.

```
<v:shape id="_x0000_s1029" style='position:absolute;left:126pt;
  top:327.375pt;width:27.625pt;height:24.75pt' coordorigin="4445,11549"
  coordsize="973,874" path="..." filled="f" strokecolor="red"
  strokeweight="1.5pt">
  <v:stroke endcap="round"/>
  <v:path shadowok="f" o:extrusionok="f" fillok="f" insetpenok="f"/>
  <o:lock v:ext="edit" rotation="t" aspectratio="t" verticies="t" text="t"
    shapetype="t"/>
  <o:ink i="..." annotation="t"/>
  <p:iscomment/>
 </v:shape>
```

end example]

6.5.1 Table of Contents

This subclause is informative.

End of informative text.

6.5.2 Elements

The following elements comprise the contents of the urn:schemas-microsoft-com:office:powerpoint namespace:

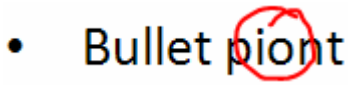
[*Note*: As the VML format is a format provided for backward compatibility, those VML elements defined in the same urn:schemas-microsoft-com:office:powerpoint namespace will remain in that namespace is it is already used by millions of documents already using VML. *end note*]

6.5.2.1 iscomment (Ink Annotation Flag)

Specifies that the object was created as an ink annotation. Default is `false`. If this element is specified without a value, it is assumed to be `true`. This element is only used with PresentationML. [*Rationale* This allows an application to treat annotation ink objects as any other annotation. For example, if annotations are hidden, the application can hide the ink object. *end rationale*]

[*Example:*

```
<v:shape ... >
  <o:ink ... annotation="true"/>
  <p:iscomment/>
</v:shape>
```



end example]

Parent Elements
shape (§6.1.2.19)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

6.5.2.2 textdata (VML Diagram Text)

This element specifies optional supplementary text information associated with a legacy VML shape that is a node in a VML diagram when it cannot otherwise be stored within the DrawingML framework.

[*Note:* An application could use this to preserve a specific diagram format for backward compatibility, but it is strongly recommended to upgrade all VML shapes to DrawingML shapes. *end note*]

Parent Elements
arc (§6.1.2.1); curve (§6.1.2.3); group (§6.1.2.7); image (§6.1.2.10); line (§6.1.2.12); oval (§6.1.2.13); polyline (§6.1.2.15); rect (§6.1.2.16); roundrect (§6.1.2.17); shape (§6.1.2.19); shapetype (§6.1.2.20)

Attributes	Description
id (Text Reference)	Specifies the identifier that is used in conjunction with a corresponding relationship file to resolve the location of the diagram shape text. [*Example:* `<v:shape ... o:dgmnodekind="0" > <v:textbox inset="0,0,0,0"/> <p:textdata id="rId1"/> </v:shape>` *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Rel">
   <attribute name="id" type="xsd:string"/>
</complexType>
```

7. Shared MLs Reference Material

7.1 Math

The following documentation describes the XML representation of equations. The outermost element is oMathPara, a paragraph of one or more equations. Each equation inside the math paragraph is represented as a single oMath. Inside each oMath is a combination of runs (r) and objects or functions (such as accents, acc, or fractions, (f).

7.1.1 Table of Contents

This subclause is informative.

End of informative text.

7.1.2 Elements

The following elements describe contents of equations.

7.1.2.1 acc (Accent)

This element specifies the accent function, consisting of a base and a combining diacritical mark. [*Example*: Example accent functions are $\dot{a}$, $\acute{a}$, and $\tilde{a}$.

```
<m:acc>
  <m:accPr>
    <m:chr m:val="&#771;"/>
    <m:ctrlPr/>
  </m:accPr>
  <m:e>
    <m:r>
      <m:t>a</m:t>
    </m:r>
  </m:e>
</m:acc>
```

end example]

Parent Elements

Parent Elements
deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); e (§7.1.2.32); fName (§7.1.2.37); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); sub (§7.1.2.112); sup (§7.1.2.114)

Child Elements	Subclause
accPr (Accent Properties)	§7.1.2.2
e (Base (Argument))	§7.1.2.32

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Acc">
   <sequence>
      <element name="accPr" type="CT_AccPr" minOccurs="0"/>
      <element name="e" type="CT_OMathArg"/>
   </sequence>
</complexType>
```

7.1.2.2 accPr (Accent Properties)

This element specifies the properties of the Accent function. [*Example*:

```
<m:accPr>
  <m:chr m:val="&#771;"/>
  <m:ctrlPr/>
</m:accPr>
```

end example]

Parent Elements
acc (§7.1.2.1)

Child Elements	Subclause
chr (Accent Character)	§7.1.2.20
ctrlPr (Control Properties)	§7.1.2.23

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AccPr">
   <sequence>
      <element name="chr" type="CT_Char" minOccurs="0"/>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.3 aln (Alignment)

This element specifies the alignment property on the box function. It is utilized only when the box is designated as an operator emulator. When true, this operator emulator serves as an alignment point; that is, designated alignment points in other equations can be aligned with it. [*Example*: For example, the following equation uses the operator emulator as an alignment point. $a == b$.

Its XML representation is as follows:

```
<m:oMath>
  <m:r>
    <m:t>a</m:t>
  </m:r>
  <m:box>
    <m:boxPr>
      <m:opEmu m:val="on"/>
      <m:aln m:val="on"/>
      <m:ctrlPr/>
    </m:boxPr>
    <m:e>
      <m:r>
        <m:t>==</m:t>
      </m:r>
    </m:e>
  </m:box>
  <m:r>
    <m:t>b</m:t>
  </m:r>
</m:oMath>
```

end example]

Parent Elements
boxPr (§7.1.2.14); rPr (§7.1.2.91)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present.

Attributes	Description
	The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.4 alnScr (Align Scripts)

This element specifies the alignment of scripts in the subscript/superscript function. When on, subscripts and superscripts are aligned to each other. When off, they are kerned to the shape of the base. If this element is omitted, scripts are not aligned. [*Example*: Example (OFF): f_2^3; Example (ON): f_2^3.

The XML representation of the second example above is:

```
<m:sSubSup>
  <m:sSubSupPr>
    <m:alnScr   m:val="on"/>
  </m:sSubSupPr>
  <m:e>
    <m:r>
      <m:t>f</m:t>
    </m:r>
  </m:e>
  <m:sub>
    <m:r>
      <m:rPr>
        <m:scr m:val="roman"/>
        <m:sty m:val="p"/>
      </m:rPr>
      <m:t>2</m:t>
    </m:r>
  </m:sub>
  <m:sup>
    <m:r>
      <m:rPr>
        <m:scr m:val="roman"/>
        <m:sty m:val="p"/>
      </m:rPr>
      <m:t>3</m:t>
    </m:r>
  </m:sup>
</m:sSubSup>
```

end example]

Parent Elements
sSubSupPr (§7.1.2.104)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.5 argPr (Argument Properties)

This element specifies any properties of the math argument. [*Example*: The XML below represents the argSz attribute on the base element e of a box:

```
<m:box>
  <m:boxPr>
    <m:noBreak m:val="off"/>
    <m:ctrlPr/>
  </m:boxPr>
  <m:e>
    <m:argPr>
      <m:argSz m:val="-1"/>
    </m:argPr>
    <m:r>
      <m:t>abc</m:t>
    </m:r>
  </m:e>
</m:box>
```

end example]

Parent Elements

Parent Elements
deg (§7.1.2.26); den (§7.1.2.28); e (§7.1.2.32); fName (§7.1.2.37); lim (§7.1.2.52); num (§7.1.2.75); sub (§7.1.2.112); sup (§7.1.2.114)

Child Elements	Subclause
argSz (Argument Size)	§7.1.2.6

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OMathArgPr">
   <sequence>
      <element name="argSz" type="CT_Integer2" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.6 argSz (Argument Size)

This element specifies the size, or script level, of an argument. If the element is omitted, the default argument size is 0. [*Example*: The following example contains three runs: regular text in the equation, a box object with the base at script size (val=-1) and a box object with the base at script-script size (val=-2).

$$abc^{abc}\ _{abc}$$

The XML below shows argSize used in the middle box.

```
<m:box>
  <m:boxPr>
    <m:noBreak m:val="off"/>
  </m:boxPr>
  <m:e>
    <m:argPr>
      <m:argSz m:val="-1"/>
    </m:argPr>
    <m:r>
      <m:t>abc</m:t>
    </m:t>
  </m:e>
</m:box>
```

end example]

Parent Elements
argPr (§7.1.2.5)

Attributes	Description

Attributes	Description
val (Value)	Specifies a value between -2 and 2 for the property defined by the parent XML element. The positive or negative sign specifies in which direction to change argument size; the absolute value specifies by how much. The table below represents two cases in which argument size can be changed: superscripts and boxes. In the superscript object a^{b^c}, by default the term c has script-script size. Should the user wish for the c to be shown at script size, `val` should be set to +1 (that is, one size larger). Should the user wish for c to be shown at text size, `val` should be set to +2 (that is, two sizes larger). **`val` of c in a^{b^c}** / **Display** Default / a^{b^c} +1 / a^{b^c} +2 / $a^{b^{\mathbf{c}}}$ **`val` of abc** / **Display** Default / abc -1 / abc -2 / abc The possible values for this attribute are defined by the ST_Integer2 simple type (§7.1.3.5).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Integer2">
   <attribute name="val" type="ST_Integer2" use="required"/>
</complexType>
```

7.1.2.7 bar (Bar)

This element specifies the bar function, consisting of a base argument and an overbar or underbar, as in $\overline{a}$and $\underline{a}$.

[*Example*: The XML below demonstrates the overbar in use.

```
<m:bar>
  <m:barPr>
   <m:pos   m:val="top"/>
  </m:barPr>
```

```
    <m:e>
      <m:r>
        <m:t>a</m:t>
      </m:r>
    </m:e>
  </m:bar>
```

end example]

Parent Elements
deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); e (§7.1.2.32); fName (§7.1.2.37); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); sub (§7.1.2.112); sup (§7.1.2.114)

Child Elements	Subclause
barPr (Bar Properties)	§7.1.2.8
e (Base (Argument))	§7.1.2.32

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Bar">
   <sequence>
      <element name="barPr" type="CT_BarPr" minOccurs="0"/>
      <element name="e" type="CT_OMathArg"/>
   </sequence>
</complexType>
```

7.1.2.8 barPr (Bar Properties)

This element specifies properties of the bar function. If this element is omitted, the bar assumes its default location of top (the mathematical overbar).

Parent Elements
bar (§7.1.2.7)

Child Elements	Subclause
ctrlPr (Control Properties)	§7.1.2.23
pos (Position (Bar))	§7.1.2.84

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BarPr">
   <sequence>
      <element name="pos" type="CT_TopBot" minOccurs="0"/>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.9 baseJc (Matrix Base Justification)

This element specifies the justification of the matrix. Text outside of the matrix can be aligned with the bottom, top, or center of a matrix function. If this element is omitted, the matrix assumes center justification.

[*Example*: This matrix has center baseJc: $\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$

This matrix has top baseJc: $\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$

This matrix has bottom baseJc: $\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$

The XML below represents the matrix with top baseJC:

```
<m:d>
  <m:dPr>
    <m:shp  m:val="match"/>
  </m:dPr>
  <m:e>
    <m:m>
      <m:mPr>
        <m:baseJc m:val="top"/>
        <m:mcs>
          <m:mc>
            <m:mcPr>
              <m:mcJc  m:val="center"/>
              <m:count m:val="2"/>
            </m:mcPr>
          </m:mc>
        </m:mcs>
      </m:mPr>
```

```
<m:mr>
  <m:e>
    <m:r>
      <m:rPr>
        <m:scr  m:val="roman"/>
        <m:sty m:val="p"/>
      </m:rPr>
      <m:t>1</m:t>
    </m:r>
  </m:e>
  <m:e>
    <m:r>
      <m:rPr>
        <m:scr  m:val="roman"/>
        <m:sty  m:val="p"/>
      </m:rPr>
      <m:t>2</m:t>
    </m:r>
  </m:e>
</m:mr>
<m:mr>
  <m:e>
    <m:r>
      <m:rPr>
        <m:scr  m:val="roman"/>
        <m:sty  m:val="p"/>
      </m:rPr>
      <m:t>3</m:t>
    </m:r>
  </m:e>
  <m:e>
    <m:r>
      <m:rPr>
        <m:scr  m:val="roman"/>
        <m:sty  m:val="p"/>
      </m:rPr>
      <m:t>4</m:t>
    </m:r>
  </m:e>
```

```
          <m:e>
            <m:r>
              <m:rPr>
                <m:scr  m:val="roman"/>
                <m:sty  m:val="p"/>
              </m:rPr>
              <m:t>5</m:t>
            </m:r>
          </m:e>
          <m:e>
            <m:r>
              <m:rPr>
                <m:scr  m:val="roman"/>
                <m:sty  m:val="p"/>
              </m:rPr>
              <m:t>6</m:t>
            </m:r>
          </m:e>
        </m:mr>
      </m:m>
    </m:e>
  </m:d>
```

end example]

Parent Elements
eqArrPr (§7.1.2.35); mPr (§7.1.2.68)

Attributes	Description
val (Value)	Specifies the vertical justification parent element respect to surrounding text. Possible values are `top`, `bot`, and `center`. [*Example*: The following examples illustrate baseJc on the matrix object m. This matrix has `center` baseJc: $\begin{pmatrix}1 & 2\\3 & 4\\5 & 6\end{pmatrix}$ This matrix has `top` baseJc: $\begin{pmatrix}1 & 2\\3 & 4\\5 & 6\end{pmatrix}$ This matrix has `bot` baseJc: $\begin{pmatrix}1 & 2\\3 & 4\\5 & 6\end{pmatrix}$ The possible values for this attribute are defined by the ST_YAlign simple type (§7.1.3.19).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_YAlign">
   <attribute name="val" type="ST_YAlign" use="required"/>
</complexType>
```

7.1.2.10 begChr (Delimiter Beginning Character)

This element specifies the beginning, or opening, delimiter character. Mathematical delimiters are enclosing characters such as parentheses, brackets, and braces. If this element is omitted, the default begChr is '('. [*Example*: In the following example, $\{a\}$ uses { and } as its enclosing characters:

```
<m:dPr>
  <m:begChr  m:val="{"/>
  <m:endChr m:val="}"/>
</m:dPr>
```

end example]

Parent Elements
dPr (§7.1.2.31)

Attributes	Description

val (value)	Specifies the character used by the parent element. When it is omitted, the parent uses its assigned default. [*Example*: delimiter object {a}:
	`<m:dPr>`
	`<m:begChr m:val="{"/>`
	`<m:endChr m:val="}"/>`
	`</m:dPr>`
	
	The possible values for this attribute are defined by the ST_Char simple type (§7.1.3.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Char">
   <attribute name="val" type="ST_Char" use="required"/>
</complexType>
```

7.1.2.11 borderBox (Border-Box Function)

This element specifies the Border Box function, consisting of a border drawn around an equation, as in $\boxed{a^2 + b^2 = c^2}$.

[*Example*: The following example shows the XML representation of the following Border Box: $\boxed{abc}$

```
<m:borderBox>
  <m:e>
    <m:r>
      <m:t>abc</m:t>
    </m:r>
  </m:e>
</m:borderBox>
```

end example]

Parent Elements
deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); e (§7.1.2.32); fName (§7.1.2.37); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); sub (§7.1.2.112); sup (§7.1.2.114)

Child Elements	Subclause
borderBoxPr (Border Box Properties)	§7.1.2.12
e (Base (Argument))	§7.1.2.32

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BorderBox">
   <sequence>
      <element name="borderBoxPr" type="CT_BorderBoxPr" minOccurs="0"/>
      <element name="e" type="CT_OMathArg"/>
   </sequence>
</complexType>
```

7.1.2.12 borderBoxPr (Border Box Properties)

This element specifies the properties of the Border Box function, which dictate the types of lines that can be drawn as part of the border. [*Example*: $\boxed{\cancel{a^2 + b^2 = c^2}}$ (Diagonal Strikethrough from Top Left) and $\underline{\overline{a^2 + b^2 = c^2}}$ (no left or right edges).

Parent Elements
borderBox (§7.1.2.11)

Child Elements	Subclause
ctrlPr (Control Properties)	§7.1.2.23
hideBot (Hide Bottom Edge)	§7.1.2.44
hideLeft (Hide Left Edge)	§7.1.2.45
hideRight (Hide Right Edge)	§7.1.2.46

Child Elements	Subclause
hideTop (Hide Top Edge)	§7.1.2.47
strikeBLTR (Border Box Strikethrough Bottom-Left to Top-Right)	§7.1.2.107
strikeH (Border Box Strikethrough Horizontal)	§7.1.2.108
strikeTLBR (Border Box Strikethrough Top-Left to Bottom-Right)	§7.1.2.109
strikeV (Border Box Strikethrough Vertical)	§7.1.2.110

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BorderBoxPr">
   <sequence>
      <element name="hideTop" type="CT_OnOff" minOccurs="0"/>
      <element name="hideBot" type="CT_OnOff" minOccurs="0"/>
      <element name="hideLeft" type="CT_OnOff" minOccurs="0"/>
      <element name="hideRight" type="CT_OnOff" minOccurs="0"/>
      <element name="strikeH" type="CT_OnOff" minOccurs="0"/>
      <element name="strikeV" type="CT_OnOff" minOccurs="0"/>
      <element name="strikeBLTR" type="CT_OnOff" minOccurs="0"/>
      <element name="strikeTLBR" type="CT_OnOff" minOccurs="0"/>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.13 box (Box Function)

This element specifies the box function, which is used to group components of an equation. A boxed object can (for example) serve as an operator emulator with or without an alignment point, serve as a line break point, have associated argSz, or be grouped such as not to allow line breaks within.

The equation $a == b$ uses a box around the double equal sign.

[*Example*: Its XML representation is as follows:

```
<m:r>
  <m:t>a</m:t>
</m:r>
<m:box>
  <m:boxPr>
    <m:opEmu    m:val="on"/>
    <m:aln/>
  </m:boxPr>
  <m:e>
      <m:r>
        <m:t>==</m:t>
      </m:r>
  </m:e>
</m:box>
```

```
<m:r>
  <m:t>b</m:t>
</m:r>
```

end example]

Parent Elements
deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); e (§7.1.2.32); fName (§7.1.2.37); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); sub (§7.1.2.112); sup (§7.1.2.114)

Child Elements	Subclause
boxPr (Box Properties)	§7.1.2.14
e (Base (Argument))	§7.1.2.32

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Box">
   <sequence>
      <element name="boxPr" type="CT_BoxPr" minOccurs="0"/>
      <element name="e" type="CT_OMathArg"/>
   </sequence>
</complexType>
```

7.1.2.14 boxPr (Box Properties)

This element specifies properties of the Box function, for example, whether the Box serves as operator emulator with or without an alignment point, serves as a line break point, or receives the correct spacing for the mathematical differential.

Parent Elements
box (§7.1.2.13)

Child Elements	Subclause
aln (Alignment)	§7.1.2.3
brk (Break)	§7.1.2.15
ctrlPr (Control Properties)	§7.1.2.23
diff (Differential)	§7.1.2.29
noBreak (No Break)	§7.1.2.73
opEmu (Operator Emulator)	§7.1.2.80

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BoxPr">
   <sequence>
      <element name="opEmu" type="CT_OnOff" minOccurs="0"/>
      <element name="noBreak" type="CT_OnOff" minOccurs="0"/>
      <element name="diff" type="CT_OnOff" minOccurs="0"/>
      <element name="brk" type="CT_ManualBreak" minOccurs="0"/>
      <element name="aln" type="CT_OnOff" minOccurs="0"/>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.15 brk (Break)

This element specifies whether there is a line break at the start of a run, or at the start of the Box function, such that the line wraps at the start of the run or function. If this element is omitted, a manual break is not inserted. The line may happen to wrap at this point if the equation exceeds the column width. [*Example*:The following example includes a manual line break at the operator emulator:

$$a \\ == b$$

```
<m:r>
  <m:t>a</m:t>
</m:r>
<m:box>
  <m:boxPr>
    <m:opEmu    m:val="on"/>
    <m:brk/>
  </m:boxPr>
  <m:e>
    <m:r>
      <m:t>==</m:t>
    </m:r>
  </m:e>
</m:box>
<m:r>
  <m:t>b</m:t>
</m:r>
```

end example]

Parent Elements
boxPr (§7.1.2.14); rPr (§7.1.2.91)

Attributes	Description
alnAt (Index of Operator to Align To)	Specifies the index of the operator on the previous line which shall be used as the alignment point for the current line. A line can be aligned to any operator on the previous line in the equation; this attribute specifies exactly which operator shall be the target of that alignment in cases where there are multiple operators. [*Example*: For example, consider the break in this equation: $$a + b + c + d + e \\ + f + g$$ The second line could theoretically be aligned to any of the four operators in the previous line. Specifying an alnAt value of 3 for the second line resolves this ambiguity; the second line is aligned to the third operator in the previous line. *end example*] The possible values for this attribute are defined by the ST_Integer255 simple type (§7.1.3.6).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_ManualBreak">
   <attribute name="alnAt" type="ST_Integer255"/>
</complexType>
```

7.1.2.16 brkBin (Break on Binary Operators)

This element specifies how binary operators are treated when they coincide with a line break. If this element is omitted, the line break occurs before the binary operator. That is, the binary operator is the first element on the wrapped line. [*Example*: For example:

$f(x) = a_{11} + a_{12} + \cdots \\ + a_{nn}$

Before

$f(x) = a_{11} + a_{12} + \cdots + \\ a_{nn}$

After

$f(x) = a_{11} + a_{12} + \cdots + \\ + a_{nn}$

Duplicate

The XML below demonstrates brkBin in use under mathPr:

```
<m:mathPr>
  <m:mathFont m:val="Cambria Math"/>
  <m:brkBin m:val="before"/>
  <m:brkBinSub m:val="--"/>
  <m:smallFrac m:val="off"/>
  <m:dispDef/>
```

```
    <m:lMargin m:val="0"/>
    <m:rMargin m:val="0"/>
    <m:defJc m:val="centerGroup"/>
    <m:wrapIndent m:val="1440"/>
    <m:intLim m:val="subSup"/>
    <m:naryLim m:val="undOvr"/>
  </m:mathPr>
```

end example]

Parent Elements
mathPr (§7.1.2.62)

Attributes	Description
val (Value)	Specifies where to break on binary operators. Possible values are `before`, `after`, and `repeat`. The possible values for this attribute are defined by the ST_BreakBin simple type (§7.1.3.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BreakBin">
   <attribute name="val" type="ST_BreakBin"/>
</complexType>
```

7.1.2.17 brkBinSub (Break on Binary Subtraction)

This element specifies how the subtraction operator is treated when it coincides with a line break, when brkBin is set to repeat. If this element is omitted, the subtraction operator is repeated before and after the break.

[*Example*: The XML below demonstrates brkBinSub in use under mathPr:

```
  <m:mathPr>
    <m:mathFont m:val="Cambria Math"/>
    <m:brkBin m:val="before"/>
    <m:brkBinSub m:val="--"/>
    <m:smallFrac m:val="off"/>
    <m:dispDef/>
    <m:lMargin m:val="0"/>
    <m:rMargin m:val="0"/>
    <m:defJc m:val="centerGroup"/>
    <m:wrapIndent m:val="1440"/>
    <m:intLim m:val="subSup"/>
    <m:naryLim m:val="undOvr"/>
  </m:mathPr>
```

end example]

Parent Elements
mathPr (§7.1.2.62)

Attributes	Description
val (Value)	Specifies how the subtraction operator is treated when it coincides with a line break, when `brkBin` is set to `repeat`. Possible values are `--`, `-+`, and `+-`. The possible values for this attribute are defined by the ST_BreakBinSub simple type (§7.1.3.2).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_BreakBinSub">
   <attribute name="val" type="ST_BreakBinSub"/>
</complexType>
```

7.1.2.18 cGp (Matrix Column Gap)

The additional (custom) column gap spacing information; the default is'0'. [*Example*: This matrix: $\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix}$ has .5 ems of additional spacing between columns. The matrix properties that demonstrate this element in use are:

```
<m:mPr>
  <m:cGpRule  m:val="4"/>
  <m:cGp  m:val="6"/>
  <m:mcs>
    <m:mc>
      <m:mcPr>
        <m:mcJc m:val="center"/>
        <m:count m:val="2"/>
      </m:mcPr>
    </m:mc>
  </m:mcs>
</m:mPr>
```

end example]

Parent Elements
mPr (§7.1.2.68)

Attributes	Description

Attributes	Description
val (Value)	Specifies the amount of space between the parent element. The manner in which this value is determined depends on the setting of the rule of the parent element. If the rule is set to 3 (or "Exactly"), then the unit is interpreted as points. If the rule is set to 4 (or "Multiple"), then the unit is interpreted as lines. The possible values for this attribute are defined by the ST_UnSignedInteger simple type (§7.1.3.17).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_UnSignedInteger">
   <attribute name="val" type="ST_UnSignedInteger" use="required"/>
</complexType>
```

7.1.2.19 cGpRule (Matrix Column Gap Rule)

This element specifies the type of horizontal spacing between columns in a matrix. Type of gap (horizontal spacing) between columns of a Matrix; the default is '0'. [*Example*: The following matrix has double spacing between columns:

$$\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix}$$

The XML that represents this property in use is:

```
<m:mPr>
  <m:cGpRule  m:val="2"/>
  <m:mcs>
    <m:mc>
      <m:mcPr>
        <m:mcJc  m:val="center"/>
        <m:count  m:val="2"/>
      </m:mcPr>
    </m:mc>
  </m:mcs>
</m:mPr>
```

end example]

Parent Elements
mPr (§7.1.2.68)

Attributes	Description
val (Value)	Specifies the type of spacing between rows and/or columns. Possible values are 0, 1, 2, 3, or 4, whose definitions are contained in the following table:

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Value</th><th>Column/Row Gap</th><th>Example</th></tr>
<tr><td>0</td><td>Single spacing gap</td><td>1 2</td></tr>
<tr><td>1</td><td>1.5 spacing gap</td><td>1 2</td></tr>
<tr><td>2</td><td>2 spacing gap</td><td>1 2</td></tr>
<tr><td>3</td><td>Exactly
(for columns, rely on value of cGp, measured in points)
(for rows, rely on value of rSp, measured in points)</td><td>1 2</td></tr>
<tr><td>4</td><td>Multiple
(for columns, rely on value of cGp, measured in lines)
(for rows, rely on value of rSp, measured in lines)</td><td>1 2</td></tr>
</table>

The possible values for this attribute are defined by the ST_SpacingRule simple type (§7.1.3.12).
</td></tr>
</table>

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SpacingRule">
   <attribute name="val" type="ST_SpacingRule" use="required"/>
</complexType>
```

7.1.2.20 chr (Accent Character)

This element specifies the type of combining diacritical mark attached to the base of the accent function. If this property is omitted, the default accent character is U+0302. [*Example*: Examples of accent characters are the dot, hat, and arrow in the following cases: $\dot{a}$ $\hat{a}$ $\vec{a}$.

For example, the following XML represents the acc $\tilde{a}$.

```
<m:acc>
  <m:accPr>
    <m:chr m:val="&#771;"/>
  </m:accPr>
  <m:e>
    <m:r>
      <m:t>a</m:t>
    </m:r>
  </m:e>
</m:acc>
```

end example]

Parent Elements
accPr (§7.1.2.2); groupChrPr (§7.1.2.42); naryPr (§7.1.2.72)

Attributes	Description

val (value)	Specifies the character used by the parent element. When it is omitted, the parent uses its assigned default. [*Example*: delimiter object {a}:
	`<m:dPr>`
	`<m:begChr m:val="{"/>`
	`<m:endChr m:val="}"/>`
	`</m:dPr>`
	
	The possible values for this attribute are defined by the ST_Char simple type (§7.1.3.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Char">
   <attribute name="val" type="ST_Char" use="required"/>
</complexType>
```

7.1.2.21 count (Matrix Column Count)

This element specifies the number of columns to which a property applies. [*Example*: The example below represents that two of the columns in the matrix described by the XML have the `center` property.

```
<m:mPr>
  <m:cSp  m:val="120"/>
  <m:mcs>
    <m:mc>
      <m:mcPr>
        <m:mcJc m:val="center"/>
        <m:count  m:val="2"/>
      </m:mcPr>
    </m:mc>
  </m:mcs>
</m:mPr>
```

end example]

Parent Elements
mcPr (§7.1.2.66)

Attributes	Description

Attributes	Description
val (Value)	Specifies the number of columns to which a column attribute applies. [Example: A count attribute value of 3 specifies that the property applies to the first three columns of the matrix. end example] The possible values for this attribute are defined by the ST_Integer255 simple type (§7.1.3.6).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Integer255">
   <attribute name="val" type="ST_Integer255" use="required"/>
</complexType>
```

7.1.2.22 cSp (Matrix Column Spacing)

This element specifies the minimum spacing between the edge of one column and the corresponding edge of the adjacent column. Additional spacing can be added to enhance appearance. If this element is omitted, the default is column spacing is '0'. [*Example*: The following matrix specifies that there should never be fewer than 6 pts. between adjacent column edges:

$$\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix}$$

```
<m:mPr>
  <m:cSp  m:val="120"/>
  <m:mcs>
    <m:mc>
      <m:mcPr>
        <m:mcJc m:val="center"/>
        <m:count  m:val="2"/>
      </m:mcPr>
    </m:mc>
  </m:mcs>
</m:mPr>
```

end example]

Parent Elements
mPr (§7.1.2.68)

Attributes	Description
val (Value)	Specifies the amount of space between the parent element. The manner in which this value is determined depends on the setting of the rule of the parent element. If the rule is set to 3 (or "Exactly"), then the unit is interpreted as points. If the rule is set to 4 (or

Attributes	Description
	"Multiple"), then the unit is interpreted as lines. The possible values for this attribute are defined by the ST_UnSignedInteger simple type (§7.1.3.17).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_UnSignedInteger">
   <attribute name="val" type="ST_UnSignedInteger" use="required"/>
</complexType>
```

7.1.2.23 ctrlPr (Control Properties)

This element specifies properties on control characters; that is, object characters that cannot be selected. Examples of control characters are n-ary operators (excluding their limits and bases), fraction bars (excluding the numerator and denominator), and grouping characters (excluding the base). ctrlPr allows formatting properties to be stored on these control characters. The control character inherits its formatting from the paragraph formatting; ctrlPr contains the formatting differences between the control character and the paragraph formatting.

[*Example*: The example below shows that the control character is of font Cambria Math. All other formatting, such as text size and color, are the same as the paragraph.

```
<m:ctrlPr>
  <w:rPr>
    <w:rFonts w:ascii="Cambria Math" w:hAnsi="Cambria Math"/>
  </w:rPr>
</m:ctrlPr>
```

end example]

CtrlPr is also used to save properties on characters used in the Linear representation of an equation, that are not displayed in the Professional form. For example, the linear string ∫_0^1 might have color on the _ or ^. Though these characters are not displayed in Professional form, their formatting is stored.

Parent Elements
accPr (§7.1.2.2); barPr (§7.1.2.8); borderBoxPr (§7.1.2.12); boxPr (§7.1.2.14); deg (§7.1.2.26); den (§7.1.2.28); dPr (§7.1.2.31); e (§7.1.2.32); eqArrPr (§7.1.2.35); fName (§7.1.2.37); fPr (§7.1.2.38); funcPr (§7.1.2.40); groupChrPr (§7.1.2.42); lim (§7.1.2.52); limLowPr (§7.1.2.55); limUppPr (§7.1.2.57); mPr (§7.1.2.68); naryPr (§7.1.2.72); num (§7.1.2.75); phantPr (§7.1.2.82); radPr (§7.1.2.89); sPrePr (§7.1.2.100); sSubPr (§7.1.2.102); sSubSupPr (§7.1.2.104); sSupPr (§7.1.2.106); sub (§7.1.2.112); sup (§7.1.2.114)

Child Elements	Subclause
del (Deleted Math Control Character)	§2.13.5.15

Child Elements	Subclause
ins (Inserted Math Control Character)	§2.13.5.17
rPr (Run Properties)	§2.3.2.25

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_CtrlPr">
   <sequence>
      <group ref="w:EG_RPrMath" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.24 d (Delimiter Function)

This element specifies the delimiter function, consisting of opening and closing delimiters (such as parentheses, braces, brackets, and vertical bars), and an element contained inside. The delimiter may have more than one element, with a designated separator character between each element. [*Example*:

Delimiter with one base: (x^2)

Delimiter with more than one base and separators, whose XML is shown below: $(x^2|y^2)$

```
<m:d>
  <m:e>
    <m:sSup>
      <m:e>
        <m:r>
          <m:t>x</m:t>
        </m:r>
      </m:e>
      <m:sup>
        <m:r>
          <m:rPr>
            <m:scr  m:val="roman"/>
            <m:sty  m:val="p"/>
          </m:rPr>
          <m:t>2</m:t>
        </m:r>
      </m:sup>
    </m:sSup>
  </m:e>
```

```
    <m:e>
      <m:sSup>
        <m:e>
          <m:r>
            <m:t>y</m:t>
          </m:r>
        </m:e>
        <m:sup>
          <m:r>
            <m:rPr>
              <m:scr m:val="roman"/>
              <m:sty  m:val="p"/>
            </m:rPr>
            <m:t>2</m:t>
          </m:r>
        </m:sup>
      </m:sSup>
    </m:e>
  </m:d>
```

end example]

Parent Elements
deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); e (§7.1.2.32); fName (§7.1.2.37); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); sub (§7.1.2.112); sup (§7.1.2.114)

Child Elements	Subclause
dPr (Delimiter Properties)	§7.1.2.31
e (Base (Argument))	§7.1.2.32

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_D">
   <sequence>
      <element name="dPr" type="CT_DPr" minOccurs="0"/>
      <element name="e" type="CT_OMathArg" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

7.1.2.25 defJc (Default Justification)

This element specifies the default justification of display math, at the document level. Individual equations can overrule the default setting.

Display math can be left justified, right justified, centered, or centered as a group. When display math is centered as a group, the equations are left aligned within a block, and the entire block is centered with respect to column margins. If this element is omitted, the equations are centered as a group. [*Example*: The XML below demonstrates defJc in use:

```
<m:mathPr>
  <m:mathFont m:val="Cambria Math"/>
  <m:brkBin m:val="before"/>
  <m:brkBinSub m:val="--"/>
  <m:smallFrac m:val="off"/>
  <m:dispDef/>
  <m:lMargin m:val="0"/>
  <m:rMargin m:val="0"/>
  <m:defJc m:val="centerGroup"/>
  <m:wrapIndent m:val="1440"/>
  <m:intLim m:val="subSup"/>
  <m:naryLim m:val="undOvr"/>
</m:mathPr>
```

end example]

Parent Elements
mathPr (§7.1.2.62)

Attributes	Description
val (Value)	Specifies the default justification of equations in the document. Possible values are `center`, `centerGroup`, `left`, and `right`. The possible values for this attribute are defined by the ST_Jc simple type (§7.1.3.7).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OMathJc">
   <attribute name="val" type="ST_Jc"/>
</complexType>
```

7.1.2.26 deg (Degree)

This element specifies the degree in the mathematical radical, for example the 3 in $\sqrt[3]{x}$ (XML representation is below). This element is optional. When omitted, the square root function, as in $\sqrt{x}$, is assumed. [*Example*:

```
<m:rad>
  <m:deg>
     <m:r>
        <m:rPr>
          <m:scr   m:val="roman"/>
          <m:sty   m:val="p"/>
        </m:rPr>
        <m:t>3</m:t>
      </m:r>
  </m:deg>
  <m:e>
    <m:r>
      <m:t>x</m:t>
    </m:r>
  </m:e>
</m:rad>
```

end example]

Parent Elements
rad (§7.1.2.88)

Child Elements	Subclause
acc (Accent)	§7.1.2.1
argPr (Argument Properties)	§7.1.2.5
bar (Bar)	§7.1.2.7
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
borderBox (Border-Box Function)	§7.1.2.11
box (Box Function)	§7.1.2.13
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
ctrlPr (Control Properties)	§7.1.2.23
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9

Child Elements	Subclause
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
d (Delimiter Function)	§7.1.2.24
del (Deleted Run Content)	§2.13.5.12
eqArr (Equation-Array Function)	§7.1.2.34
f (Fraction Function)	§7.1.2.36
func (Function Apply Function)	§7.1.2.39
groupChr (Group-Character Function)	§7.1.2.41
ins (Inserted Run Content)	§2.13.5.20
limLow (Lower-Limit Function)	§7.1.2.54
limUpp (Upper-Limit Function)	§7.1.2.56
m (Matrix Function)	§7.1.2.60
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
nary (n-ary Operator Function)	§7.1.2.70
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
phant (Phantom Function)	§7.1.2.81
proofErr (Proofing Error Anchor)	§2.13.8.1
r (Run)	§7.1.2.87
rad (Radical Function)	§7.1.2.88
sPre (Pre-Sub-Superscript Function)	§7.1.2.99
sSub (Subscript Function)	§7.1.2.101
sSubSup (Sub-Superscript Function)	§7.1.2.103
sSup (Superscript Function)	§7.1.2.105

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OMathArg">
   <sequence>
      <element name="argPr" type="CT_OMathArgPr" minOccurs="0"/>
      <group ref="EG_OMathElements" minOccurs="0" maxOccurs="unbounded"/>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.27 degHide (Hide Degree)

This element specifies the per-object option to hide the degree of a radical. Every rad has a deg, but the deg can appear or not appear. When degHide is set to 'on,' the degree is not shown, as in $\sqrt{x}$ (XML shown below). When degHide is omitted, the default is 'off'; that is, the degree is not hidden. [*Example*:

```
<m:rad>
  <m:radPr>
    <m:degHide  m:val="on"/>
  </m:radPr>
  <m:deg>
  </m:deg>
  <m:e>
    <m:r>
      <m:t>x</m:t>
    </m:r>
  </m:e>
</m:rad>
```

end example]

Parent Elements
radPr (§7.1.2.89)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.28 den (Denominator)

This element specifies the denominator of a fraction. [*Example*: For example, the b in $^{a}/_{b}$:

```
<m:f>
  <m:fPr>
    <m:type  m:val="skw"/>
  </m:fPr>
  <m:num>
    <m:r>
      <m:t>a</m:t>
    </m:r>
  </m:num>
  <m:den>
    <m:r>
      <m:t>b</m:t>
    </m:r>
  </m:den>
</m:f>
```

end example]

Parent Elements
f (§7.1.2.36)

Child Elements	Subclause
acc (Accent)	§7.1.2.1
argPr (Argument Properties)	§7.1.2.5
bar (Bar)	§7.1.2.7
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
borderBox (Border-Box Function)	§7.1.2.11
box (Box Function)	§7.1.2.13
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
ctrlPr (Control Properties)	§7.1.2.23

Child Elements	Subclause
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
d (Delimiter Function)	§7.1.2.24
del (Deleted Run Content)	§2.13.5.12
eqArr (Equation-Array Function)	§7.1.2.34
f (Fraction Function)	§7.1.2.36
func (Function Apply Function)	§7.1.2.39
groupChr (Group-Character Function)	§7.1.2.41
ins (Inserted Run Content)	§2.13.5.20
limLow (Lower-Limit Function)	§7.1.2.54
limUpp (Upper-Limit Function)	§7.1.2.56
m (Matrix Function)	§7.1.2.60
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
nary (n-ary Operator Function)	§7.1.2.70
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
phant (Phantom Function)	§7.1.2.81
proofErr (Proofing Error Anchor)	§2.13.8.1
r (Run)	§7.1.2.87
rad (Radical Function)	§7.1.2.88
sPre (Pre-Sub-Superscript Function)	§7.1.2.99

Child Elements	Subclause
sSub (Subscript Function)	§7.1.2.101
sSubSup (Sub-Superscript Function)	§7.1.2.103
sSup (Superscript Function)	§7.1.2.105

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OMathArg">
   <sequence>
      <element name="argPr" type="CT_OMathArgPr" minOccurs="0"/>
      <group ref="EG_OMathElements" minOccurs="0" maxOccurs="unbounded"/>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.29 diff (Differential)

The element specifies the differential property on box. When 'on,' the box acts as a differential (e.g., dx in an integrand), and receives the appropriate horizontal spacing for the mathematical differential. When this property is omitted, the box is not treated as a differential. [*Example*: The following example demonstrates a box set as differential in use, both in its proper form and in XML:

$$\int_0^1 x\,dx$$

```
<m:nary>
  <m:naryPr>
    <m:chr m:val="&#8747;"/>
  </m:naryPr>
  <m:sub>
      <m:r>
        <m:rPr>
          <m:scr m:val="roman"/>
          <m:sty m:val="p"/>
        </m:rPr>
        <m:t>0</m:t>
      </m:r>
  </m:sub>
```

```
<m:sup>
    <m:r>
      <m:rPr>
        <m:scr  m:val="roman"/>
        <m:sty  m:val="p"/>
      </m:rPr>
      <m:t>1</m:t>
    </m:r>
</m:sup>
<m:e>
  <m:r>
    <m:t>x</m:t>
  </m:r>
  <m:box>
    <m:boxPr>
      <m:diff  m:val="on"/>
    </m:boxPr>
    <m:e>
      <m:r>
        <m:t>dx</m:t>
      </m:r>
    </m:e>
  </m:box>
</m:e>
</m:nary>
```

end example]

Parent Elements
boxPr (§7.1.2.14)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.30 dispDef (Use Display Math Defaults)

This element specifies the document-level property to overwrite paragraph settings for equations. When omitted, this element is set to 'on' and special math settings are applied.

[*Example*: The XML below demonstrates dispDef in use:

```
<m:mathPr>
  <m:mathFont m:val="Cambria Math"/>
  <m:brkBin m:val="before"/>
  <m:brkBinSub m:val="--"/>
  <m:smallFrac m:val="off"/>
  <m:dispDef/>
  <m:lMargin m:val="0"/>
  <m:rMargin m:val="0"/>
  <m:defJc m:val="centerGroup"/>
  <m:wrapIndent m:val="1440"/>
  <m:intLim m:val="subSup"/>
  <m:naryLim m:val="undOvr"/>
</m:mathPr>
```

end example]

Parent Elements
mathPr (§7.1.2.62)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.31 dPr (Delimiter Properties)

This element specifies the properties of d, including the enclosing and separating characters, and the properties that affect the shape of the delimiters.

Parent Elements
d (§7.1.2.24)

Child Elements	Subclause
begChr (Delimiter Beginning Character)	§7.1.2.10
ctrlPr (Control Properties)	§7.1.2.23
endChr (Delimiter Ending Character)	§7.1.2.33
grow (n-ary Grow)	§7.1.2.43
sepChr (Delimiter Separator Character)	§7.1.2.95
shp (Shape (Delimiters))	§7.1.2.97

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DPr">
   <sequence>
      <element name="begChr" type="CT_Char" minOccurs="0"/>
      <element name="sepChr" type="CT_Char" minOccurs="0"/>
      <element name="endChr" type="CT_Char" minOccurs="0"/>
      <element name="grow" type="CT_OnOff" minOccurs="0"/>
      <element name="shp" type="CT_Shp" minOccurs="0"/>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.32 e (Base (Argument))

This element specifies the base argument of a mathematical function.

[*Example*: For example, the func $\lim_{n\to\infty} x_n$ has fName $\lim_{n\to\infty}$ and e x_n:

```
<m:func>
  <m:fName>
    <m:limLow>
      <m:e>
        <m:r>
          <m:rPr>
            <m:scr m:val="roman"/>
            <m:sty  m:val="p"/>
          </m:rPr>
          <m:t>lim</m:t>
        </m:r>
      </m:e>
      <m:lim>
        <m:r>
          <m:t>n&#8594;&#8734;</m:t>
        </m:r>
      </m:lim>
    </m:limLow>
  </m:fName>
  <m:e>
    <m:sSub>
      <m:e>
        <m:r>
          <m:t>x</m:t>
        </m:r>
      </m:e>
      <m:sub>
        <m:r>
          <m:t>n</m:t>
        </m:r>
      </m:sub>
    </m:sSub>
  </m:e>
</m:func>
```

end example]

Parent Elements
acc (§7.1.2.1); bar (§7.1.2.7); borderBox (§7.1.2.11); box (§7.1.2.13); d (§7.1.2.24); eqArr (§7.1.2.34); func (§7.1.2.39); groupChr (§7.1.2.41); limLow (§7.1.2.54); limUpp (§7.1.2.56); mr (§7.1.2.69); nary (§7.1.2.70); phant (§7.1.2.81); rad (§7.1.2.88); sPre (§7.1.2.99); sSub (§7.1.2.101); sSubSup (§7.1.2.103); sSup (§7.1.2.105)

Child Elements	Subclause

Child Elements	Subclause
acc (Accent)	§7.1.2.1
argPr (Argument Properties)	§7.1.2.5
bar (Bar)	§7.1.2.7
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
borderBox (Border-Box Function)	§7.1.2.11
box (Box Function)	§7.1.2.13
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
ctrlPr (Control Properties)	§7.1.2.23
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
d (Delimiter Function)	§7.1.2.24
del (Deleted Run Content)	§2.13.5.12
eqArr (Equation-Array Function)	§7.1.2.34
f (Fraction Function)	§7.1.2.36
func (Function Apply Function)	§7.1.2.39
groupChr (Group-Character Function)	§7.1.2.41
ins (Inserted Run Content)	§2.13.5.20
limLow (Lower-Limit Function)	§7.1.2.54
limUpp (Upper-Limit Function)	§7.1.2.56
m (Matrix Function)	§7.1.2.60
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28

Child Elements	Subclause
nary (n-ary Operator Function)	§7.1.2.70
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
phant (Phantom Function)	§7.1.2.81
proofErr (Proofing Error Anchor)	§2.13.8.1
r (Run)	§7.1.2.87
rad (Radical Function)	§7.1.2.88
sPre (Pre-Sub-Superscript Function)	§7.1.2.99
sSub (Subscript Function)	§7.1.2.101
sSubSup (Sub-Superscript Function)	§7.1.2.103
sSup (Superscript Function)	§7.1.2.105

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OMathArg">
   <sequence>
      <element name="argPr" type="CT_OMathArgPr" minOccurs="0"/>
      <group ref="EG_OMathElements" minOccurs="0" maxOccurs="unbounded"/>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.33 endChr (Delimiter Ending Character)

This element specifies the ending, or closing, delimiter character. Mathematical delimiters are enclosing characters such as parentheses, brackets, and braces. If this element is omitted, the default endChr is ')'.

[*Example*: In the following example, $\{a\}$ uses { and } as its enclosing characters:

```
<m:dPr>
  <m:begChr  m:val="{"/>
  <m:endChr m:val="}"/>
</m:dPr>
```

end example]

Parent Elements
dPr (§7.1.2.31)

Attributes	Description

Attributes	Description

val (value)	Specifies the character used by the parent element. When it is omitted, the parent uses its assigned default. [*Example*: delimiter object {a}:
	`<m:dPr>`
	`<m:begChr m:val="{"/>`
	`<m:endChr m:val="}"/>`
	`</m:dPr>`
	
	The possible values for this attribute are defined by the ST_Char simple type (§7.1.3.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Char">
   <attribute name="val" type="ST_Char" use="required"/>
</complexType>
```

7.1.2.34 eqArr (Equation-Array Function)

This element specifies the Equation-Array function, an object consisting of one or more equations that can be vertically justified as a unit respect to surrounding text on the line. Alignment of multiple points within each equation can occur within the equation array. [*Example*: An example of an equation array with alignment points is:

$$\begin{aligned} x- \quad y+ \quad z= \quad 10 \\ 3x+ \quad y+2z= \quad 34 \\ -5x+2y- \quad z=-14 \end{aligned}$$

Notice that the variables, operators, and tens digits of the sums line up properly.

The XML of a simple eqArr $\begin{matrix} a=b+c \\ d+e=f \end{matrix}$ is:

```
<m:eqArr>
  <m:e>
    <m:r>
      <m:t>a=b+c</m:t>
    </m:r>
  </m:e>
  <m:e>
    <m:r>
      <m:t>d+e=f</m:t>
    </m:r>
  </m:e>
</m:eqArr>
```

end example]

Parent Elements

Parent Elements
deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); e (§7.1.2.32); fName (§7.1.2.37); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); sub (§7.1.2.112); sup (§7.1.2.114)

Child Elements	Subclause
e (Base (Argument))	§7.1.2.32
eqArrPr (Equation Array Properties)	§7.1.2.35

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EqArr">
   <sequence>
      <element name="eqArrPr" type="CT_EqArrPr" minOccurs="0"/>
      <element name="e" type="CT_OMathArg" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

7.1.2.35 eqArrPr (Equation Array Properties)

This element specifies the properties of the equation array object, including the vertical justification of the object and layout inside the object.

Parent Elements
eqArr (§7.1.2.34)

Child Elements	Subclause
baseJc (Matrix Base Justification)	§7.1.2.9
ctrlPr (Control Properties)	§7.1.2.23
maxDist (Maximum Distribution)	§7.1.2.63
objDist (Object Distribution)	§7.1.2.76
rSp (Row Spacing (Equation Array))	§7.1.2.92
rSpRule (Row Spacing Rule)	§7.1.2.93

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_EqArrPr">
   <sequence>
      <element name="baseJc" type="CT_YAlign" minOccurs="0"/>
      <element name="maxDist" type="CT_OnOff" minOccurs="0"/>
      <element name="objDist" type="CT_OnOff" minOccurs="0"/>
      <element name="rSpRule" type="CT_SpacingRule" minOccurs="0"/>
      <element name="rSp" type="CT_UnSignedInteger" minOccurs="0"/>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.36 f (Fraction Function)

This element specifies the fraction object, consisting of a numerator and denominator separated by a fraction bar. The fraction bar can be horizontal or diagonal, depending on the fraction properties. The fraction object is also used to represent the stack function, which places one element above another, with no fraction bar. [*Example*: Examples of fractions are:

Stacked Fraction: $\frac{a}{b}$

Skewed Fraction: a/b

Linear Fraction: a/b

Stack Object (No-Bar Fraction): $\begin{matrix} n \\ k \end{matrix}$

The fraction a/b is represented as:

```
<m:f>
  <m:fPr>
    <m:type  m:val="skw"/>
  </m:fPr>
  <m:num>
    <m:r>
      <m:t>a</m:t>
    </m:r>
  </m:num>
  <m:den>
    <m:r>
      <m:t>b</m:t>
    </m:r>
  </m:den>
</m:f>
```

end example]

Parent Elements
deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); e (§7.1.2.32); fName (§7.1.2.37); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); sub (§7.1.2.112); sup (§7.1.2.114)

Child Elements	Subclause
den (Denominator)	§7.1.2.28
fPr (Fraction Properties)	§7.1.2.38
num (Numerator)	§7.1.2.75

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_F">
   <sequence>
      <element name="fPr" type="CT_FPr" minOccurs="0"/>
      <element name="num" type="CT_OMathArg"/>
      <element name="den" type="CT_OMathArg"/>
   </sequence>
</complexType>
```

7.1.2.37 fName (Function Name)

This element specifies the name of the function in the Function-Apply object func. For example, function names are sin and cos.

[*Example*: As an example, the func $\lim_{n\to\infty} x_n$ has fName $\lim_{n\to\infty}$ and e x_n:

```
<m:func>
  <m:fName>
    <m:limLow>
      <m:e>
        <m:r>
          <m:rPr>
            <m:scr m:val="roman"/>
            <m:sty  m:val="p"/>
          </m:rPr>
          <m:t>lim</m:t>
        </m:r>
      </m:e>
```

```
        <m:lim>
          <m:r>
            <m:t>n&#8594;&#8734;</m:t>
          </m:r>
        </m:lim>
      </m:limLow>
    </m:fName>
    <m:e>
      <m:sSub>
        <m:e>
          <m:r>
            <m:t>x</m:t>
          </m:r>
        </m:e>
        <m:sub>
          <m:r>
            <m:t>n</m:t>
          </m:r>
        </m:sub>
      </m:sSub>
    </m:e>
  </m:func>
```

end example]

Parent Elements
func (§7.1.2.39)

Child Elements	Subclause
acc (Accent)	§7.1.2.1
argPr (Argument Properties)	§7.1.2.5
bar (Bar)	§7.1.2.7
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
borderBox (Border-Box Function)	§7.1.2.11
box (Box Function)	§7.1.2.13
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
ctrlPr (Control Properties)	§7.1.2.23
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4

Child Elements	Subclause
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
d (Delimiter Function)	§7.1.2.24
del (Deleted Run Content)	§2.13.5.12
eqArr (Equation-Array Function)	§7.1.2.34
f (Fraction Function)	§7.1.2.36
func (Function Apply Function)	§7.1.2.39
groupChr (Group-Character Function)	§7.1.2.41
ins (Inserted Run Content)	§2.13.5.20
limLow (Lower-Limit Function)	§7.1.2.54
limUpp (Upper-Limit Function)	§7.1.2.56
m (Matrix Function)	§7.1.2.60
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
nary (n-ary Operator Function)	§7.1.2.70
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
phant (Phantom Function)	§7.1.2.81
proofErr (Proofing Error Anchor)	§2.13.8.1
r (Run)	§7.1.2.87
rad (Radical Function)	§7.1.2.88
sPre (Pre-Sub-Superscript Function)	§7.1.2.99
sSub (Subscript Function)	§7.1.2.101

Child Elements	Subclause
sSubSup (Sub-Superscript Function)	§7.1.2.103
sSup (Superscript Function)	§7.1.2.105

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OMathArg">
    <sequence>
        <element name="argPr" type="CT_OMathArgPr" minOccurs="0"/>
        <group ref="EG_OMathElements" minOccurs="0" maxOccurs="unbounded"/>
        <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
    </sequence>
</complexType>
```

7.1.2.38 fPr (Fraction Properties)

This element specifies the properties of the fraction function f. Properties of the Fraction function include the type or style of the fraction. The fraction bar can be horizontal or diagonal, depending on the fraction properties. The fraction object is also used to represent the stack function, which places one element above another, with no fraction bar. [*Example*: Examples of fractions are:

Stacked Fraction: $\frac{a}{b}$

Skewed Fraction: a/b

Linear Fraction: a/b

Stack Object (No-Bar Fraction): $\binom{n}{k}$

end example]

Parent Elements
f (§7.1.2.36)

Child Elements	Subclause
ctrlPr (Control Properties)	§7.1.2.23
type (Fraction type)	§7.1.2.118

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FPr">
    <sequence>
        <element name="type" type="CT_FType" minOccurs="0"/>
        <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
    </sequence>
</complexType>
```

7.1.2.39 func (Function Apply Function)

This element specifies the Function-Apply function, which consists of a function name and an argument acted upon. [*Example*: Examples of Function-Apply objects include: $\sin x$, $\tan^{-1} x^2$, and $\max_{0\le x\le 1} xe^{-x^2}$.

As an example, the func $\lim_{n\to\infty} x_n$ has fName $\lim_{n\to\infty}$ and e x_n:

```
<m:func>
  <m:fName>
    <m:limLow>
      <m:e>
        <m:r>
          <m:rPr>
            <m:scr m:val="roman"/>
            <m:sty  m:val="p"/>
          </m:rPr>
          <m:t>lim</m:t>
        </m:r>
      </m:e>
      <m:lim>
        <m:r>
          <m:t>n&#8594;&#8734;</m:t>
        </m:r>
      </m:lim>
    </m:limLow>
  </m:fName>
  <m:e>
    <m:sSub>
      <m:e>
        <m:r>
          <m:t>x</m:t>
        </m:r>
      </m:e>
      <m:sub>
        <m:r>
          <m:t>n</m:t>
        </m:r>
      </m:sub>
    </m:sSub>
  </m:e>
</m:func>
```

end example]

Parent Elements
deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); e (§7.1.2.32); fName (§7.1.2.37); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); sub (§7.1.2.112); sup (§7.1.2.114)

Child Elements	Subclause
e (Base (Argument))	§7.1.2.32
fName (Function Name)	§7.1.2.37
funcPr (Function Properties)	§7.1.2.40

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Func">
   <sequence>
      <element name="funcPr" type="CT_FuncPr" minOccurs="0"/>
      <element name="fName" type="CT_OMathArg"/>
      <element name="e" type="CT_OMathArg"/>
   </sequence>
</complexType>
```

7.1.2.40 funcPr (Function Properties)

This element specifies properties such as ctrlPr that can be stored on the function apply object func.

Parent Elements
func (§7.1.2.39)

Child Elements	Subclause
ctrlPr (Control Properties)	§7.1.2.23

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FuncPr">
   <sequence>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.41 groupChr (Group-Character Function)

This element specifies the Group-Character function, consisting of a character drawn above or below text, often with the purpose of visually grouping items. [*Example*: The following example demonstrates the groupChr in use, both in its proper form and in XML:

$$\overbrace{x+x+\cdots}$$

```
<m:groupChr>
  <m:groupChrPr>
    <m:chr  m:val="&#9182;"/>
    <m:pos  m:val="top"/>
  </m:groupChrPr>
  <m:e>
    <m:r>
      <m:t>x+x+…</m:t>
    </m:r>
  </m:e>
</m:groupChr>
```

end example]

Parent Elements
deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); e (§7.1.2.32); fName (§7.1.2.37); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); sub (§7.1.2.112); sup (§7.1.2.114)

Child Elements	Subclause
e (Base (Argument))	§7.1.2.32
groupChrPr (Group-Character Properties)	§7.1.2.42

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GroupChr">
   <sequence>
      <element name="groupChrPr" type="CT_GroupChrPr" minOccurs="0"/>
      <element name="e" type="CT_OMathArg"/>
   </sequence>
</complexType>
```

7.1.2.42 groupChrPr (Group-Character Properties)

This element specifies the properties of the Group-Character function groupChr. These properties can be used to specify the character placed above or below the argument, and the position of the character. When omitted, U+23DF is used as the chr and its pos is set to bot.

Parent Elements
groupChr (§7.1.2.41)

Child Elements	Subclause
chr (Accent Character)	§7.1.2.20

Child Elements	Subclause
ctrlPr (Control Properties)	§7.1.2.23
pos (Position (Bar))	§7.1.2.84
vertJc (Vertical Justification)	§7.1.2.119

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_GroupChrPr">
   <sequence>
      <element name="chr" type="CT_Char" minOccurs="0"/>
      <element name="pos" type="CT_TopBot" minOccurs="0"/>
      <element name="vertJc" type="CT_TopBot" minOccurs="0"/>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.43 grow (n-ary Grow)

This element specifies the growth property of n-ary operators at the document level. When off, n-ary operators such as integrals and summations do not grow to match the size of their operand height. When on, the n-ary operator grows vertically to match its operand height. If this property is omitted, grow is set to off. [*Example*: The two integrals below demonstrate the difference between grow = off and grow = on.

$$\int_0^1 \frac{x^2}{x+y}dx \int_0^1 \frac{y^2}{x+y}dy$$

The XML that defines nary growth is:

```
<m:naryPr>
  <m:chr  m:val="&#8747;"/>
  <m:grow m:val="on"/>
</m:naryPr>
```

end example]

Parent Elements
dPr (§7.1.2.31); naryPr (§7.1.2.72)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present.

Attributes	Description
	A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.44 hideBot (Hide Bottom Edge)

This element specifies the hidden or shown state of the bottom edge of borderBox. When this element is omitted, the bottom edge is shown. When on, the bottom border is hidden, as in $\overline{abc}$. [*Example*:

```
<m:borderBox>
  <m:borderBoxPr>
    <m:hideBot/>
  </m:borderBoxPr>
  <m:e>
    <m:r>
      <m:t>abc</m:t>
    </m:r>
  </m:e>
</m:borderBox>
```

end example]

Parent Elements
borderBoxPr (§7.1.2.12)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.45 hideLeft (Hide Left Edge)

This element specifies the hidden or shown state of the left edge of borderBox. When this element is omitted, the edge is shown. When on, the left border is hidden, as in $\boxed{abc}$. [*Example*:

```
<m:borderBox>
  <m:borderBoxPr>
    <m:hideLeft/>
  </m:borderBoxPr>
  <m:e>
    <m:r>
      <m:t>abc</m:t>
    </m:r>
  </m:e>
</m:borderBox>
```

end example]

Parent Elements
borderBoxPr (§7.1.2.12)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.46 hideRight (Hide Right Edge)

This element specifies the hidden or shown state of the right edge of borderBox. When this element is omitted, the edge is shown. When on, the right border is hidden, as in $\boxed{abc}$. [*Example*:

```
<m:borderBox>
  <m:borderBoxPr>
    <m:hideRight/>
  </m:borderBoxPr>
  <m:e>
    <m:r>
      <m:t>abc</m:t>
    </m:r>
  </m:e>
</m:borderBox>
```

end example]

Parent Elements
borderBoxPr (§7.1.2.12)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.47 hideTop (Hide Top Edge)

This element specifies the hidden or shown state of the top edge of borderBox. When this element is omitted, the edge is shown.

When on, the top border is hidden, as in $\underline{abc}$. [*Example*:

```
<m:borderBox>
  <m:borderBoxPr>
    <m:hideTop/>
  </m:borderBoxPr>
```

```
    <m:e>
      <m:r>
        <m:t>abc</m:t>
      </m:r>
    </m:e>
  </m:borderBox>
```

end example]

Parent Elements
borderBoxPr (§7.1.2.12)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.48 interSp (Inter-Equation Spacing)

This element specifies spacing between equations within a display math paragraph, in twips.

Parent Elements
mathPr (§7.1.2.62)

Attributes	Description
val (Value)	Specifies the value, in twips, of the parent element. The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§7.1.3.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TwipsMeasure">
   <attribute name="val" type="ST_TwipsMeasure" use="required"/>
</complexType>
```

7.1.2.49 intLim (Integral Limit Locations)

This element specifies the document setting for the default placement of integral limits, when converted from a linear format to a two-dimensional output. Limits can be either centered above and below the integral, or positioned just to the right of the operator, as in:

$$\int_a^b x\,dx \quad \int_a^b x\,dx$$

When this integral object is written linearly, as $\int \_a\^b$, the placement of limits is ambiguous. intLim is a document-level property that specifies the default positioning. When this element is omitted, the default placement of integral limits is subSup (that is, the location of subscripts and superscripts, or just to the right of the base or operator).

[*Example*: The XML that specifies this property in use is:

```
<m:mathPr>
  <m:mathFont m:val="Cambria Math"/>
  <m:brkBin m:val="before"/>
  <m:brkBinSub m:val="--"/>
  <m:smallFrac m:val="off"/>
  <m:dispDef/>
  <m:lMargin m:val="0"/>
  <m:rMargin m:val="0"/>
  <m:defJc m:val="centerGroup"/>
  <m:wrapIndent m:val="1440"/>
  <m:intLim m:val="subSup"/>
  <m:naryLim m:val="undOvr"/>
</m:mathPr>
```

end example]

Parent Elements
mathPr (§7.1.2.62)

Attributes	Description
val (Value)	Specifies the default location of limits on an integral. Possible values are subSup and undOvr.

Attributes	Description
	The possible values for this attribute are defined by the ST_LimLoc simple type (§7.1.3.8).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LimLoc">
   <attribute name="val" type="ST_LimLoc" use="required"/>
</complexType>
```

7.1.2.50 intraSp (Intra-Equation Spacing)

This element specifies the spacing between adjacent display math paragraphs, in twips.

Parent Elements
mathPr (§7.1.2.62)

Attributes	Description
val (Value)	Specifies the value, in twips, of the parent element. The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§7.1.3.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TwipsMeasure">
   <attribute name="val" type="ST_TwipsMeasure" use="required"/>
</complexType>
```

7.1.2.51 jc (Justification)

This element specifies justification of the math paragraph (a series of adjacent equations within the same paragraph). A math paragraph can be Left Justified, Right Justified, Centered, or Centered as Group. If this element is omitted, the math paragraph is Centered as Group. This means that the equations can be aligned with respect to each other, but the entire group of equations is centered as a whole. [*Example*: An example of Centered as Group is the following example, in which each equation is left-aligned, but the series is centered:

$$x = x_1 + x_2 + x_3 + \cdots$$
$$y = y_1 + y_2 + y_3 + y_4 + \cdots$$
$$z = z_1 + z_2 + z_3 + z_4 + z_5 + \cdots$$

The XML that demonstrates jc in use is:

```
<m:oMathParaPr>
  <m:jc m:val="right"/>
</m:oMathParaPr>
```

end example]

Parent Elements
oMathParaPr (§7.1.2.79)

Attributes	Description
val (Value)	Specifies the default justification of equations in the document. Possible values are `center`, `centerGroup`, `left`, and `right`. The possible values for this attribute are defined by the ST_Jc simple type (§7.1.3.7).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OMathJc">
   <attribute name="val" type="ST_Jc"/>
</complexType>
```

7.1.2.52 lim (Limit (Lower))

This element specifies the lower limit of the limLow function. [*Example*: For example, the limit of the limLow $\lim_{n\to\infty}$ is $n \to \infty$. The XML that specifies this function is:

```
<m:limLow>
  <m:e>
    <m:r>
      <m:rPr>
        <m:scr m:val="roman"/>
        <m:sty  m:val="p"/>
      </m:rPr>
      <m:t>lim</m:t>
    </m:r>
  </m:e>
  <m:lim>
    <m:r>
      <m:t>n&#8594;&#8734;</m:t>
    </m:r>
  </m:lim>
</m:limLow>
```

end example]

Parent Elements
limLow (§7.1.2.54); limUpp (§7.1.2.56)

Child Elements	Subclause

Child Elements	Subclause
acc (Accent)	§7.1.2.1
argPr (Argument Properties)	§7.1.2.5
bar (Bar)	§7.1.2.7
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
borderBox (Border-Box Function)	§7.1.2.11
box (Box Function)	§7.1.2.13
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
ctrlPr (Control Properties)	§7.1.2.23
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
d (Delimiter Function)	§7.1.2.24
del (Deleted Run Content)	§2.13.5.12
eqArr (Equation-Array Function)	§7.1.2.34
f (Fraction Function)	§7.1.2.36
func (Function Apply Function)	§7.1.2.39
groupChr (Group-Character Function)	§7.1.2.41
ins (Inserted Run Content)	§2.13.5.20
limLow (Lower-Limit Function)	§7.1.2.54
limUpp (Upper-Limit Function)	§7.1.2.56
m (Matrix Function)	§7.1.2.60
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28

Child Elements	Subclause
nary (n-ary Operator Function)	§7.1.2.70
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
phant (Phantom Function)	§7.1.2.81
proofErr (Proofing Error Anchor)	§2.13.8.1
r (Run)	§7.1.2.87
rad (Radical Function)	§7.1.2.88
sPre (Pre-Sub-Superscript Function)	§7.1.2.99
sSub (Subscript Function)	§7.1.2.101
sSubSup (Sub-Superscript Function)	§7.1.2.103
sSup (Superscript Function)	§7.1.2.105

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OMathArg">
   <sequence>
      <element name="argPr" type="CT_OMathArgPr" minOccurs="0"/>
      <group ref="EG_OMathElements" minOccurs="0" maxOccurs="unbounded"/>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.53 limLoc (n-ary Limit Location)

This element specifies the location of limits in n-ary operators. Limits can be either centered above and below the n-ary operator, or positioned just to the right of the operator, as in:

$$\sum_{i=0}^{n} x_n \sum_{i=0}^{n} x_n$$

When this element is omitted, the default location is undOvr. [*Example*: The XML representing this property in use is:

```
<m:naryPr>
  <m:chr  m:val="&#8721;"/>
  <m:limLoc m:val="subSup"/>
  <m:grow m:val="on"/>
</m:naryPr>
```

end example]

Parent Elements
naryPr (§7.1.2.72)

Attributes	Description
val (Value)	Specifies the default location of limits on an integral. Possible values are subSup and undOvr. The possible values for this attribute are defined by the ST_LimLoc simple type (§7.1.3.8).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LimLoc">
   <attribute name="val" type="ST_LimLoc" use="required"/>
</complexType>
```

7.1.2.54 limLow (Lower-Limit Function)

This element specifies the Lower-Limit function, consisting of text on the baseline and reduced-size text immediately below it. Examples of limLow include $\lim_{n \to \infty}$ and $\max_{0 \leq x \leq 1}$. [*Example*: The XML that represents $\lim_{n \to \infty}$ is:

```
<m:limLow>
  <m:e>
    <m:r>
      <m:rPr>
        <m:scr m:val="roman"/>
        <m:sty  m:val="p"/>
      </m:rPr>
      <m:t>lim</m:t>
    </m:r>
  </m:e>
  <m:lim>
    <m:r>
      <m:t>n&#8594;&#8734;</m:t>
    </m:r>
  </m:lim>
</m:limLow>
```

end example]

Parent Elements
deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); e (§7.1.2.32); fName (§7.1.2.37); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); sub (§7.1.2.112); sup (§7.1.2.114)

Child Elements	Subclause
e (Base (Argument))	§7.1.2.32
lim (Limit (Lower))	§7.1.2.52
limLowPr (Lower Limit Properties)	§7.1.2.55

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LimLow">
   <sequence>
      <element name="limLowPr" type="CT_LimLowPr" minOccurs="0"/>
      <element name="e" type="CT_OMathArg"/>
      <element name="lim" type="CT_OMathArg"/>
   </sequence>
</complexType>
```

7.1.2.55 limLowPr (Lower Limit Properties)

This element specifies control properties (ctrlPr) that can be stored on the Lower Limit (limLow).

Parent Elements
limLow (§7.1.2.54)

Child Elements	Subclause
ctrlPr (Control Properties)	§7.1.2.23

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LimLowPr">
   <sequence>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.56 limUpp (Upper-Limit Function)

This element specifies the Upper-Limit function, consisting of text on the baseline and reduced-size text immediately above it. [*Example*: Examples of limUpp include $\overbrace{x+x+x}^{k \text{ times}}$ and $\overset{\text{def}}{=}$.

The XML that specifies the $\text{limUpp}\overset{\text{def}}{=}$ is:

```
<m:limUpp>
  <m:e>
    <m:r>
      <m:t>=</m:t>
    </m:r>
  </m:e>
```

```
    <m:lim>
      <m:r>
        <m:rPr>
          <m:nor/>
        </m:rPr>
        <m:t>def</m:t>
      </m:r>
    </m:lim>
  </m:limUpp>
```

end example]

Parent Elements
deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); e (§7.1.2.32); fName (§7.1.2.37); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); sub (§7.1.2.112); sup (§7.1.2.114)

Child Elements	Subclause
e (Base (Argument))	§7.1.2.32
lim (Limit (Lower))	§7.1.2.52
limUppPr (Upper Limit Properties)	§7.1.2.57

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LimUpp">
   <sequence>
      <element name="limUppPr" type="CT_LimUppPr" minOccurs="0"/>
      <element name="e" type="CT_OMathArg"/>
      <element name="lim" type="CT_OMathArg"/>
   </sequence>
</complexType>
```

7.1.2.57 limUppPr (Upper Limit Properties)

This element specifies control properties (ctrlPr) that can be stored on the Upper Limit (limUpp).

Parent Elements
limUpp (§7.1.2.56)

Child Elements	Subclause
ctrlPr (Control Properties)	§7.1.2.23

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LimUppPr">
   <sequence>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.58 lit (Literal)

This element specifies that the characters in the run are literal; that is, they are to be interpreted literally and not take on any special mathematical meaning such as operators or characters that trigger conversion to a 2-dimensional format. [*Example*: In the following XML, the + operator is treated literally and does not receive proper binary spacing:

```
<m:r>a</m:r>
<m:r>
  <m:rPr>
    <m:lit/>
  </m:rPr>
  <m:t>+</m:t>
</m:r>
<m:r>
  <m:t>b</m:t>
</m:r>
```

end example]

Parent Elements
rPr (§7.1.2.91)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.59 lMargin (Left Margin)

This element specifies the left margin for math, in twips. Math margins are added to the paragraph settings for margins. [*Example*: The following XML demonstrates an lmargin setting of 1".

```
<m:mathPr>
  <m:mathFont m:val="Cambria Math"/>
  <m:brkBin m:val="before"/>
  <m:brkBinSub m:val="--"/>
  <m:smallFrac m:val="off"/>
  <m:dispDef/>
  <m:lMargin m:val="1440"/>
  <m:rMargin m:val="0"/>
  <m:defJc m:val="centerGroup"/>
  <m:wrapIndent m:val="1440"/>
  <m:intLim m:val="subSup"/>
  <m:naryLim m:val="undOvr"/>
</m:mathPr>
```

end example]

Parent Elements
mathPr (§7.1.2.62)

Attributes	Description
val (Value)	Specifies the value, in twips, of the parent element. The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§7.1.3.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TwipsMeasure">
   <attribute name="val" type="ST_TwipsMeasure" use="required"/>
</complexType>
```

7.1.2.60 m (Matrix Function)

This element specifies the Matrix function, consisting of one or more elements laid out in one or more rows and one or more columns. [*Example*: Examples of matrices are: $\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$ and $\begin{bmatrix} 1 & \\ & 1 \end{bmatrix}$. Below is a 2x2 matrix, in its proper form an in XML.

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

```
<m:m>
  <m:mPr>
    <m:mcs>
      <m:mc>
        <m:mcPr>
          <m:mcJc  m:val="center"/>
          <m:count m:val="2"/>
        </m:mcPr>
      </m:mc>
    </m:mcs>
  </m:mPr>
  <m:mr>
    <m:e>
      <m:r>
        <m:rPr>
          <m:scr  m:val="roman"/>
          <m:sty  m:val="p"/>
        </m:rPr>
        <m:t>1</m:t>
      </m:r >
    </m:e>
    <m:e>
      <m:r>
        <m:rPr>
          <m:scr  m:val="roman"/>
          <m:sty  m:val="p"/>
        </m:rPr>
        <m:t>2</m:t>
      </m:r >
    </m:e>
  </m:mr>
  <m:mr>
    <m:e>
      <m:r>
        <m:rPr>
          <m:scr  m:val="roman"/>
          <m:sty  m:val="p"/>
        </m:rPr>
        <m:t>3</m:t>
      </m:r >
    </m:e>
```

```
      <m:e>
        <m:r>
          <m:rPr>
            <m:scr  m:val="roman"/>
            <m:sty  m:val="p"/>
          </m:rPr>
          <m:t>4</m:t>
        </m:r >
      </m:e>
    </m:mr>
  </m:m>
```

end example]

Parent Elements
deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); e (§7.1.2.32); fName (§7.1.2.37); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); sub (§7.1.2.112); sup (§7.1.2.114)

Child Elements	Subclause
mPr (Matrix Properties)	§7.1.2.68
mr (Matrix Row)	§7.1.2.69

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_M">
   <sequence>
      <element name="mPr" type="CT_MPr" minOccurs="0"/>
      <element name="mr" type="CT_MR" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

7.1.2.61 mathFont (Math Font)

This element specifies the default math font to be used in the document. [*Example*: The XML containing this property is:

```
  <m:mathPr>
    <m:mathFont m:val="Cambria Math"/>
    <m:brkBin m:val="before"/>
    <m:brkBinSub m:val="--"/>
    <m:smallFrac m:val="off"/>
    <m:dispDef/>
```

```
    <m:lMargin m:val="1440"/>
    <m:rMargin m:val="0"/>
    <m:defJc m:val="centerGroup"/>
    <m:wrapIndent m:val="1440"/>
    <m:intLim m:val="subSup"/>
    <m:naryLim m:val="undOvr"/>
  </m:mathPr>
```

end example]

Parent Elements
mathPr (§7.1.2.62)

Attributes	Description
val (value)	Specifies the default math font to be used in the document. The possible values for this attribute are defined by the ST_String simple type (§7.1.3.13).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_String">
   <attribute name="val" type="ST_String"/>
</complexType>
```

7.1.2.62 mathPr (Math Properties)

This element specifies the document-level properties for all math in the document.

Parent Elements
settings (§2.15.1.78)

Child Elements	Subclause
brkBin (Break on Binary Operators)	§7.1.2.16
brkBinSub (Break on Binary Subtraction)	§7.1.2.17
defJc (Default Justification)	§7.1.2.25
dispDef (Use Display Math Defaults)	§7.1.2.30
interSp (Inter-Equation Spacing)	§7.1.2.48
intLim (Integral Limit Locations)	§7.1.2.49
intraSp (Intra-Equation Spacing)	§7.1.2.50
lMargin (Left Margin)	§7.1.2.59
mathFont (Math Font)	§7.1.2.61

Child Elements	Subclause
naryLim (n-ary Limit Location)	§7.1.2.71
postSp (Post-Equation Spacing)	§7.1.2.85
preSp (Pre-Equation Spacing)	§7.1.2.86
rMargin (Right Margin)	§7.1.2.90
smallFrac (Small Fraction)	§7.1.2.98
wrapIndent (Wrap Indent)	§7.1.2.120
wrapRight (Wrap Right)	§7.1.2.121

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MathPr">
   <sequence>
      <element name="mathFont" type="CT_String" minOccurs="0"/>
      <element name="brkBin" type="CT_BreakBin" minOccurs="0"/>
      <element name="brkBinSub" type="CT_BreakBinSub" minOccurs="0"/>
      <element name="smallFrac" type="CT_OnOff" minOccurs="0"/>
      <element name="dispDef" type="CT_OnOff" minOccurs="0"/>
      <element name="lMargin" type="CT_TwipsMeasure" minOccurs="0"/>
      <element name="rMargin" type="CT_TwipsMeasure" minOccurs="0"/>
      <element name="defJc" type="CT_OMathJc" minOccurs="0"/>
      <element name="preSp" type="CT_TwipsMeasure" minOccurs="0"/>
      <element name="postSp" type="CT_TwipsMeasure" minOccurs="0"/>
      <element name="interSp" type="CT_TwipsMeasure" minOccurs="0"/>
      <element name="intraSp" type="CT_TwipsMeasure" minOccurs="0"/>
      <choice minOccurs="0">
         <element name="wrapIndent" type="CT_TwipsMeasure"/>
         <element name="wrapRight" type="CT_OnOff"/>
      </choice>
      <element name="intLim" type="CT_LimLoc" minOccurs="0"/>
      <element name="naryLim" type="CT_LimLoc" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.63 maxDist (Maximum Distribution)

This element specifies Equation Array Maximum Distribution. When on, the equation array is spaced to the maximum width of the containing element (page, column, cell, etc.). When this element is omitted, Equation Array Maximum Distribution is off.

Parent Elements
eqArrPr (§7.1.2.35)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element.

Attributes	Description
	A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.64 mc (Matrix Column)

This element specifies a single column in a matrix m. [*Example*: An example of this element in use is:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

```
<m:m>
  <m:mPr>
    <m:mcs>
      <m:mc>
        <m:mcPr>
          <m:mcJc  m:val="center"/>
          <m:count m:val="2"/>
        </m:mcPr>
      </m:mc>
    </m:mcs>
  </m:mPr>
  <m:mr>
    <m:e>
      <m:r>
        <m:rPr>
          <m:scr  m:val="roman"/>
          <m:sty  m:val="p"/>
        </m:rPr>
        <m:t>1</m:t>
      </m:r >
    </m:e>
```

```
      <m:e>
        <m:r>
          <m:rPr>
            <m:scr  m:val="roman"/>
            <m:sty  m:val="p"/>
          </m:rPr>
          <m:t>2</m:t>
        </m:r >
      </m:e>
    </m:mr>
    <m:mr>
      <m:e>
        <m:r>
          <m:rPr>
            <m:scr  m:val="roman"/>
            <m:sty  m:val="p"/>
          </m:rPr>
          <m:t>3</m:t>
        </m:r >
      </m:e>
      <m:e>
        <m:r>
          <m:rPr>
            <m:scr  m:val="roman"/>
            <m:sty  m:val="p"/>
          </m:rPr>
          <m:t>4</m:t>
        </m:r >
      </m:e>
    </m:mr>
  </m:m>
```

end example]

Parent Elements
mcs (§7.1.2.67)

Child Elements	Subclause
mcPr (Matrix Column Properties)	§7.1.2.66

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MC">
   <sequence>
      <element name="mcPr" type="CT_MCPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.65 mcJc (Matrix Column Justification)

This element specifies the justification of a matrix column (or group of matrix columns) mc. When this element is omitted, the column is centered. The matrix below has three columns. The leftmost column is left-justified, the rightmost column is right-justified, and the center column is centered: $\begin{pmatrix} 1 & 1 & 1 \\ 23 & 23 & 23 \\ 456 & 456 & 456 \end{pmatrix}$

[*Example*: A simple example of this property in use is a 2x2 matrix with both columns centered:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

```
<m:m>
  <m:mPr>
    <m:mcs>
      <m:mc>
        <m:mcPr>
          <m:mcJc  m:val="center"/>
          <m:count m:val="2"/>
        </m:mcPr>
      </m:mc>
    </m:mcs>
  </m:mPr>
  <m:mr>
    <m:e>
      <m:r>
        <m:rPr>
          <m:scr  m:val="roman"/>
          <m:sty  m:val="p"/>
        </m:rPr>
        <m:t>1</m:t>
      </m:r >
    </m:e>
```

```
      <m:e>
        <m:r>
          <m:rPr>
            <m:scr  m:val="roman"/>
            <m:sty  m:val="p"/>
          </m:rPr>
          <m:t>2</m:t>
        </m:r >
      </m:e>
    </m:mr>
    <m:mr>
      <m:e>
        <m:r>
          <m:rPr>
            <m:scr  m:val="roman"/>
            <m:sty  m:val="p"/>
          </m:rPr>
          <m:t>3</m:t>
        </m:r >
      </m:e>
      <m:e>
        <m:r>
          <m:rPr>
            <m:scr  m:val="roman"/>
            <m:sty  m:val="p"/>
          </m:rPr>
          <m:t>4</m:t>
        </m:r >
      </m:e>
    </m:mr>
  </m:m>
```

end example]

Parent Elements
mcPr (§7.1.2.66)

Attributes	Description

val (Value)	Specifies the horizontal alignment of the parent element. Possible values are `left`, `right`, and `center`. [*Example*:
	`<m:mcPr>`
	`<m:mcJc  m:val="center"/>`
	`<m:count m:val="2"/>`

Attributes	Description
	</m:mcPr> The possible values for this attribute are defined by the ST_XAlign simple type (§7.1.3.18).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_XAlign">
   <attribute name="val" type="ST_XAlign" use="required"/>
</complexType>
```

7.1.2.66 mcPr (Matrix Column Properties)

This element specifies the properties of the matrix column mn, including the number of columns and the type of justification. [*Example*: As an extreme example, the following matrix has two columns that are left justified (count is 2) and three columns that are right justified (count is 3). $\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 23 & 23 & 23 & 23 & 23 \\ 456 & 456 & 456 & 456 & 456 \end{pmatrix}$ *end example*]

Parent Elements
mc (§7.1.2.64)

Child Elements	Subclause
count (Matrix Column Count)	§7.1.2.21
mcJc (Matrix Column Justification)	§7.1.2.65

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MCPr">
   <sequence>
      <element name="count" type="CT_Integer255" minOccurs="0"/>
      <element name="mcJc" type="CT_XAlign" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.67 mcs (Matrix Columns)

This element specifies the collection of columns of the matrix m. [*Example*: An example of this element in use is:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

```
<m:m>
  <m:mPr>
    <m:mcs>
      <m:mc>
        <m:mcPr>
          <m:mcJc  m:val="center"/>
          <m:count m:val="2"/>
        </m:mcPr>
      </m:mc>
    </m:mcs>
  </m:mPr>
  <m:mr>
    <m:e>
      <m:r>
        <m:rPr>
          <m:scr  m:val="roman"/>
          <m:sty  m:val="p"/>
        </m:rPr>
        <m:t>1</m:t>
      </m:r >
    </m:e>
    <m:e>
      <m:r>
        <m:rPr>
          <m:scr  m:val="roman"/>
          <m:sty  m:val="p"/>
        </m:rPr>
        <m:t>2</m:t>
      </m:r >
    </m:e>
  </m:mr>
  <m:mr>
    <m:e>
      <m:r>
        <m:rPr>
          <m:scr  m:val="roman"/>
          <m:sty  m:val="p"/>
        </m:rPr>
        <m:t>3</m:t>
      </m:r >
    </m:e>
```

```
      <m:e>
        <m:r>
          <m:rPr>
            <m:scr  m:val="roman"/>
            <m:sty  m:val="p"/>
          </m:rPr>
          <m:t>4</m:t>
        </m:r >
      </m:e>
    </m:mr>
  </m:m>
```

end example]

Parent Elements
mPr (§7.1.2.68)

Child Elements	Subclause
mc (Matrix Column)	§7.1.2.64

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MCS">
   <sequence>
      <element name="mc" type="CT_MC" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

7.1.2.68 mPr (Matrix Properties)

This element specifies properties of the matrix m, including the justification of the matrix and the layout of elements within the matrix.

Parent Elements
m (§7.1.2.60)

Child Elements	Subclause
baseJc (Matrix Base Justification)	§7.1.2.9
cGp (Matrix Column Gap)	§7.1.2.18
cGpRule (Matrix Column Gap Rule)	§7.1.2.19
cSp (Matrix Column Spacing)	§7.1.2.22
ctrlPr (Control Properties)	§7.1.2.23

Child Elements	Subclause
mcs (Matrix Columns)	§7.1.2.67
plcHide (Hide Placeholders (Matrix))	§7.1.2.83
rSp (Row Spacing (Equation Array))	§7.1.2.92
rSpRule (Row Spacing Rule)	§7.1.2.93

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MPr">
   <sequence>
      <element name="baseJc" type="CT_YAlign" minOccurs="0"/>
      <element name="plcHide" type="CT_OnOff" minOccurs="0"/>
      <element name="rSpRule" type="CT_SpacingRule" minOccurs="0"/>
      <element name="cGpRule" type="CT_SpacingRule" minOccurs="0"/>
      <element name="rSp" type="CT_UnSignedInteger" minOccurs="0"/>
      <element name="cSp" type="CT_UnSignedInteger" minOccurs="0"/>
      <element name="cGp" type="CT_UnSignedInteger" minOccurs="0"/>
      <element name="mcs" type="CT_MCS" minOccurs="0"/>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.69 mr (Matrix Row)

This element specifies a single row of the matrix m. [*Example*: An example of this element in use is the following example, a 2x2 matrix. There are two rows; the first contains the elements 1 and 2; the second contains 3 and 4.

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

```
<m:m>
  <m:mPr>
    <m:mcs>
      <m:mc>
        <m:mcPr>
          <m:mcJc  m:val="center"/>
          <m:count m:val="2"/>
        </m:mcPr>
      </m:mc>
    </m:mcs>
  </m:mPr>
```

```
    <m:mr>
      <m:e>
        <m:r>
          <m:rPr>
            <m:scr  m:val="roman"/>
            <m:sty  m:val="p"/>
          </m:rPr>
          <m:t>1</m:t>
        </m:r >
      </m:e>
      <m:e>
        <m:r>
          <m:rPr>
            <m:scr  m:val="roman"/>
            <m:sty  m:val="p"/>
          </m:rPr>
          <m:t>2</m:t>
        </m:r >
      </m:e>
    </m:mr>
    <m:mr>
      <m:e>
        <m:r>
          <m:rPr>
            <m:scr  m:val="roman"/>
            <m:sty  m:val="p"/>
          </m:rPr>
          <m:t>3</m:t>
        </m:r >
      </m:e>
      <m:e>
        <m:r>
          <m:rPr>
            <m:scr  m:val="roman"/>
            <m:sty  m:val="p"/>
          </m:rPr>
          <m:t>4</m:t>
        </m:r >
      </m:e>
    </m:mr>
  </m:m>
```

end example]

Parent Elements
m (§7.1.2.60)

Child Elements	Subclause
e (Base (Argument))	§7.1.2.32

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_MR">
   <sequence>
      <element name="e" type="CT_OMathArg" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

7.1.2.70 nary (n-ary Operator Function)

This element specifies an n-ary object, consisting of an n-ary object, a base (or operand), and optional upper and lower limits. Examples of n-ary objects are: $\int_0^1 x\,dx$, $\sum_k \binom{n}{k}$, $\prod_{k=1}^{n} A_k$, and $\cup_{=1}^{m}(X_n \cap Y_n)$. [*Example*: The example below demonstrates an n-ary object in its proper form and XML representation:

$$\int_0^1 x\,dx$$

```
<m:nary>
  <m:naryPr>
    <m:chr m:val="&#8747;"/>
  </m:naryPr>
  <m:sub>
      <m:r>
        <m:rPr>
          <m:scr m:val="roman"/>
          <m:sty m:val="p"/>
        </m:rPr>
        <m:t>0</m:t>
      </m:r>
  </m:sub>
  <m:sup>
      <m:r>
        <m:rPr>
          <m:scr  m:val="roman"/>
          <m:sty  m:val="p"/>
        </m:rPr>
        <m:t>1</m:t>
      </m:r>
  </m:sup>
```

```
    <m:e>
      <m:r>
        <m:t>x</m:t>
      </m:r>
      <m:box>
        <m:boxPr>
          <m:diff  m:val="on"/>
        </m:boxPr>
        <m:e>
          <m:r>
            <m:t>dx</m:t>
          </m:r>
        </m:e>
      </m:box>
    </m:e>
  </m:nary>
```

end example]

Parent Elements
deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); e (§7.1.2.32); fName (§7.1.2.37); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); sub (§7.1.2.112); sup (§7.1.2.114)

Child Elements	Subclause
e (Base (Argument))	§7.1.2.32
naryPr (n-ary Properties)	§7.1.2.72
sub (Subscript (Pre-Sub-Superscript))	§7.1.2.112
sup (Superscript (Superscript function))	§7.1.2.114

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Nary">
   <sequence>
      <element name="naryPr" type="CT_NaryPr" minOccurs="0"/>
      <element name="sub" type="CT_OMathArg"/>
      <element name="sup" type="CT_OMathArg"/>
      <element name="e" type="CT_OMathArg"/>
   </sequence>
</complexType>
```

7.1.2.71 naryLim (n-ary Limit Location)

This element specifies the document setting for the default placement of n-ary limits other than integrals (since integrals are most often written as subSup and other n-ary operators are most often written as undOvr), when

converted from a linear format to a two-dimensional output. Limits can be either centered above and below the n-ary operator, or positioned just to the right of the operator, as in:

$$\sum_{i=0}^{n} x_i \quad \sum_{i=0}^{n} x_i$$

When this integral object is written linearly, as $\sum_(i = 0)^n$, the placement of limits is ambiguous. naryLim specifies this positioning. When this element is omitted, the default placement of n-ary limits is undOvr (that is, the location of lower and upper limits). [*Example*: An example XML of this element in use is:

```
<m:mathPr>
  <m:mathFont m:val="Cambria Math"/>
  <m:brkBin m:val="before"/>
  <m:brkBinSub m:val="--"/>
  <m:smallFrac m:val="off"/>
  <m:dispDef/>
  <m:lMargin m:val="0"/>
  <m:rMargin m:val="0"/>
  <m:defJc m:val="centerGroup"/>
  <m:wrapIndent m:val="1440"/>
  <m:intLim m:val="subSup"/>
  <m:naryLim m:val="undOvr"/>
</m:mathPr>
```

end example]

Parent Elements
mathPr (§7.1.2.62)

Attributes	Description
val (Value)	Specifies the default location of limits on an integral. Possible values are subSup and undOvr. The possible values for this attribute are defined by the ST_LimLoc simple type (§7.1.3.8).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_LimLoc">
   <attribute name="val" type="ST_LimLoc" use="required"/>
</complexType>
```

7.1.2.72 naryPr (n-ary Properties)

This element specifies the properties of the n-ary object, including the type of n-ary operator that is used, the shape and height of the operator, the location of limits, and whether limits are shown or hidden.

Parent Elements
nary (§7.1.2.70)

Child Elements	Subclause
chr (Accent Character)	§7.1.2.20
ctrlPr (Control Properties)	§7.1.2.23
grow (n-ary Grow)	§7.1.2.43
limLoc (n-ary Limit Location)	§7.1.2.53
subHide (Hide Subscript (n-ary))	§7.1.2.113
supHide (Hide Superscript (n-ary))	§7.1.2.115

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NaryPr">
   <sequence>
      <element name="chr" type="CT_Char" minOccurs="0"/>
      <element name="limLoc" type="CT_LimLoc" minOccurs="0"/>
      <element name="grow" type="CT_OnOff" minOccurs="0"/>
      <element name="subHide" type="CT_OnOff" minOccurs="0"/>
      <element name="supHide" type="CT_OnOff" minOccurs="0"/>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.73 noBreak (No Break)

This property specifies the "unbreakable" property on the Box function box. When on, no line breaks can occur within the box. This can be important for operator emulators that consist of more than one binary operator. When this element is not specified, breaks can occur inside box. [*Example*: Sample XML containing this element is below. In this box, breaks are allowed.

```
<m:boxPr>
  <m:noBreak m:val="off"/>
</m:boxPr>
```

end example]

Parent Elements
boxPr (§7.1.2.14)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element.

Attributes	Description
	A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.74 nor (Normal Text)

This element specifies that the run is normal text, i.e., math italics and math spacing are not applied. In a normal text run, no characters will trigger reformatting of a linear expression into a two-dimensional expression. [*Example*: The example below illustrates three runs of normal text, along with the XML representation of the equation:

$$\text{rate} = \frac{\text{distance}}{\text{time}}$$

```
<m:r>
  <m:rPr>
    <m:nor/>
  </m:rPr>
  <m:t>rate</m:t>
</m:r>
<m:r>
  <m:t>=</m:t>
</m:r>
<m:f>
  <m:num>
    <m:r>
      <m:rPr>
        <m:nor/>
      </m:rPr>
      <m:t>distance</m:t>
    </m:r>
  </m:num>
```

```
    <m:den>
      <m:r>
        <m:rPr>
          <m:nor/>
        </m:rPr>
        <m:t>time</m:t>
      </m:r>
    </m:den>
  </m:f>
```

end example]

Parent Elements
rPr (§7.1.2.91)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.75 num (Numerator)

This element specifies the numerator of the Fraction function f. [*Example*: The a in $^{a}/_{b}$:

```
  <m:f>
    <m:fPr>
      <m:type  m:val="skw"/>
    </m:fPr>
    <m:num>
      <m:r>
        <m:t>a</m:t>
      </m:r>
    </m:num>
```

```
      <m:den>
        <m:r>
          <m:t>b</m:t>
        </m:r>
      </m:den>
    </m:f>
```

end example]

Parent Elements
f (§7.1.2.36)

Child Elements	Subclause
acc (Accent)	§7.1.2.1
argPr (Argument Properties)	§7.1.2.5
bar (Bar)	§7.1.2.7
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
borderBox (Border-Box Function)	§7.1.2.11
box (Box Function)	§7.1.2.13
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
ctrlPr (Control Properties)	§7.1.2.23
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
d (Delimiter Function)	§7.1.2.24
del (Deleted Run Content)	§2.13.5.12
eqArr (Equation-Array Function)	§7.1.2.34
f (Fraction Function)	§7.1.2.36
func (Function Apply Function)	§7.1.2.39
groupChr (Group-Character Function)	§7.1.2.41

Child Elements	Subclause
ins (Inserted Run Content)	§2.13.5.20
limLow (Lower-Limit Function)	§7.1.2.54
limUpp (Upper-Limit Function)	§7.1.2.56
m (Matrix Function)	§7.1.2.60
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
nary (n-ary Operator Function)	§7.1.2.70
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
phant (Phantom Function)	§7.1.2.81
proofErr (Proofing Error Anchor)	§2.13.8.1
r (Run)	§7.1.2.87
rad (Radical Function)	§7.1.2.88
sPre (Pre-Sub-Superscript Function)	§7.1.2.99
sSub (Subscript Function)	§7.1.2.101
sSubSup (Sub-Superscript Function)	§7.1.2.103
sSup (Superscript Function)	§7.1.2.105

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OMathArg">
   <sequence>
      <element name="argPr" type="CT_OMathArgPr" minOccurs="0"/>
      <group ref="EG_OMathElements" minOccurs="0" maxOccurs="unbounded"/>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.76 objDist (Object Distribution)

This element specifies Equation Array Object Distribution. When on, the contents of the equation array are spaced to the maximum width of the equation array object. When this element is omitted, the equation array does not receive object distribution.

Parent Elements
eqArrPr (§7.1.2.35)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.77 oMath (Office Math)

This element specifies an equation or mathematical expression. All equations are surrounded by oMath tags.

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32); endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); oMathPara (§7.1.2.78); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Child Elements	Subclause
acc (Accent)	§7.1.2.1
bar (Bar)	§7.1.2.7
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
borderBox (Border-Box Function)	§7.1.2.11
box (Box Function)	§7.1.2.13
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4

Child Elements	Subclause
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
d (Delimiter Function)	§7.1.2.24
del (Deleted Run Content)	§2.13.5.12
eqArr (Equation-Array Function)	§7.1.2.34
f (Fraction Function)	§7.1.2.36
func (Function Apply Function)	§7.1.2.39
groupChr (Group-Character Function)	§7.1.2.41
ins (Inserted Run Content)	§2.13.5.20
limLow (Lower-Limit Function)	§7.1.2.54
limUpp (Upper-Limit Function)	§7.1.2.56
m (Matrix Function)	§7.1.2.60
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
nary (n-ary Operator Function)	§7.1.2.70
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
phant (Phantom Function)	§7.1.2.81
proofErr (Proofing Error Anchor)	§2.13.8.1
r (Run)	§7.1.2.87
rad (Radical Function)	§7.1.2.88
sPre (Pre-Sub-Superscript Function)	§7.1.2.99

Child Elements	Subclause
sSub (Subscript Function)	§7.1.2.101
sSubSup (Sub-Superscript Function)	§7.1.2.103
sSup (Superscript Function)	§7.1.2.105

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OMath">
    <sequence>
        <group ref="EG_OMathElements" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
</complexType>
```

7.1.2.78 oMathPara (Math Paragraph)

This element specifies a math paragraph, one or more display equations within a single paragraph.

Parent Elements
body (§2.2.2); comment (§2.13.4.2); customXml (§2.5.1.3); customXml (§2.5.1.4); customXml (§2.5.1.5); customXml (§2.5.1.6); deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); docPartBody (§2.12.6); e (§7.1.2.32); endnote (§2.11.2); fldSimple (§2.16.21); fName (§7.1.2.37); footnote (§2.11.10); ftr (§2.10.3); hdr (§2.10.4); hyperlink (§2.16.24); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); p (§2.3.1.22); rt (§2.3.3.23); rubyBase (§2.3.3.26); sdtContent (§2.5.2.32); sdtContent (§2.5.2.33); sdtContent (§2.5.2.34); sdtContent (§2.5.2.35); smartTag (§2.5.1.9); sub (§7.1.2.112); sup (§7.1.2.114); tbl (§2.4.36); tc (§2.4.62); tr (§2.4.75); txbxContent (§2.17.1.1)

Child Elements	Subclause
oMath (Office Math)	§7.1.2.77
oMathParaPr (Office Math Paragraph Properties)	§7.1.2.79

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OMathPara">
    <sequence>
        <element name="oMathParaPr" type="CT_OMathParaPr" minOccurs="0" maxOccurs="1"/>
        <element name="oMath" type="CT_OMath" minOccurs="1" maxOccurs="unbounded">
            <unique name="uniqueContentAnchorIdsInsideMath">
                <selector xpath="m:annotation/m:content"/>
                <field xpath="@id"/>
            </unique>
            <unique name="uniqueContextAnchorIdsInsideMath">
                <selector xpath="m:annotation/m:context"/>
                <field xpath="@id"/>
            </unique>
        </element>
    </sequence>
</complexType>
```

7.1.2.79 oMathParaPr (Office Math Paragraph Properties)

This property specifies properties of the math paragraph oMathPara, including justification jc.

Parent Elements
oMathPara (§7.1.2.78)

Child Elements	Subclause
jc (Justification)	§7.1.2.51

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OMathParaPr">
   <sequence>
      <element name="jc" type="CT_OMathJc" minOccurs="0" maxOccurs="1"/>
   </sequence>
</complexType>
```

7.1.2.80 opEmu (Operator Emulator)

This element specifies the Operator Emulator property on box. When on, the box and its contents behave as a single operator and inherit the properties of an operator. This means, for example, that the character can serve as a point for a line break and can be aligned to other operators. Operator Emulators are often used when one or more glyphs combine to form an operator, such as ==. The following equation uses an Operator Emulator:

$$a == b$$

[*Example*: Its XML representation is as follows:

```
<m:r>
  <m:t>a</m:t>
</m:r>
<m:box>
  <m:boxPr>
    <m:opEmu    m:val="on"/>
    <m:aln/>
  </m:boxPr>
  <m:e>
    <m:r>
      <m:t>==</m:t>
    </m:r>
  </m:e>
</m:box>
<m:r>
  <m:t>b</m:t>
</m:r>
```

end example]

Parent Elements
boxPr (§7.1.2.14)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.81 phant (Phantom Function)

This element specifies the phantom function. phant has two primary uses: adding the spacing of the phantom base e without displaying that base; and suppressing part of the glyph from spacing considerations.

Parent Elements
deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); e (§7.1.2.32); fName (§7.1.2.37); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); sub (§7.1.2.112); sup (§7.1.2.114)

Child Elements	Subclause
e (Base (Argument))	§7.1.2.32
phantPr (Phantom Properties)	§7.1.2.82

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Phant">
   <sequence>
      <element name="phantPr" type="CT_PhantPr" minOccurs="0"/>
      <element name="e" type="CT_OMathArg"/>
   </sequence>
</complexType>
```

7.1.2.82 phantPr (Phantom Properties)

This element specifies properties of the Phantom function, including whether the phantom is hidden or visible, and the amount of space that is taken into account when laying out text and objects around phantoms.

Parent Elements
phant (§7.1.2.81)

Child Elements	Subclause
ctrlPr (Control Properties)	§7.1.2.23
show (Phantom Show)	§7.1.2.96
transp (Transparent (Phantom))	§7.1.2.117
zeroAsc (Phantom Zero Ascent)	§7.1.2.122
zeroDesc (Phantom Zero Descent)	§7.1.2.123
zeroWid (Phantom Zero Width)	§7.1.2.124

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PhantPr">
   <sequence>
      <element name="show" type="CT_OnOff" minOccurs="0"/>
      <element name="zeroWid" type="CT_OnOff" minOccurs="0"/>
      <element name="zeroAsc" type="CT_OnOff" minOccurs="0"/>
      <element name="zeroDesc" type="CT_OnOff" minOccurs="0"/>
      <element name="transp" type="CT_OnOff" minOccurs="0"/>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.83 plcHide (Hide Placeholders (Matrix))

This element specifies the Hide Placeholders property on a matrix m. When this property is on, placeholders do not appear in the matrix. If this element is omitted, placeholders do appear such that the locations where text can be inserted are made visible. [*Example*: The following two examples of matrices show the hidden and visible states of placeholders:

$$\begin{pmatrix} 1 & & \\ & 1 & \\ & & 1 \end{pmatrix}\begin{pmatrix} 1 & & \\ & 1 & \\ & & 1 \end{pmatrix}$$

In the matrix described by the XML below, placeholders will be hidden:

```
<m:mPr>
  <m:plcHide  m:val="on"/>
  <m:mcs>
    <m:mc>
      <m:mcPr>
        <m:mcJc m:val="center"/>
        <m:count  m:val="3"/>
      </m:mcPr>
    </m:mc>
  </m:mcs>
</m:mPr>
```

end example]

Parent Elements
mPr (§7.1.2.68)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.84 pos (Position (Bar))

This element specifies the position of the bar in the bar object; the default is 'top,' signifying the mathematical overbar. [*Example*: The XML representation for the mathematical overbar is:

```
<m:bar>
  <m:barPr>
    <m:pos   m:val="top"/>
  </m:barPr>
  <m:e>
    <m:r>
      <m:t>a</m:t>
    </m:r>
  </m:e>
</m:bar>
```

end example]

Parent Elements
barPr (§7.1.2.8); groupChrPr (§7.1.2.42)

Attributes	Description

val (Value)	Specifies the position of the parent element. Possible values are `top` and `bot`.
	
	[*Example*:
	`<m:barPr>`
	`  <m:pos   m:val="top"/>`
	`</m:barPr>`
	
	The possible values for this attribute are defined by the ST_TopBot simple type (§7.1.3.15).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TopBot">
   <attribute name="val" type="ST_TopBot" use="required"/>
</complexType>
```

7.1.2.85 postSp (Post-Equation Spacing)

This element specifies the spacing after math paragraph, in twips

Parent Elements
mathPr (§7.1.2.62)

Attributes	Description
val (Value)	Specifies the value, in twips, of the parent element. The possible values for this attribute are defined by the ST_TwipsMeasure simple type

Attributes	Description
	(§7.1.3.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TwipsMeasure">
   <attribute name="val" type="ST_TwipsMeasure" use="required"/>
</complexType>
```

7.1.2.86 preSp (Pre-Equation Spacing)

This element specifies the spacing before a math paragraph, in twips

Parent Elements
mathPr (§7.1.2.62)

Attributes	Description
val (Value)	Specifies the value, in twips, of the parent element. The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§7.1.3.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TwipsMeasure">
   <attribute name="val" type="ST_TwipsMeasure" use="required"/>
</complexType>
```

7.1.2.87 r (Run)

This element specifies a run of math text.

Parent Elements
deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); e (§7.1.2.32); fName (§7.1.2.37); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); sub (§7.1.2.112); sup (§7.1.2.114)

Child Elements	Subclause
annotationRef (Comment Information Block)	§2.13.4.1
br (Break)	§2.3.3.1
commentReference (Comment Content Reference Mark)	§2.13.4.5
continuationSeparator (Continuation Separator Mark)	§2.11.1
cr (Carriage Return)	§2.3.3.4
dayLong (Date Block - Long Day Format)	§2.3.3.5

Child Elements	Subclause
dayShort (Date Block - Short Day Format)	§2.3.3.6
delInstrText (Deleted Field Code)	§2.16.13
delText (Deleted Text)	§2.3.3.7
drawing (DrawingML Object)	§2.3.3.9
endnoteRef (Endnote Reference Mark)	§2.11.6
endnoteReference (Endnote Reference)	§2.11.7
fldChar (Complex Field Character)	§2.16.18
footnoteRef (Footnote Reference Mark)	§2.11.13
footnoteReference (Footnote Reference)	§2.11.14
instrText (Field Code)	§2.16.25
lastRenderedPageBreak (Position of Last Calculated Page Break)	§2.3.3.13
monthLong (Date Block - Long Month Format)	§2.3.3.15
monthShort (Date Block - Short Month Format)	§2.3.3.16
noBreakHyphen (Non Breaking Hyphen Character)	§2.3.3.18
object (Inline Embedded Object)	§2.3.3.19
pgNum (Page Number Block)	§2.3.3.20
pict (VML Object)	§2.3.3.21
ptab (Absolute Position Tab Character)	§2.3.3.22
rPr (Run Properties)	§7.1.2.91
rPr (Run Properties)	§2.3.2.25
ruby (Phonetic Guide)	§2.3.3.24
separator (Footnote/Endnote Separator Mark)	§2.11.23
softHyphen (Optional Hyphen Character)	§2.3.3.28
sym (Symbol Character)	§2.3.3.29
t (Text)	§7.1.2.116
t (Text)	§2.3.3.30
tab (Tab Character)	§2.3.3.31
yearLong (Date Block - Long Year Format)	§2.3.3.32
yearShort (Date Block - Short Year Format)	§2.3.3.33

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_R">
   <sequence>
      <element name="rPr" type="CT_RPR" minOccurs="0"/>
      <group ref="w:EG_RPr" minOccurs="0"/>
      <choice minOccurs="0" maxOccurs="unbounded">
         <group ref="w:EG_RunInnerContent"/>
         <element name="t" type="CT_Text" minOccurs="0"/>
      </choice>
   </sequence>
</complexType>
```

7.1.2.88 rad (Radical Function)

This element specifies the radical function, consisting of a radical, a base e, and an optional degree deg.

[*Example*: Examples of rad are $\sqrt[3]{x}$ (XML shown below) and $\sqrt{x}$.

```
<m:rad>
  <m:deg>
     <m:r>
        <m:rPr>
          <m:scr   m:val="roman"/>
          <m:sty   m:val="p"/>
        </m:rPr>
        <m:t>3</m:t>
     </m:r>
  </m:deg>
  <m:e>
    <m:r>
      <m:t>x</m:t>
    </m:r>
  </m:e>
</m:rad>
```

end example]

Parent Elements
deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); e (§7.1.2.32); fName (§7.1.2.37); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); sub (§7.1.2.112); sup (§7.1.2.114)

Child Elements	Subclause
deg (Degree)	§7.1.2.26
e (Base (Argument))	§7.1.2.32
radPr (Radical Properties)	§7.1.2.89

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Rad">
   <sequence>
      <element name="radPr" type="CT_RadPr" minOccurs="0"/>
      <element name="deg" type="CT_OMathArg"/>
      <element name="e" type="CT_OMathArg"/>
   </sequence>
</complexType>
```

7.1.2.89 radPr (Radical Properties)

This element specifies properties of the Radical function rad, including the hidden or shown state of the degree deg.

Parent Elements
rad (§7.1.2.88)

Child Elements	Subclause
ctrlPr (Control Properties)	§7.1.2.23
degHide (Hide Degree)	§7.1.2.27

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RadPr">
   <sequence>
      <element name="degHide" type="CT_OnOff" minOccurs="0"/>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.90 rMargin (Right Margin)

This element specifies the right margin for math, in twips. Math margins are added to the paragraph settings for margins. [*Example*: The following XML demonstrates an rmargin setting of 1".

```
<m:mathPr>
  <m:mathFont m:val="Cambria Math"/>
  <m:brkBin m:val="before"/>
  <m:brkBinSub m:val="--"/>
  <m:smallFrac m:val="off"/>
  <m:dispDef/>
```

```
    <m:lMargin m:val="0"/>
    <m:rMargin m:val="1440"/>
    <m:defJc m:val="centerGroup"/>
    <m:wrapIndent m:val="1440"/>
    <m:intLim m:val="subSup"/>
    <m:naryLim m:val="undOvr"/>
  </m:mathPr>
```

end example]

Parent Elements
mathPr (§7.1.2.62)

Attributes	Description
val (Value)	Specifies the value, in twips, of the parent element. The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§7.1.3.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TwipsMeasure">
   <attribute name="val" type="ST_TwipsMeasure" use="required"/>
</complexType>
```

7.1.2.91 rPr (Run Properties)

This element specifies the properties of the math run r.

Parent Elements
r (§7.1.2.87)

Child Elements	Subclause
aln (Alignment)	§7.1.2.3
brk (Break)	§7.1.2.15
lit (Literal)	§7.1.2.58
nor (Normal Text)	§7.1.2.74
scr (Script)	§7.1.2.94
sty (style)	§7.1.2.111

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_RPR">
   <sequence>
      <element name="lit" minOccurs="0" type="CT_OnOff"/>
      <choice>
         <element name="nor" minOccurs="0" type="CT_OnOff"/>
         <sequence>
            <group ref="EG_ScriptStyle"/>
         </sequence>
      </choice>
      <element name="brk" minOccurs="0" type="CT_ManualBreak"/>
      <element name="aln" minOccurs="0" type="CT_OnOff"/>
   </sequence>
</complexType>
```

7.1.2.92 rSp (Row Spacing (Equation Array))

This element specifies spacing between rows of an equation array eqArr; it is used only when rSpRule is set to 3 (exactly; in which case the unit of measure is points) or 4 (Multiple; in which case the unit of measure is lines). If this element is omitted, single line spacing is used in the equation array, and no additional spacing is used in the layout of rows.

[*Example*: Below are three examples of the same equation array, with single line spacing, 1.5 line spacing, and double line spacing:

$$\begin{pmatrix} x- & y+ & z= & 10 \\ 3x+ & y+2z= & 34 \\ -5x+2y- & z= & -14 \end{pmatrix} \begin{pmatrix} x- & y+ & z= & 10 \\ 3x+ & y+2z= & 34 \\ -5x+2y- & z= & -14 \end{pmatrix} \begin{pmatrix} x- & y+ & z= & 10 \\ 3x+ & y+2z= & 34 \\ -5x+2y- & z= & -14 \end{pmatrix}$$

The following eqArr $\begin{matrix} a=b+c \\ d+e=f \end{matrix}$ has rSp of 1.6:

```
<m:eqArr>
  <m:eqArrPr>
    <m:rSpRule  m:val="4"/>
    <m:rSp m:val="3"/>
  </m:eqArrPr>
  <m:e>
    <m:r>
      <m:t>a=b+c</m:t>
    </m:r>
  </m:e>
```

```
    <m:e>
      <m:r>
        <m:t>d+e=f</m:t>
      </m:r>
    </m:e>
  </m:eqArr>
```

end example]

Parent Elements
eqArrPr (§7.1.2.35); mPr (§7.1.2.68)

Attributes	Description
val (Value)	Specifies the amount of space between the parent element. The manner in which this value is determined depends on the setting of the rule of the parent element. If the rule is set to 3 (or "Exactly"), then the unit is interpreted as points. If the rule is set to 4 (or "Multiple"), then the unit is interpreted as lines. The possible values for this attribute are defined by the ST_UnSignedInteger simple type (§7.1.3.17).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_UnSignedInteger">
   <attribute name="val" type="ST_UnSignedInteger" use="required"/>
</complexType>
```

7.1.2.93 rSpRule (Row Spacing Rule)

This element specifies the type of vertical spacing between columns in a matrix. The following table demonstrates possible values of rSpRule along with their definitions and examples.

Value	Line spacing between rows	Example (non-normative)
0	Single line gap	1 2 3 4
1	1.5 line gap	1 2 3 4
2	2 line gap	1 2 3 4
3	Exactly (rely on value of rGp, measured in points)	1 2 3 4
4	Multiple (rely on value of rGp, measured in lines)	1 2 3 4

Parent Elements
eqArrPr (§7.1.2.35); mPr (§7.1.2.68)

Attributes	Description
val (Value)	Specifies the type of spacing between rows and/or columns. Possible values are 0, 1, 2, 3, or 4, whose definitions are contained in the following table: **Value** / **Column/Row Gap** / **Example** 0 / Single spacing gap / 1 2 1 / 1.5 spacing gap / 1 2 2 / 2 spacing gap / 1 2 3 / Exactly (for columns, rely on value of cGp, measured in points) (for rows, rely on value of rSp, measured in points) / 1 2 4 / Multiple (for columns, rely on value of cGp, measured in lines) (for rows, rely on value of rSp, measured in lines) / 1 2 The possible values for this attribute are defined by the ST_SpacingRule simple type (§7.1.3.12).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SpacingRule">
   <attribute name="val" type="ST_SpacingRule" use="required"/>
</complexType>
```

7.1.2.94 scr (Script)

This element describes the script applied to the characters in the run. The XML includes the ASCII value of the character along with the script of the character. The application maps the ASCII value and script type to the appropriate Unicode range. [*Example*: Example: 𝔞𝔞

```
<m:r>
  <m:rPr>
    <m:scr m:val="fraktur"/>
    <m:sty m:val="p"/>
  </m:rPr>
  <m:t>a</m:t>
</m:r>
<m:r>
  <m:rPr>
    <m:scr m:val=“double-struck"/>
    <m:sty m:val="p"/>
  </m:rPr>
  <m:t>a</m:t>
</m:r>
```

end example]

Parent Elements
rPr (§7.1.2.91)

Attributes	Description
val (Value)	Specifies the script type of the parent element. Possible values are: `double-struck`, `fraktur`, `monospace`, `roman`, `sans-serif`, and `script`. The possible values for this attribute are defined by the ST_Script simple type (§7.1.3.10).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Script">
   <attribute name="val" type="ST_Script"/>
</complexType>
```

7.1.2.95 sepChr (Delimiter Separator Character)

This element specifies the character that separates base arguments e in the delimiter object d. If this element is omitted, the default sepChr is '|'. [*Example*: Examples of d, each with a different sepChr, are: $(a_1|a_2)(a_1:a_2)(a_1;a_2)$. The following example describes a separator character if :.

```
<m:dPr>
  <m:sepChr val=&#0058/>
</m:dPr>
```

end example]

Parent Elements

Parent Elements
dPr (§7.1.2.31)

Attributes	Description

val (value)	Specifies the character used by the parent element. When it is omitted, the parent uses its assigned default. [*Example*: delimiter object {a}:
	`<m:dPr>`
	`<m:begChr m:val="{"/>`
	`<m:endChr m:val="}"/>`
	`</m:dPr>`
	
	The possible values for this attribute are defined by the ST_Char simple type (§7.1.3.3).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Char">
   <attribute name="val" type="ST_Char" use="required"/>
</complexType>
```

7.1.2.96 show (Phantom Show)

This element specifies the show property of the phantom phant. When off, the phant base e is hidden. If this element is omitted, the base e is shown. [*Example*: In the following example, there is a phantom of the fraction a/b in the second radical such that only the height is preserved. The fraction does not show.

$$\sqrt{\frac{a}{b}} = \sqrt{x}$$

```
<m:phantPr>
  <m:show  m:val="off"/>
  <m:zeroDesc m:val="on"/>
</m:phantPr>
```

end example]

Parent Elements
phantPr (§7.1.2.82)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of `on` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of `off` specifies that the property shall be explicitly turned off. This is implied

Attributes	Description
	when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.97 shp (Shape (Delimiters))

This element specifies the shape of delimiters in the delimiter object d. Delimiters can be centered around the entire height of their contents, or their height can be altered to exactly match their contents' height. When this element is omitted, delimiters are 'centered'. [*Example*: In the example below, delimiters will be matched to the height of their contents:

```
<m:dPr>
  <m:shp  m:val="match"/>
</m:dPr>
```

end example]

Parent Elements
dPr (§7.1.2.31)

Attributes	Description
val (Value)	Specifies the shape of the parent element. Possible values are `match` and `centered`. The possible values for this attribute are defined by the ST_Shp simple type (§7.1.3.11).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Shp">
   <attribute name="val" type="ST_Shp" use="required"/>
</complexType>
```

7.1.2.98 smallFrac (Small Fraction)

This element specifies a reduced fraction size display math, such that the numerator and denominator are written in script size instead of at the size of regular text. [*Example*: The XML containing this element in use is:

```
<m:mathPr>
  <m:mathFont m:val="Cambria Math"/>
  <m:brkBin m:val="before"/>
  <m:brkBinSub m:val="--"/>
  <m:smallFrac m:val="off"/>
  <m:dispDef/>
  <m:lMargin m:val="0"/>
  <m:rMargin m:val="0"/>
  <m:defJc m:val="centerGroup"/>
  <m:wrapIndent m:val="1440"/>
  <m:intLim m:val="subSup"/>
  <m:naryLim m:val="undOvr"/>
</m:mathPr>
```

end example]

Parent Elements
mathPr (§7.1.2.62)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.99 sPre (Pre-Sub-Superscript Function)

This element specifies the Pre-Sub-Superscript function, which consists of a base e and a subscript and superscript placed to the left of the base, as in ${}_{1}^{2}A$. [*Example*: The XML that specifies this function is:

```
<m:sPre>
  <m:sub>
    <m:r>
      <m:t>1</m:t>
    </m:r>
  </m:sub>
  <m:sup>
    <m:r>
      <m:t>2</m:t>
    </m:r>
  </m:sup>
  <m:e>
    <m:r>
      <m:t>A</m:t>
    </m:r>
  </m:e>
</m:sPre>
```

end example]

Parent Elements
deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); e (§7.1.2.32); fName (§7.1.2.37); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); sub (§7.1.2.112); sup (§7.1.2.114)

Child Elements	Subclause
e (Base (Argument))	§7.1.2.32
sPrePr (Pre-Sub-Superscript Properties)	§7.1.2.100
sub (Subscript (Pre-Sub-Superscript))	§7.1.2.112
sup (Superscript (Superscript function))	§7.1.2.114

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SPre">
   <sequence>
      <element name="sPrePr" type="CT_SPrePr" minOccurs="0"/>
      <element name="sub" type="CT_OMathArg"/>
      <element name="sup" type="CT_OMathArg"/>
      <element name="e" type="CT_OMathArg"/>
   </sequence>
</complexType>
```

7.1.2.100 sPrePr (Pre-Sub-Superscript Properties)

This element specifies properties such as ctrlPr that can be stored on the Pre-Sub-Superscript object sPre.

Parent Elements
sPre (§7.1.2.99)

Child Elements	Subclause
ctrlPr (Control Properties)	§7.1.2.23

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SPrePr">
   <sequence>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.101 sSub (Subscript Function)

This element specifies the subscript function sSub, which consists of a base e and a reduced-size scr placed below and to the right, as in x_n. [*Example*: The XML that specifies this function is:

```
<m:sSub>
  <m:e>
    <m:r>
      <m:t>x</m:t>
    </r>
  </m:e>
  <m:sub>
    <m:r>
      <m:t>n</m:t>
    </r>
  </m:sub>
</m:sSub>
```

end example]

Parent Elements
deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); e (§7.1.2.32); fName (§7.1.2.37); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); sub (§7.1.2.112); sup (§7.1.2.114)

Child Elements	Subclause
e (Base (Argument))	§7.1.2.32
sSubPr (Subscript Properties)	§7.1.2.102
sub (Subscript (Pre-Sub-Superscript))	§7.1.2.112

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SSub">
   <sequence>
      <element name="sSubPr" type="CT_SSubPr" minOccurs="0"/>
      <element name="e" type="CT_OMathArg"/>
      <element name="sub" type="CT_OMathArg"/>
   </sequence>
</complexType>
```

7.1.2.102 sSubPr (Subscript Properties)

This element specifies properties such as ctrlPr that can be stored on the Subscript function sSub.

Parent Elements
sSub (§7.1.2.101)

Child Elements	Subclause
ctrlPr (Control Properties)	§7.1.2.23

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SSubPr">
   <sequence>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.103 sSubSup (Sub-Superscript Function)

This element specifies the sub-superscript function, which consists of a base e, a reduced-size scr placed below and to the right, and a reduced-size scr placed above and to the right, as in x_m^n.

[*Example*: The XML that specifies this function is:

```
<m:sSubSup>
  <m:e>
    <m:r>
      <m:t>x</m:t>
    </r>
  </m:e>
  <m:sub>
    <m:r>
      <m:t>m</m:t>
    </r>
  </m:sub>
```

```
    <m:sup>
      <m:r>
        <m:t>n</m:t>
      </r>
    </m:sup>
  </m:sSubSup>
```

end example]

Parent Elements
deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); e (§7.1.2.32); fName (§7.1.2.37); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); sub (§7.1.2.112); sup (§7.1.2.114)

Child Elements	Subclause
e (Base (Argument))	§7.1.2.32
sSubSupPr (Sub-Superscript Properties)	§7.1.2.104
sub (Subscript (Pre-Sub-Superscript))	§7.1.2.112
sup (Superscript (Superscript function))	§7.1.2.114

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SSubSup">
   <sequence>
      <element name="sSubSupPr" type="CT_SSubSupPr" minOccurs="0"/>
      <element name="e" type="CT_OMathArg"/>
      <element name="sub" type="CT_OMathArg"/>
      <element name="sup" type="CT_OMathArg"/>
   </sequence>
</complexType>
```

7.1.2.104 sSubSupPr (Sub-Superscript Properties)

This element specifies properties of the Sub-Superscript function, including the alignment of scripts.

Parent Elements
sSubSup (§7.1.2.103)

Child Elements	Subclause
alnScr (Align Scripts)	§7.1.2.4
ctrlPr (Control Properties)	§7.1.2.23

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SSubSupPr">
   <sequence>
      <element name="alnScr" type="CT_OnOff" minOccurs="0"/>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.105 sSup (Superscript Function)

This element specifies the subscript function sSub, which consists of a base e and a reduced-size scr placed below and to the right, as in x^n. [*Example*: The XML that specifies this function is:

```
<m:sSup>
  <m:e>
    <m:r>
      <m:t>x</m:t>
    </r>
  </m:e>
  <m:sup>
    <m:r>
      <m:t>n</m:t>
    </r>
  </m:sup>
</m:sSup>
```

end example]

Parent Elements
deg (§7.1.2.26); del (§2.13.5.12); den (§7.1.2.28); e (§7.1.2.32); fName (§7.1.2.37); ins (§2.13.5.20); lim (§7.1.2.52); moveFrom (§2.13.5.21); moveTo (§2.13.5.26); num (§7.1.2.75); oMath (§7.1.2.77); sub (§7.1.2.112); sup (§7.1.2.114)

Child Elements	Subclause
e (Base (Argument))	§7.1.2.32
sSupPr (Superscript Properties)	§7.1.2.106
sup (Superscript (Superscript function))	§7.1.2.114

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SSup">
   <sequence>
      <element name="sSupPr" type="CT_SSupPr" minOccurs="0"/>
      <element name="e" type="CT_OMathArg"/>
      <element name="sup" type="CT_OMathArg"/>
   </sequence>
</complexType>
```

7.1.2.106 sSupPr (Superscript Properties)

This element specifies properties such as ctrlPr that can be stored on the Superscript function sSup.

Parent Elements
sSup (§7.1.2.105)

Child Elements	Subclause
ctrlPr (Control Properties)	§7.1.2.23

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SSupPr">
   <sequence>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.107 strikeBLTR (Border Box Strikethrough Bottom-Left to Top-Right)

This element specifies the hidden or shown state of a strikethrough diagonal line from the bottom-left corner to the top-right corner of borderBox. When this element is omitted, the strikethrough is not drawn. When on, a strikethrough is drawn, as in ~~abc~~. [*Example*:

```
<m:borderBox>
  <m:borderBoxPr>
    <m:hideTop  m:val="on"/>
    <m:hideBot  m:val="on"/>
    <m:hideLeft  m:val="on"/>
    <m:hideRight  m:val="on"/>
    <m:strikeBLTR m:val="on"/>
  </m:borderBoxPr>
  <m:e>
    <m:r>
      <m:t>abc</m:t>
    </m:r>
  </m:e>
</m:borderBox>
```

end example]

Parent Elements
borderBoxPr (§7.1.2.12)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.108 strikeH (Border Box Strikethrough Horizontal)

This element specifies the hidden or shown state of a strikethrough horizontal line in borderBox. When this element is omitted, the strikethrough is not drawn. When on, a horizontal strikethrough is drawn, as in ~~*abc*~~. [*Example*:

```
<m:borderBox>
  <m:borderBoxPr>
    <m:hideTop  m:val="on"/>
    <m:hideBot  m:val="on"/>
    <m:hideLeft  m:val="on"/>
    <m:hideRight  m:val="on"/>
    <m:strikeH m:val="on"/>
  </m:borderBoxPr>
  <m:e>
    <m:r>
      <m:t>abc</m:t>
    </m:r>
  </m:e>
</m:borderBox>
```

end example]

Parent Elements

Parent Elements
borderBoxPr (§7.1.2.12)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.109 strikeTLBR (Border Box Strikethrough Top-Left to Bottom-Right)

This element specifies the hidden or shown state of a strikethrough diagonal line from the top-left corner to the bottom-right corner of borderBox. When this element is omitted, the strikethrough is not drawn. When on, a strikethrough is drawn, as in ~~abc~~. [*Example*:

```
<m:borderBox>
  <m:borderBoxPr>
    <m:hideTop  m:val="on"/>
    <m:hideBot  m:val="on"/>
    <m:hideLeft  m:val="on"/>
    <m:hideRight  m:val="on"/>
    <m:strikeTLBR m:val="on"/>
  </m:borderBoxPr>
  <m:e>
    <m:r>
      <m:t>abc</m:t>
    </m:r>
  </m:e>
</m:borderBox>
```

end example]

Parent Elements
borderBoxPr (§7.1.2.12)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.110 strikeV (Border Box Strikethrough Vertical)

This element specifies the hidden or shown state of a strikethrough vertical line in borderBox. When this element is omitted, the strikethrough is not drawn. When on, a strikethrough is drawn, as in $\boxed{a}$. [*Example*:

```
<m:borderBox>
  <m:borderBoxPr>
    <m:strikeV m:val="on"/>
  </m:borderBoxPr>
  <m:e>
    <m:r>
      <m:t>abc</m:t>
    </m:r>
  </m:e>
</m:borderBox>
```

end example]

Parent Elements
borderBoxPr (§7.1.2.12)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present.

Attributes	Description
	A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.111 sty (style)

This element describes the script applied to the characters in the run. The XML includes the ASCII value of the character along with the style of the character. The application maps the ASCII value and style to the appropriate Unicode range. [*Example*: Example: $\mathbf{a}\boldsymbol{bc}$

```
<m:oMath>
  <m:r>
    <m:rPr>
      <m:scr  m:val="roman"/>
      <m:sty m:val="b"/>
    </m:rPr>
    <m:t>abc</m:t>
  </m:r>
</m:oMath>
```

end example]

Parent Elements
rPr (§7.1.2.91)

Attributes	Description
val (Value)	Specifies the style of the parent element. Possible values are b (bold), i (italic), bi (bold-italic), and p (plain). The possible values for this attribute are defined by the ST_Style simple type (§7.1.3.14).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Style">
   <attribute name="val" type="ST_Style"/>
</complexType>
```

7.1.2.112 sub (Subscript (Pre-Sub-Superscript))

This element specifies the subscript of the Pre-Sub-Superscript function sPre. [*Example*: For example, the sub in the object ${}_{1}^{2}A$ is 1. An example of this element in use is:

```
<m:sPre>
   <m:sub>
    <m:r>
      <m:t>1</m:t>
    </m:r>
  </m:sub>
  <m:sup>
    <m:r>
      <m:t>2</m:t>
    </m:r>
  </m:sup>
  <m:e>
    <m:r>
      <m:t>A</m:t>
    </m:r>
  </m:e>
</m:sPre>
```

end example]

Parent Elements
nary (§7.1.2.70); sPre (§7.1.2.99); sSub (§7.1.2.101); sSubSup (§7.1.2.103)

Child Elements	Subclause
acc (Accent)	§7.1.2.1
argPr (Argument Properties)	§7.1.2.5
bar (Bar)	§7.1.2.7
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
borderBox (Border-Box Function)	§7.1.2.11
box (Box Function)	§7.1.2.13
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
ctrlPr (Control Properties)	§7.1.2.23
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4

Child Elements	Subclause
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
d (Delimiter Function)	§7.1.2.24
del (Deleted Run Content)	§2.13.5.12
eqArr (Equation-Array Function)	§7.1.2.34
f (Fraction Function)	§7.1.2.36
func (Function Apply Function)	§7.1.2.39
groupChr (Group-Character Function)	§7.1.2.41
ins (Inserted Run Content)	§2.13.5.20
limLow (Lower-Limit Function)	§7.1.2.54
limUpp (Upper-Limit Function)	§7.1.2.56
m (Matrix Function)	§7.1.2.60
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
nary (n-ary Operator Function)	§7.1.2.70
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
phant (Phantom Function)	§7.1.2.81
proofErr (Proofing Error Anchor)	§2.13.8.1
r (Run)	§7.1.2.87
rad (Radical Function)	§7.1.2.88
sPre (Pre-Sub-Superscript Function)	§7.1.2.99
sSub (Subscript Function)	§7.1.2.101

Child Elements	Subclause
sSubSup (Sub-Superscript Function)	§7.1.2.103
sSup (Superscript Function)	§7.1.2.105

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OMathArg">
   <sequence>
      <element name="argPr" type="CT_OMathArgPr" minOccurs="0"/>
      <group ref="EG_OMathElements" minOccurs="0" maxOccurs="unbounded"/>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.113 subHide (Hide Subscript (n-ary))

This element specifies the n-ary Hide Subscript property. When on, the lower limit does not appear, as in $\int^{\infty} \frac{x}{x+1} dx$. If this element is omitted, the lower limit appears. [*Example*: An example of this element in use is:

```
<m:naryPr>
  <m:chr  m:val="&#8747;"/>
  <m:subHide m:val="on"/>
</m:naryPr>
```

end example]

Parent Elements
naryPr (§7.1.2.72)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.114 sup (Superscript (Superscript function))

This element specifies the superscript of the superscript function sSup. For example, the sup in the superscript object x^n is n. [*Example*: An example of this element in use is:

```
<m:sSup>
  <m:e>
    <m:r>
      <m:t>x</m:t>
    </m:r>
  </m:e>
  <m:sup>
    <m:r>
      <m:t>n</m:t>
    </m:r>
  </m:sup>
</m:sSup>
```

end example]

Parent Elements
nary (§7.1.2.70); sPre (§7.1.2.99); sSubSup (§7.1.2.103); sSup (§7.1.2.105)

Child Elements	Subclause
acc (Accent)	§7.1.2.1
argPr (Argument Properties)	§7.1.2.5
bar (Bar)	§7.1.2.7
bookmarkEnd (Bookmark End)	§2.13.6.1
bookmarkStart (Bookmark Start)	§2.13.6.2
borderBox (Border-Box Function)	§7.1.2.11
box (Box Function)	§7.1.2.13
commentRangeEnd (Comment Anchor Range End)	§2.13.4.3
commentRangeStart (Comment Anchor Range Start)	§2.13.4.4
ctrlPr (Control Properties)	§7.1.2.23
customXmlDelRangeEnd (Custom XML Markup Deletion End)	§2.13.5.4
customXmlDelRangeStart (Custom XML Markup Deletion Start)	§2.13.5.5
customXmlInsRangeEnd (Custom XML Markup Insertion End)	§2.13.5.6
customXmlInsRangeStart (Custom XML Markup Insertion Start)	§2.13.5.7
customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)	§2.13.5.8
customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)	§2.13.5.9

Child Elements	Subclause
customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)	§2.13.5.10
customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)	§2.13.5.11
d (Delimiter Function)	§7.1.2.24
del (Deleted Run Content)	§2.13.5.12
eqArr (Equation-Array Function)	§7.1.2.34
f (Fraction Function)	§7.1.2.36
func (Function Apply Function)	§7.1.2.39
groupChr (Group-Character Function)	§7.1.2.41
ins (Inserted Run Content)	§2.13.5.20
limLow (Lower-Limit Function)	§7.1.2.54
limUpp (Upper-Limit Function)	§7.1.2.56
m (Matrix Function)	§7.1.2.60
moveFrom (Move Source Run Content)	§2.13.5.21
moveFromRangeEnd (Move Source Location Container - End)	§2.13.5.23
moveFromRangeStart (Move Source Location Container - Start)	§2.13.5.24
moveTo (Move Destination Run Content)	§2.13.5.26
moveToRangeEnd (Move Destination Location Container - End)	§2.13.5.27
moveToRangeStart (Move Destination Location Container - Start)	§2.13.5.28
nary (n-ary Operator Function)	§7.1.2.70
oMath (Office Math)	§7.1.2.77
oMathPara (Math Paragraph)	§7.1.2.78
permEnd (Range Permission End)	§2.13.7.1
permStart (Range Permission Start)	§2.13.7.2
phant (Phantom Function)	§7.1.2.81
proofErr (Proofing Error Anchor)	§2.13.8.1
r (Run)	§7.1.2.87
rad (Radical Function)	§7.1.2.88
sPre (Pre-Sub-Superscript Function)	§7.1.2.99
sSub (Subscript Function)	§7.1.2.101
sSubSup (Sub-Superscript Function)	§7.1.2.103
sSup (Superscript Function)	§7.1.2.105

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OMathArg">
   <sequence>
      <element name="argPr" type="CT_OMathArgPr" minOccurs="0"/>
      <group ref="EG_OMathElements" minOccurs="0" maxOccurs="unbounded"/>
      <element name="ctrlPr" type="CT_CtrlPr" minOccurs="0"/>
   </sequence>
</complexType>
```

7.1.2.115 supHide (Hide Superscript (n-ary))

This element specifies the n-ary Hide Superscript property. When on, the upper limit does not appear, as in $\int_0 \frac{x}{x+1} dx$. If this element is omitted, the lower limit appears. [*Example*: An example of this element in use is:

```
<m:naryPr>
  <m:chr  m:val="&#8747;"/>
  <m:supHide m:val="on"/>
</m:naryPr>
```

end example]

Parent Elements
naryPr (§7.1.2.72)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.116 t (Text)

This element specifies the text in a math run r.

Parent Elements
r (§7.1.2.87)

Attributes	Description

space (Content Contains Significant Whitespace)
 	
Namespace: http://www.w3.org/XML/1998/namespace
	Specifies how white space should be handled for the contents of this element using the W3C space preservation rules. [*Example*: Consider the following run contained within a WordprocessingML document:
	
	`<w:r>`
	`  <w:t>significant whitespace  </w:t>`
	`</w:r>`
	
	Although there are three spaces on each side of the text content in the run, that whitespace has not been specifically marked as significant, therefore it is removed when this run is added to the document. *end example*]
	
	The possible values for this attribute are defined by the type in the namespace.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Text">
   <simpleContent>
      <extension base="ST_String">
         <attribute ref="xml:space" use="optional"/>
      </extension>
   </simpleContent>
</complexType>
```

7.1.2.117 transp (Transparent (Phantom))

This element specifies that the phantom is transparent for spacing. This means that if the contents of the phantom are belonging to a special spacing class (such as binary operators, relational operators, differentials, etc.), the contents of that phantom are taken into consideration when laying out text. If transparency is off, then the contents of the phantom are ignored during layout. When this element is omitted, transparency is 'off'. In the following example, transparency is off on the phantom around the differential term. $\int xdx$. The spacing is incorrect. In the following integral, the only difference is that transparency is on: $\int x\,dx$. Now the spacing is correct.

[*Example*: An example of this element in XML is:

```
<m:phantPr>
  <m:zeroAsc  m:val="on"/>
  <m:zeroDesc  m:val="on"/>
  <m:transp  m:val="on"/>
</m:phantPr>
```

end example]

Parent Elements
phantPr (§7.1.2.82)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of `on` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of `off` specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.118 type (Fraction type)

This element specifies the type of fraction f; the default is 'bar'. Fractions types are:

Stacked Fraction: $\frac{a}{b}$

Skewed Fraction: a/b

Linear Fraction: a/b

Stack Object (No-Bar Fraction): $\binom{n}{k}$

Parent Elements
fPr (§7.1.2.38)

Attributes	Description
val (Value)	Specifies the type of fraction. Possible values are `bar` (Bar Fraction), `lin` (Linear Fraction), `noBar` (No-Bar Fraction (Stack)), and `skw` (Skewed). The possible values for this attribute are defined by the ST_FType simple type (§7.1.3.4).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_FType">
   <attribute name="val" type="ST_FType" use="required"/>
</complexType>
```

7.1.2.119 vertJc (Vertical Justification)

This element, combined with pos of groupChrPr, specifies the vertical layout of the groupChr object. Where pos specifies the position of the grouping character, vertJc specifies the alignment of the object with respect to the baseline. For example, when the group character is above the object, vertJc of top signifies that the top of the object falls on the baseline; when vertJc is set to bot, the bottom of the object is on the baseline. The table below demonstrates the four possible combinations of groupChr layout:

Pos	vertJc	Layout
top	top	$a \underset{bcd}{\longleftarrow} e$
top	bot	$a \overbrace{bcd} e$
bot	top	$a \underbrace{bcd} e$
bot	bot	$a \overset{yields}{\longrightarrow} b$

Parent Elements
groupChrPr (§7.1.2.42)

Attributes	Description

val (Value)	Specifies the position of the parent element. Possible values are top and bot.
	
	[*Example*:
	`<m:barPr>`
	`  <m:pos  m:val="top"/>`
	`</m:barPr>`
	
	The possible values for this attribute are defined by the ST_TopBot simple type (§7.1.3.15).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TopBot">
   <attribute name="val" type="ST_TopBot" use="required"/>
</complexType>
```

7.1.2.120 wrapIndent (Wrap Indent)

This element specifies the indent of the wrapped line of an equation. The line or lines of a wrapped equation after the line break can either be indented by a specified amount from the left margin, or right aligned. The default indent is 1".

[*Example*: The XML below demonstrates wrapIndent in use:

```
<m:mathPr>
  <m:mathFont m:val="Cambria Math"/>
  <m:brkBin m:val="before"/>
  <m:brkBinSub m:val="--"/>
  <m:smallFrac m:val="off"/>
  <m:dispDef/>
  <m:lMargin m:val="0"/>
  <m:rMargin m:val="0"/>
  <m:defJc m:val="centerGroup"/>
  <m:wrapIndent m:val="1440"/>
  <m:intLim m:val="subSup"/>
  <m:naryLim m:val="undOvr"/>
</m:mathPr>
```

end example]

Parent Elements
mathPr (§7.1.2.62)

Attributes	Description
val (Value)	Specifies the value, in twips, of the parent element. The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§7.1.3.16).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_TwipsMeasure">
   <attribute name="val" type="ST_TwipsMeasure" use="required"/>
</complexType>
```

7.1.2.121 wrapRight (Wrap Right)

This element specifies the right justification of the wrapped line of an equation. The line or lines of a wrapped equation after the line break can either be indented by a specified amount from the left margin, or right aligned. If this element is present, the continuation is right aligned. [*Example*: An example of this element in use is:

```
<m:mathPr>
  <m:mathFont m:val="Cambria Math"/>
  <m:brkBin m:val="before"/>
  <m:brkBinSub m:val="--"/>
  <m:smallFrac m:val="off"/>
  <m:dispDef/>
```

```
    <m:lMargin m:val="0"/>
    <m:rMargin m:val="0"/>
    <m:defJc m:val="centerGroup"/>
    <m:wrapRight/>
    <m:intLim m:val="subSup"/>
    <m:naryLim m:val="undOvr"/>
  </m:mathPr>
```

end example]

Parent Elements
mathPr (§7.1.2.62)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
    <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.122 zeroAsc (Phantom Zero Ascent)

This element specifies that the phantom has zero ascent. The ascent of the contents of the phantom is not taken into account during layout. When this property is omitted, the phantom does have ascent (zero ascent is off). In the following example, the differential term is contained in a phantom that zero ascent. As a result, spacing is reduced between the tip of the "d" and the radical bar: $\sqrt{xdx}$. [*Example*:

```
  <m:phantPr>
    <m:zeroAsc  m:val="on"/>
  </m:phantPr>
```

end example]

Parent Elements
phantPr (§7.1.2.82)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.123 zeroDesc (Phantom Zero Descent)

This element specifies that the phantom has zero descent. The descent of the contents of the phantom is not taken into account during layout. When this property is omitted, the phantom does have descent (zero descent is off). [*Example*: In the following two examples, only the second has zero descent around the "y". Note that the radical is smaller than in the first case. $\sqrt{y}\sqrt{y}$

```
<m:phantPr>
  <m:zeroDesc  m:val="on"/>
</m:phantPr>
```

end example]

Parent Elements
phantPr (§7.1.2.82)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.2.124 zeroWid (Phantom Zero Width)

This element specifies that the phantom has zero width. The width of the contents of the phantom is not taken into account during layout. When this property is omitted, the phantom does have width (zero width is off). [*Example*: In the folowing example, the second radical contains a phantom of the fraction a/b. The phantom has zero width, such that only the height grows to accommodate the hidden fraction: $\sqrt{\frac{a}{b}} = \sqrt{x}$

```
<m:phantPr>
  <m:show  m:val="off"/>
  <m:zeroDesc  m:val="on"/>
</m:phantPr>
```

end example]

Parent Elements
phantPr (§7.1.2.82)

Attributes	Description
val (value)	Specifies a binary value for the property defined by the parent XML element. A value of on specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present. A value of off specifies that the property shall be explicitly turned off. This is implied when the parent element is not present. The possible values for this attribute are defined by the ST_OnOff simple type (§7.1.3.9).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_OnOff">
   <attribute name="val" type="ST_OnOff"/>
</complexType>
```

7.1.3 Simple Types

This is the complete list of simple types in the http://schemas.openxmlformats.org/officeDocument/2006/math namespace.

7.1.3.1 ST_BreakBin (Break Binary Operators)

This defines how to represent binary operators with respect to a line-wrapping break. The line can wrap before the operator or after the operator; alternately, the operator can appear both at the end of the first line and the beginning of the second.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
after (After)	When line-wrapping breaks occur on binary operators, the binary operator appears after the break (at the start of the next line).
before (Before)	When line-wrapping breaks occur on binary operators, the binary operator appears before the break (at the end of the first line).
repeat (Repeat)	When line-wrapping breaks occur on binary operators, the binary operator appears on both sides of the break (at the end of the first line and the start of the next line).

Referenced By
brkBin@val (§7.1.2.16)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_BreakBin">
   <restriction base="xsd:string">
      <enumeration value="before"/>
      <enumeration value="after"/>
      <enumeration value="repeat"/>
   </restriction>
</simpleType>
```

7.1.3.2 ST_BreakBinSub (Break on Binary Subtraction)

This simple type specifies how to represent subtraction on both sides of a line-wrapping break, when the Break Binary Operators option is set to repeat. The first character represents the sign at the end of the line with the break; the second represents the sign at the start of the wrapped line. Options are --, -+, and +-.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
-- (Minus Minus)	Repetition of subtraction sign after a line-wrapping

Enumeration Value	Description
	break is minus on the first and second lines.
-+ (Minus Plus)	Repetition of subtraction sign after a line-wrapping break is minus on the first line and plus on the second line.
+- (Plus Minus)	Repetition of subtraction sign after a line-wrapping break is plus on the first line and minus on the second line.

Referenced By
brkBinSub@val (§7.1.2.17)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_BreakBinSub">
   <restriction base="xsd:string">
      <enumeration value="--"/>
      <enumeration value="-+"/>
      <enumeration value="+-"/>
   </restriction>
</simpleType>
```

7.1.3.3 ST_Char (Character)

This Simple Type specifies the single character used by the parent element.

[*Example*: In the following example, $\{a\}$ uses { and } as its enclosing characters, instead of the default (and).

```
<m:dPr>
  <m:begChr  m:val="{"/>
  <m:endChr m:val="}"/>
</m:dPr>
```

end example]

This simple type's contents are a restriction of the XML Schema string datatype.

This simple type also specifies the following restrictions:

- This simple type's contents have a maximum length of 1 characters.

Referenced By
begChr@val (§7.1.2.10); chr@val (§7.1.2.20); endChr@val (§7.1.2.33); sepChr@val (§7.1.2.95)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Char">
   <restriction base="xsd:string">
      <maxLength value="1"/>
   </restriction>
</simpleType>
```

7.1.3.4 ST_FType (Fraction Type)

Fractions can be of type bar (horizontal fraction bar), skewed ("skw" - diagonal fraction bar with kerned and vertically adjusted numerator and denominator), linear ("lin" - diagonal fraction bar, takes up exactly one line of space), and the "stack" object ("noBar").

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
bar (Bar Fraction)	Fraction with a horizontal fraction bar.
lin (Linear Fraction)	Fraction with slanted fraction bar, that takes up no additional vertical space.
noBar (No-Bar Fraction (Stack))	Stack object, which looks like a fraction with no fraction bar.
skw (Skewed)	Fraction with diagonal fraction bar.

Referenced By
type@val (§7.1.2.118)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_FType">
   <restriction base="xsd:string">
      <enumeration value="bar"/>
      <enumeration value="skw"/>
      <enumeration value="lin"/>
      <enumeration value="noBar"/>
   </restriction>
</simpleType>
```

7.1.3.5 ST_Integer2 (Integer value (-2 to 2))

This simple type contains a value from (-2,+2) which specifies the size of the argument. The effects of each value are described by the referencing element.

This simple type's contents are a restriction of the XML Schema integer datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to -2.
- This simple type has a maximum value of less than or equal to 2.

Referenced By
argSz@val (§7.1.2.6)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Integer2">
   <restriction base="xsd:integer">
      <minInclusive value="-2"/>
      <maxInclusive value="2"/>
   </restriction>
</simpleType>
```

7.1.3.6 ST_Integer255 (Integer value (1 to 255))

This simple type specifies an integer value. The semantics of each value are discussed by the referencing element.

This simple type's contents are a restriction of the XML Schema integer datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 1.
- This simple type has a maximum value of less than or equal to 255.

Referenced By
brk@alnAt (§7.1.2.15); count@val (§7.1.2.21)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Integer255">
   <restriction base="xsd:integer">
      <minInclusive value="1"/>
      <maxInclusive value="255"/>
   </restriction>
</simpleType>
```

7.1.3.7 ST_Jc (Justification)

This Simple Type specifies the justification of Math Paragraphs. Justification of the Math Paragraph can be Left, Right, Centered, or Centered as Group.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
center (Center (Equation))	Centers each equation individually with respect to

Enumeration Value	Description
	margins.
centerGroup (Centered as Group (Equations))	Justifies equations with respect to each other, and centers the group of equations (the Math Paragraph) with respect to the page.
left (Left Justification)	Left justification of Math Paragraph
right (Right)	Right Justification of Math Paragraph

Referenced By
defJc@val (§7.1.2.25); jc@val (§7.1.2.51)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Jc">
   <restriction base="xsd:string">
      <enumeration value="left"/>
      <enumeration value="right"/>
      <enumeration value="center"/>
      <enumeration value="centerGroup"/>
   </restriction>
</simpleType>
```

7.1.3.8 ST_LimLoc (Limit Location)

Limits can be in one of two positions: Under-Over (undOvr - above and below the base), and Subscript-Superscript (subSup - positioned to the side of the base, in the position of subscripts and superscripts).

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
subSup (Subscript-Superscript location)	Limits placed to the side of the base, as opposed to directly over and under.
undOvr (Under-Over location)	Limits placed to the directly above and below the base, as opposed to on the side.

Referenced By
intLim@val (§7.1.2.49); limLoc@val (§7.1.2.53); naryLim@val (§7.1.2.71)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_LimLoc">
   <restriction base="xsd:string">
      <enumeration value="undOvr"/>
      <enumeration value="subSup"/>
   </restriction>
</simpleType>
```

7.1.3.9 ST_OnOff (On Off)

The boolean value of either on or off.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
off (Off)	Off
on (On)	On

Referenced By
aln@val (§7.1.2.3); alnScr@val (§7.1.2.4); degHide@val (§7.1.2.27); diff@val (§7.1.2.29); dispDef@val (§7.1.2.30); grow@val (§7.1.2.43); hideBot@val (§7.1.2.44); hideLeft@val (§7.1.2.45); hideRight@val (§7.1.2.46); hideTop@val (§7.1.2.47); lit@val (§7.1.2.58); maxDist@val (§7.1.2.63); noBreak@val (§7.1.2.73); nor@val (§7.1.2.74); objDist@val (§7.1.2.76); opEmu@val (§7.1.2.80); plcHide@val (§7.1.2.83); show@val (§7.1.2.96); smallFrac@val (§7.1.2.98); strikeBLTR@val (§7.1.2.107); strikeH@val (§7.1.2.108); strikeTLBR@val (§7.1.2.109); strikeV@val (§7.1.2.110); subHide@val (§7.1.2.113); supHide@val (§7.1.2.115); transp@val (§7.1.2.117); wrapRight@val (§7.1.2.121); zeroAsc@val (§7.1.2.122); zeroDesc@val (§7.1.2.123); zeroWid@val (§7.1.2.124)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_OnOff">
   <restriction base="xsd:string">
      <enumeration value="on"/>
      <enumeration value="off"/>
   </restriction>
</simpleType>
```

7.1.3.10 ST_Script (Script)

Script can be of type Roman, Script, Fraktur, Double-Struck, Sans-Serif, or Monospace.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description

Enumeration Value	Description
double-struck (double-struck)	Double-Struck Script Type
fraktur (Fraktur)	Fraktur Script Type
monospace (Monospace)	Monospace Script Type
roman (Roman)	Roman Script Type
sans-serif (Sans-Serif)	Sans-Serif Script Type
script (Script)	Script Type

Referenced By
scr@val (§7.1.2.94)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Script">
   <restriction base="xsd:string">
      <enumeration value="roman"/>
      <enumeration value="script"/>
      <enumeration value="fraktur"/>
      <enumeration value="double-struck"/>
      <enumeration value="sans-serif"/>
      <enumeration value="monospace"/>
   </restriction>
</simpleType>
```

7.1.3.11 ST_Shp (Shape (Delimiters))

Delimiters shape can be centered around the argument, or matched to the shape of the argument.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
centered (Centered (Delimiters))	Delimiters are centered around their argument.
match (Match)	Match shape of contents of delimiters.

Referenced By
shp@val (§7.1.2.97)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Shp">
   <restriction base="xsd:string">
      <enumeration value="centered"/>
      <enumeration value="match"/>
   </restriction>
</simpleType>
```

7.1.3.12 ST_SpacingRule (Spacing Rule)

Integer value (0 to 4), representing the type of spacing between rows.

This simple type's contents are a restriction of the XML Schema integer datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 0.
- This simple type has a maximum value of less than or equal to 4.

Referenced By
cGpRule@val (§7.1.2.19); rSpRule@val (§7.1.2.93)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_SpacingRule">
   <restriction base="xsd:integer">
      <minInclusive value="0"/>
      <maxInclusive value="4"/>
   </restriction>
</simpleType>
```

7.1.3.13 ST_String (String)

String

This simple type's contents are a restriction of the XML Schema string datatype.

Referenced By
mathFont@val (§7.1.2.61)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_String">
   <restriction base="xsd:string"/>
</simpleType>
```

7.1.3.14 ST_Style (Style)

Style of math can be plain, bold, italic, or bold-italic (p, bi, i, or bi).

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
b (Bold)	Bold
bi (Bold-Italic)	Bold-Italic
i (Italic)	Italic
p (Plain)	Plain

Referenced By
sty@val (§7.1.2.111)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Style">
   <restriction base="xsd:string">
      <enumeration value="p"/>
      <enumeration value="b"/>
      <enumeration value="i"/>
      <enumeration value="bi"/>
   </restriction>
</simpleType>
```

7.1.3.15 ST_TopBot (Top-Bottom)

Possible values are `top` and `bot`.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
bot (Bottom Alignment)	Aligns the bottom of the function to the baseline of the surrounding text.
top (Top)	Aligns the top row of the function to the baseline of the surrounding text.

Referenced By
pos@val (§7.1.2.84); vertJc@val (§7.1.2.119)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TopBot">
   <restriction base="xsd:string">
      <enumeration value="top"/>
      <enumeration value="bot"/>
   </restriction>
</simpleType>
```

7.1.3.16 ST_TwipsMeasure (Twips measurement)

Positive measurement in twips (twentieths of a point, 1/1440 of an inch).

This simple type's contents are a restriction of the XML Schema unsignedInt datatype.

Referenced By
interSp@val (§7.1.2.48); intraSp@val (§7.1.2.50); lMargin@val (§7.1.2.59); postSp@val (§7.1.2.85); preSp@val (§7.1.2.86); rMargin@val (§7.1.2.90); wrapIndent@val (§7.1.2.120)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_TwipsMeasure">
   <restriction base="xsd:unsignedInt"/>
</simpleType>
```

7.1.3.17 ST_UnSignedInteger (Unsigned integer.)

Unsigned Integer

This simple type's contents are a restriction of the XML Schema unsignedInt datatype.

Referenced By
cGp@val (§7.1.2.18); cSp@val (§7.1.2.22); rSp@val (§7.1.2.92)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_UnSignedInteger">
   <restriction base="xsd:unsignedInt"/>
</simpleType>
```

7.1.3.18 ST_XAlign (Horizontal Alignment)

Possible values are `left`, `center`, and `right`.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
center (Center)	Centers the argument in the containing function.
left (Left Justification)	Aligns the argument to the left of the containing

Enumeration Value	Description
	function.
right (Right)	Aligns the argument to the right of the containing function.

Referenced By
mcJc@val (§7.1.2.65)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_XAlign">
   <restriction base="xsd:string">
      <enumeration value="left"/>
      <enumeration value="center"/>
      <enumeration value="right"/>
   </restriction>
</simpleType>
```

7.1.3.19 ST_YAlign (Vertical Alignment)

Possible values are `bot`, `center`, and `top`.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
bot (Bottom Alignment)	Aligns the bottom of the function to the baseline of the surrounding text.
center (Center (Function))	Centers the argument in the containing function.
top (Top)	Aligns the top row of the function to the baseline of the surrounding text.

Referenced By
baseJc@val (§7.1.2.9)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_YAlign">
   <restriction base="xsd:string">
      <enumeration value="top"/>
      <enumeration value="center"/>
      <enumeration value="bot"/>
   </restriction>
</simpleType>
```

7.2 Extended Properties

Extended properties are a predefined set of metadata properties that are applicable to Office Open XML documents. These properties extend the set of core properties defined in Part 2: "Open Packaging Conventions" which are common to all packages.

Extended properties are stored within an Extended File Properties part with:

- Source Relationship: http://schemas.openxmlformats.org/officeDocument/2006/relationships/extended-properties
- Content Type: application/vnd.openxmlformats-officedocument.extended-properties+xml

Each extended property is represented as an element in the extended properties part. Extended properties elements are non-repeatable and may be empty or omitted. If all extended property elements are omitted then the extended properties part may be excluded from a document.

[*Example*: A sample extended file properties part:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Properties xmlns="http://…/extended-properties"
  xmlns:vt="http://…/docPropsVTypes">
..<Template>Sales Report.dotm</Template>
..<TotalTime>0</TotalTime>
..<Pages>1</Pages>
..<Words>166</Words>
..<Characters>948</Characters>
..<Application>Microsoft Office Word</Application>
..<DocSecurity>0</DocSecurity>
..<Lines>7</Lines>
..<Paragraphs>2</Paragraphs>
..<ScaleCrop>false</ScaleCrop>
..<Company>Northwind Traders</Company>
..<LinksUpToDate>false</LinksUpToDate>
..<CharactersWithSpaces>1112</CharactersWithSpaces>
..<SharedDoc>false</SharedDoc>
..<HyperlinksChanged>false</HyperlinksChanged>
..<AppVersion>12.0000</AppVersion>
</Properties>
```

end example]

7.2.1 Table of Contents

This subclause is informative.

End of informative text.

7.2.2 Elements

The following elements specify the contents of this namespace:

7.2.2.1 Application (Application Name)

This element specifies the name of the application that created this document.

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
Properties (§7.2.2.21)

7.2.2.2 AppVersion (Application Version)

This element specifies the version of the application which produced this document.

The content of this element shall be of the form XX.YYYY where X and Y represent numerical values, or the document shall be considered non-conformant.

[*Note*: The contents of this element do not represent absolute values, but rather qualify the contents of the Application element to differentiate between different versions of the same producer. Applications should use this information in an informative manner only (as document metadata). *end note*]

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
Properties (§7.2.2.21)

7.2.2.3 Characters (Total Number of Characters)

This element specifies the total number of characters in a document.

The possible values for this element are defined by the XML Schema int datatype.

Parent Elements
Properties (§7.2.2.21)

7.2.2.4 CharactersWithSpaces (Number of Characters (With Spaces))

This element specifies the last count of the number of characters (including spaces) in this document.

The possible values for this element are defined by the XML Schema int datatype.

Parent Elements
Properties (§7.2.2.21)

7.2.2.5 Company (Name of Company)

This element specifies the name of a company associated with the document.

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
Properties (§7.2.2.21)

7.2.2.6 DigSig (Digital Signature)

This element contains the signature of a digitally signed document.

[*Note*: This property is a mechanism used by legacy documents to store the digital signature of its binary representation, and should be considered deprecated in favor of the well-defined mechanism defined in Part 2. Any use of this property should be for legacy compatibility only, and is application-defined. *end note*]

Parent Elements

Parent Elements
Properties (§7.2.2.21)

Child Elements	Subclause
blob (Binary Blob)	§7.4.2.2

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DigSigBlob">
   <sequence minOccurs="1" maxOccurs="1">
      <element ref="vt:blob"/>
   </sequence>
</complexType>
```

7.2.2.7 DocSecurity (Document Security)

This metadata element specifies the security level of a document as a numeric value. Document security is defined as:

DocSecurity	Security Level
1	Document is password protected.
2	Document is recommended to be opened as read-only.
4	Document is enforced to be opened as read-only.
8	Document is locked for annotation.

The possible values for this element are defined by the XML Schema int datatype.

Parent Elements
Properties (§7.2.2.21)

7.2.2.8 HeadingPairs (Heading Pairs)

Heading pairs indicates the grouping of document parts and the number of parts in each group. These parts are not document parts but conceptual representations of document sections.

[*Example*: A presentation composing of three slides with an applied theme may have the following HeadingPairs representation:

```
<HeadingPairs>
  <vt:vector size="4" baseType="variant">
    <vt:variant>
      <vt:lpstr>Theme</vt:lpstr>
    </vt:variant>
```

```
      <vt:variant>
        <vt:i4>1</vt:i4>
      </vt:variant>
      <vt:variant>
        <vt:lpstr>Slide Titles</vt:lpstr>
      </vt:variant>
      <vt:variant>
        <vt:i4>3</vt:i4>
      </vt:variant>
    </vt:vector>
  </HeadingPairs>
```

end example]

Parent Elements
Properties (§7.2.2.21)

Child Elements	Subclause
vector (Vector)	§7.4.2.34

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_VectorVariant">
   <sequence minOccurs="1" maxOccurs="1">
      <element ref="vt:vector"/>
   </sequence>
</complexType>
```

7.2.2.9 HiddenSlides (Number of Hidden Slides)

This element specifies the number of hidden slides in a presentation document.

The possible values for this element are defined by the XML Schema int datatype.

Parent Elements
Properties (§7.2.2.21)

7.2.2.10 HLinks (Hyperlink List)

This element specifies the set of hyperlinks that were in this document when last saved.

Parent Elements
Properties (§7.2.2.21)

Child Elements	Subclause

Child Elements	Subclause
vector (Vector)	§7.4.2.34

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_VectorVariant">
   <sequence minOccurs="1" maxOccurs="1">
      <element ref="vt:vector"/>
   </sequence>
</complexType>
```

7.2.2.11 HyperlinkBase (Relative Hyperlink Base)

This element specifies the base string used for evaluating relative hyperlinks in this document.

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
Properties (§7.2.2.21)

7.2.2.12 HyperlinksChanged (Hyperlinks Changed)

This element specifies that one or more hyperlinks in this part were updated exclusively in this part by a producer. The next producer to open this document shall update the hyperlink relationships with the new hyperlinks specified in this part.

The possible values for this element are defined by the XML Schema boolean datatype.

Parent Elements
Properties (§7.2.2.21)

7.2.2.13 Lines (Number of Lines)

This element specifies the total number of lines in a document when last saved by a conforming producer if applicable.

The possible values for this element are defined by the XML Schema int datatype.

Parent Elements
Properties (§7.2.2.21)

7.2.2.14 LinksUpToDate (Links Up-to-Date)

This element indicates whether hyperlinks in a document are up-to-date. Set this element to TRUE to indicate that hyperlinks are updated. Set this element to FALSE to indicate that hyperlinks are outdated.

The possible values for this element are defined by the XML Schema boolean datatype.

Parent Elements

Parent Elements
Properties (§7.2.2.21)

7.2.2.15 Manager (Name of Manager)

This element specifies the name of a supervisor associated with the document.

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
Properties (§7.2.2.21)

7.2.2.16 MMClips (Total Number of Multimedia Clips)

This element specifies the total number of sound or video clips that are present in the document.

The possible values for this element are defined by the XML Schema int datatype.

Parent Elements
Properties (§7.2.2.21)

7.2.2.17 Notes (Number of Slides Containing Notes)

This element specifies the number of slides in a presentation containing notes.

The possible values for this element are defined by the XML Schema int datatype.

Parent Elements
Properties (§7.2.2.21)

7.2.2.18 Pages (Total Number of Pages)

This element specifies the total number of pages of a document if applicable.

The possible values for this element are defined by the XML Schema int datatype.

Parent Elements
Properties (§7.2.2.21)

7.2.2.19 Paragraphs (Total Number of Paragraphs)

This element specifies the total number of paragraphs found in a document if applicable.

The possible values for this element are defined by the XML Schema int datatype.

Parent Elements
Properties (§7.2.2.21)

7.2.2.20 PresentationFormat (Intended Format of Presentation)

This element specifies the intended format for a presentation document. For example, a presentation intended to be shown on video will have PresentationFormat "Video".

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
Properties (§7.2.2.21)

7.2.2.21 Properties (Application Specific File Properties)

This element specifies the application properties of a document. For properties of type string, NCR escape format (_xHHHH_) is used for any invalid XML characters.

Parent Elements
Root element of Shared Extended File Properties part

Child Elements	Subclause
Application (Application Name)	§7.2.2.1
AppVersion (Application Version)	§7.2.2.2
Characters (Total Number of Characters)	§7.2.2.3
CharactersWithSpaces (Number of Characters (With Spaces))	§7.2.2.4
Company (Name of Company)	§7.2.2.5
DigSig (Digital Signature)	§7.2.2.6
DocSecurity (Document Security)	§7.2.2.7
HeadingPairs (Heading Pairs)	§7.2.2.8
HiddenSlides (Number of Hidden Slides)	§7.2.2.9
HLinks (Hyperlink List)	§7.2.2.10
HyperlinkBase (Relative Hyperlink Base)	§7.2.2.11
HyperlinksChanged (Hyperlinks Changed)	§7.2.2.12
Lines (Number of Lines)	§7.2.2.13
LinksUpToDate (Links Up-to-Date)	§7.2.2.14
Manager (Name of Manager)	§7.2.2.15
MMClips (Total Number of Multimedia Clips)	§7.2.2.16
Notes (Number of Slides Containing Notes)	§7.2.2.17
Pages (Total Number of Pages)	§7.2.2.18
Paragraphs (Total Number of Paragraphs)	§7.2.2.19
PresentationFormat (Intended Format of Presentation)	§7.2.2.20

Child Elements	Subclause
ScaleCrop (Thumbnail Display Mode)	§7.2.2.22
SharedDoc (Shared Document)	§7.2.2.23
Slides (Slides Metadata Element)	§7.2.2.24
Template (Name of Document Template)	§7.2.2.25
TitlesOfParts (Part Titles)	§7.2.2.26
TotalTime (Total Edit Time Metadata Element)	§7.2.2.27
Words (Word Count)	§7.2.2.28

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Properties">
   <all>
      <element name="Template" minOccurs="0" maxOccurs="1" type="xsd:string"/>
      <element name="Manager" minOccurs="0" maxOccurs="1" type="xsd:string"/>
      <element name="Company" minOccurs="0" maxOccurs="1" type="xsd:string"/>
      <element name="Pages" minOccurs="0" maxOccurs="1" type="xsd:int"/>
      <element name="Words" minOccurs="0" maxOccurs="1" type="xsd:int"/>
      <element name="Characters" minOccurs="0" maxOccurs="1" type="xsd:int"/>
      <element name="PresentationFormat" minOccurs="0" maxOccurs="1" type="xsd:string"/>
      <element name="Lines" minOccurs="0" maxOccurs="1" type="xsd:int"/>
      <element name="Paragraphs" minOccurs="0" maxOccurs="1" type="xsd:int"/>
      <element name="Slides" minOccurs="0" maxOccurs="1" type="xsd:int"/>
      <element name="Notes" minOccurs="0" maxOccurs="1" type="xsd:int"/>
      <element name="TotalTime" minOccurs="0" maxOccurs="1" type="xsd:int"/>
      <element name="HiddenSlides" minOccurs="0" maxOccurs="1" type="xsd:int"/>
      <element name="MMClips" minOccurs="0" maxOccurs="1" type="xsd:int"/>
      <element name="ScaleCrop" minOccurs="0" maxOccurs="1" type="xsd:boolean"/>
      <element name="HeadingPairs" minOccurs="0" maxOccurs="1" type="CT_VectorVariant"/>
      <element name="TitlesOfParts" minOccurs="0" maxOccurs="1" type="CT_VectorLpstr"/>
      <element name="LinksUpToDate" minOccurs="0" maxOccurs="1" type="xsd:boolean"/>
      <element name="CharactersWithSpaces" minOccurs="0" maxOccurs="1" type="xsd:int"/>
      <element name="SharedDoc" minOccurs="0" maxOccurs="1" type="xsd:boolean"/>
      <element name="HyperlinkBase" minOccurs="0" maxOccurs="1" type="xsd:string"/>
      <element name="HLinks" minOccurs="0" maxOccurs="1" type="CT_VectorVariant"/>
      <element name="HyperlinksChanged" minOccurs="0" maxOccurs="1" type="xsd:boolean"/>
      <element name="DigSig" minOccurs="0" maxOccurs="1" type="CT_DigSigBlob"/>
      <element name="Application" minOccurs="0" maxOccurs="1" type="xsd:string"/>
      <element name="AppVersion" minOccurs="0" maxOccurs="1" type="xsd:string"/>
      <element name="DocSecurity" minOccurs="0" maxOccurs="1" type="xsd:int"/>
   </all>
</complexType>
```

7.2.2.22 ScaleCrop (Thumbnail Display Mode)

This element indicates the display mode of the document thumbnail. Set this element to TRUE to enable scaling of the document thumbnail to the display. Set this element to FALSE to enable cropping of the document thumbnail to show only sections that will fit the display.

The possible values for this element are defined by the XML Schema boolean datatype.

Parent Elements
Properties (§7.2.2.21)

7.2.2.23 SharedDoc (Shared Document)

This element indicates if this document is currently shared between multiple producers. If this element is set to TRUE, producers should take care when updating the document.

The possible values for this element are defined by the XML Schema boolean datatype.

Parent Elements
Properties (§7.2.2.21)

7.2.2.24 Slides (Slides Metadata Element)

This element specifies the total number of slides in a presentation document.

The possible values for this element are defined by the XML Schema int datatype.

Parent Elements
Properties (§7.2.2.21)

7.2.2.25 Template (Name of Document Template)

This element specifies the name of an external document template containing format and style information used to create the current document.

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
Properties (§7.2.2.21)

7.2.2.26 TitlesOfParts (Part Titles)

This element specifies the title of each document. These parts are not document parts but conceptual representations of document sections.

[*Example*: A presentation composing of three slides with an applied theme "Currency" may have the following TitlesofParts representation:

```
<TitlesofParts>
  <vt:vector size="4" baseType="lpstr">
    <vt:lpstr>Currency</vt:lpstr>
    <vt:lpstr>Slide 1</vt:lpstr>
    <vt:lpstr>Slide 2</vt:lpstr>
    <vt:lpstr>Slide 3</vt:lpstr>
  </vt:vector>
</TitlesofParts>
```

end example]

Parent Elements
Properties (§7.2.2.21)

Child Elements	Subclause
vector (Vector)	§7.4.2.34

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_VectorLpstr">
   <sequence minOccurs="1" maxOccurs="1">
      <element ref="vt:vector"/>
   </sequence>
</complexType>
```

7.2.2.27 TotalTime (Total Edit Time Metadata Element)

Total time that a document has been edited. The default time unit is minutes.

The possible values for this element are defined by the XML Schema int datatype.

Parent Elements
Properties (§7.2.2.21)

7.2.2.28 Words (Word Count)

This element specifies the total number of words contained in a document when last saved.

The possible values for this element are defined by the XML Schema int datatype.

Parent Elements
Properties (§7.2.2.21)

7.3 Custom Properties

Custom properties enable users to define custom metadata properties through a set of well-defined data types.

Custom properties are represented by property elements (§7.3.2.2) stored in the Custom File Properties part with:

- Source Relationship: http://schemas.openxmlformats.org/officeDocument/2006/relationships/custom-properties
- Content Type: application/vnd.openxmlformats-officedocument.custom-properties+xml

Custom property elements are non-repeatable and may be empty or omitted. If all custom property elements are omitted then the custom properties part may be excluded from a document.

The type and value of custom properties are specified by child XML elements in the File Properties Variant Types namespace (discussed in detail in §7.4). User defined properties are uniquely identified through the name attribute of the property element. Custom properties can be associated with OLE document properties through the fmtid and pid attributes.

[*Example*: A custom OLE `Editor` property of type `string` can be defined as follows:

```
<property fmtid="{D5CDD505-2E9C-101B-9397-08002B2CF9AE}" pid="2"
  name="Editor">
  <vt:lpwstr>John Smith</vt:lpwstr>
</property>
```

end example]

7.3.1 Table of Contents

This subclause is informative.

End of informative text.

7.3.2 Elements

This subclause specifies the set of elements that define this namespace:

7.3.2.1 Properties (Custom File Properties)

Parent element for the custom file properties part.

Parent Elements
Root element of Shared Custom File Properties part

Child Elements	Subclause

Child Elements	Subclause
property (Custom File Property)	§7.3.2.2

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Properties">
   <sequence>
      <element name="property" minOccurs="0" maxOccurs="unbounded" type="CT_Property"/>
   </sequence>
</complexType>
```

7.3.2.2 property (Custom File Property)

This element specifies a single custom file property. Custom file property type is defined through child elements in the File Properties Variant Type namespace. Custom file property value can be set by setting the appropriate Variant Type child element value.

Parent Elements
Properties (§7.3.2.1)

Child Elements	Subclause
array (Array)	§7.4.2.1
blob (Binary Blob)	§7.4.2.2
bool (Boolean)	§7.4.2.3
bstr (Basic String)	§7.4.2.4
cf (Clipboard Data)	§7.4.2.5
clsid (Class ID)	§7.4.2.6
cy (Currency)	§7.4.2.7
date (Date and Time)	§7.4.2.8
decimal (Decimal)	§7.4.2.9
empty (Empty)	§7.4.2.10
error (Error Status Code)	§7.4.2.11
filetime (File Time)	§7.4.2.12
i1 (1-Byte Signed Integer)	§7.4.2.13
i2 (2-Byte Signed Integer)	§7.4.2.14
i4 (4-Byte Signed Integer)	§7.4.2.15
i8 (8-Byte Signed Integer)	§7.4.2.16
int (Integer)	§7.4.2.17
lpstr (LPSTR)	§7.4.2.18
lpwstr (LPWSTR)	§7.4.2.19

Child Elements	Subclause
null (Null)	§7.4.2.20
oblob (Binary Blob Object)	§7.4.2.21
ostorage (Binary Storage Object)	§7.4.2.22
ostream (Binary Stream Object)	§7.4.2.23
r4 (4-Byte Real Number)	§7.4.2.24
r8 (8-Byte Real Number)	§7.4.2.25
storage (Binary Storage)	§7.4.2.26
stream (Binary Stream)	§7.4.2.27
ui1 (1-Byte Unsigned Integer)	§7.4.2.28
ui2 (2-Byte Unsigned Integer)	§7.4.2.29
ui4 (4-Byte Unsigned Integer)	§7.4.2.30
ui8 (8-Byte Unsigned Integer)	§7.4.2.31
uint (Unsigned Integer)	§7.4.2.32
vector (Vector)	§7.4.2.34
vstream (Binary Versioned Stream)	§7.4.2.35

Attributes	Description
fmtid (Format ID)	Uniquely relates a custom property with an OLE property. The value of this attribute is a Globally Unique Identifier in the form of {HHHHHHHH-HHHH-HHHH-HHHH-HHHHHHHH} where each H is a hexidecimal. The possible values for this attribute are defined by the ST_Clsid simple type (§7.4.3.2).
linkTarget (Bookmark Link Target)	Specifies the name of a bookmark in the current document (for WordprocessingML), or a table or named cell (for SpreadsheetML) from which the value of this custom document property should be extracted. If this attribute is present, then any value under this element shall be considered a cache and replaced with the value of this bookmark (if present) on save. If the bookmark is not present, then this link shall be considered broken and the cached value shall be retained. The possible values for this attribute are defined by the XML Schema string datatype.
name (Custom File Property Name)	Specifies the name of this custom file property. The possible values for this attribute are defined by the XML Schema string datatype.
pid (Property ID)	Uniquely relates a custom property with an OLE property. The possible values for this attribute are defined by the XML Schema int datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Property">
   <choice minOccurs="1" maxOccurs="1">
      <element ref="vt:vector"/>
      <element ref="vt:array"/>
      <element ref="vt:blob"/>
      <element ref="vt:oblob"/>
      <element ref="vt:empty"/>
      <element ref="vt:null"/>
      <element ref="vt:i1"/>
      <element ref="vt:i2"/>
      <element ref="vt:i4"/>
      <element ref="vt:i8"/>
      <element ref="vt:int"/>
      <element ref="vt:ui1"/>
      <element ref="vt:ui2"/>
      <element ref="vt:ui4"/>
      <element ref="vt:ui8"/>
      <element ref="vt:uint"/>
      <element ref="vt:r4"/>
      <element ref="vt:r8"/>
      <element ref="vt:decimal"/>
      <element ref="vt:lpstr"/>
      <element ref="vt:lpwstr"/>
      <element ref="vt:bstr"/>
      <element ref="vt:date"/>
      <element ref="vt:filetime"/>
      <element ref="vt:bool"/>
      <element ref="vt:cy"/>
      <element ref="vt:error"/>
      <element ref="vt:stream"/>
      <element ref="vt:ostream"/>
      <element ref="vt:storage"/>
      <element ref="vt:ostorage"/>
      <element ref="vt:vstream"/>
      <element ref="vt:clsid"/>
      <element ref="vt:cf"/>
   </choice>
   <attribute name="fmtid" use="required" type="vt:ST_Clsid"/>
   <attribute name="pid" use="required" type="xsd:int"/>
   <attribute name="name" use="optional" type="xsd:string"/>
   <attribute name="linkTarget" use="optional" type="xsd:string"/>
</complexType>
```

7.4 Variant Types

This subclause specifies the set of data types which may be included within file properties that accept variant type structures.

7.4.1 Table of Contents

This subclause is informative.

End of informative text.

7.4.2 Elements

The following elements define the contents of this namespace:

7.4.2.1 array (Array)

The array element defines the array variant type. Array contents must be of uniform type as specified by the baseType attribute. The contents of an array are defined using repeated child elements of the appropriate variant type.

Multi-dimensional arrays can be defined by specifying the length of each dimension in the lBound and uBound attributes through the use of the "," delimiter. Child elements of multi-dimensional arrays are indexed along each dimension in the order the dimensions are declared.

In other words, the array shall be filled as follows:

- The first index shall be incremented to its maximum value [*Example*: [0,0,0] to [max,0,0] *end example*]
- The second index shall be incremented to its maximum value [*Example*: [0,1,0] to [0,max,0] *end example*]
- Subsequent indices shall be filled until all provided values have been added
- All other values shall have null values within the array (i.e. no default value shall be assumed).

[*Example*: A 2x3 variant type array of type "i4" is specified as follows:

```
<vt:array lBounds="0,0" uBounds="1,2" baseType="i4">
  <vt:i4>0</vt:i4>
  <vt:i4>1</vt:i4>
  <vt:i4>2</vt:i4>
  <vt:i4>3</vt:i4>
  <vt:i4>4</vt:i4>
</vt:array>
```

The resulting array: [0,0] = 0, [1,0] = 1, [0,1] = 2, [1,1] = 3, [0,2] = 4. *end example*]

Parent Elements
property (§7.3.2.2); variant (§7.4.2.33)

Child Elements	Subclause
bool (Boolean)	§7.4.2.3
bstr (Basic String)	§7.4.2.4
cy (Currency)	§7.4.2.7
date (Date and Time)	§7.4.2.8
decimal (Decimal)	§7.4.2.9

Child Elements	Subclause
error (Error Status Code)	§7.4.2.11
i1 (1-Byte Signed Integer)	§7.4.2.13
i2 (2-Byte Signed Integer)	§7.4.2.14
i4 (4-Byte Signed Integer)	§7.4.2.15
int (Integer)	§7.4.2.17
r4 (4-Byte Real Number)	§7.4.2.24
r8 (8-Byte Real Number)	§7.4.2.25
ui1 (1-Byte Unsigned Integer)	§7.4.2.28
ui2 (2-Byte Unsigned Integer)	§7.4.2.29
ui4 (4-Byte Unsigned Integer)	§7.4.2.30
uint (Unsigned Integer)	§7.4.2.32
variant (Variant)	§7.4.2.33

Attributes	Description
baseType (Array Base Type)	The baseType attribute specifies the base variant type of an array. The allowed values are: variant, i1, i2, i4, int, ui1, ui2, ui4, uint, r4, r8, decimal, bstr, date, bool, cy, and error. The possible values for this attribute are defined by the ST_ArrayBaseType simple type (§7.4.3.1).
lBounds (Array Lower Bounds Attribute)	The lBounds attribute specifies the lower bound of an array in the format: #, #, # ... # where each # represents an integer. The possible values for this attribute are defined by the XML Schema int datatype.
uBounds (Array Upper Bounds Attribute)	The uBounds attribute specifies the upper bound of an array in the format: #, #, # ... # where each # represents an integer. The possible values for this attribute are defined by the XML Schema int datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Array">
   <choice minOccurs="1" maxOccurs="unbounded">
      <element ref="variant"/>
      <element ref="i1"/>
      <element ref="i2"/>
      <element ref="i4"/>
      <element ref="int"/>
      <element ref="ui1"/>
      <element ref="ui2"/>
      <element ref="ui4"/>
      <element ref="uint"/>
      <element ref="r4"/>
      <element ref="r8"/>
      <element ref="decimal"/>
      <element ref="bstr"/>
      <element ref="date"/>
      <element ref="bool"/>
      <element ref="error"/>
      <element ref="cy"/>
   </choice>
   <attribute name="lBounds" type="xsd:int" use="required"/>
   <attribute name="uBounds" type="xsd:int" use="required"/>
   <attribute name="baseType" type="ST_ArrayBaseType" use="required"/>
</complexType>
```

7.4.2.2 blob (Binary Blob)

This element specifies a base64 binary blob variant type.

This type is defined as follows: a DWORD count of bytes, followed by that many bytes of data. The byte count does not include the four bytes for the length of the count itself; an empty blob member would have a count of zero, followed by zero bytes.

The possible values for this element are defined by the XML Schema base64Binary datatype.

Parent Elements
DigSig (§7.2.2.6); property (§7.3.2.2); variant (§7.4.2.33)

7.4.2.3 bool (Boolean)

This element specifies a Boolean variant type.

The possible values for this element are defined by the XML Schema boolean datatype.

Parent Elements
array (§7.4.2.1); property (§7.3.2.2); variant (§7.4.2.33); vector (§7.4.2.34)

7.4.2.4 bstr (Basic String)

This element defines a binary basic string variant type. For all characters that cannot be represented in XML as defined by the XML 1.0 specification, the characters are escaped using the Unicode numerical character representation escape character format _xHHHH_, where H represents a hexadecimal character in the character's value. [*Example*: The Unicode character 8 is not permitted in an XML 1.0 document, so it shall be escaped as _x0008_. *end example*]

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
array (§7.4.2.1); property (§7.3.2.2); variant (§7.4.2.33); vector (§7.4.2.34)

7.4.2.5 cf (Clipboard Data)

This element specifies base64-encoded binary clipboard data.

Parent Elements
property (§7.3.2.2); variant (§7.4.2.33); vector (§7.4.2.34)

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>format (Format Attribute)</td><td>Specifies the format of the clipboard data and must be: -3, -2, -1, 0, or any positive integer.

<table>
<tr><th>Format Value</th><th>Clipboard Data</th></tr>
<tr><td>0</td><td>No data.</td></tr>
<tr><td>-1</td><td>A long that contains a built-in Windows clipboard format value.</td></tr>
<tr><td>-2</td><td>A long that contains a Macintosh clipboard format value.</td></tr>
<tr><td>-3</td><td>A GUID that contains a format identifier (FMTID).</td></tr>
<tr><td>any positive value</td><td>A null-terminated string that contains a clipboard format name to be registered. The Format value indicates the string length, including the null byte at the end.</td></tr>
</table>

The possible values for this attribute are defined by the ST_Format simple type (§7.4.3.5).</td></tr>
</table>

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Cf">
    <simpleContent>
        <extension base="xsd:base64Binary">
            <attribute name="format" type="ST_Format"/>
        </extension>
    </simpleContent>
</complexType>
```

7.4.2.6 clsid (Class ID)

This element specifies a class ID variant type. The value must be a Globally Unique Identifier with format: {HHHHHHHH-HHHH-HHHH-HHHH-HHHHHHHH}.

The possible values for this element are defined by the ST_Clsid simple type (§7.4.3.2).

Parent Elements
property (§7.3.2.2); variant (§7.4.2.33); vector (§7.4.2.34)

7.4.2.7 cy (Currency)

This element specifies a currency variant type with exactly four digits after the decimal point.

The possible values for this element are defined by the ST_Cy simple type (§7.4.3.3).

Parent Elements
array (§7.4.2.1); property (§7.3.2.2); variant (§7.4.2.33); vector (§7.4.2.34)

7.4.2.8 date (Date and Time)

This element specifies a date variant type of type date-time as defined in RFC 3339.

The possible values for this element are defined by the XML Schema dateTime datatype.

Parent Elements
array (§7.4.2.1); property (§7.3.2.2); variant (§7.4.2.33); vector (§7.4.2.34)

7.4.2.9 decimal (Decimal)

This element specifies a decimal variant type.

The possible values for this element are defined by the XML Schema decimal datatype.

Parent Elements
array (§7.4.2.1); property (§7.3.2.2); variant (§7.4.2.33)

7.4.2.10 empty (Empty)

This element specifies an empty variant type. No values or child elements are allowed.

Parent Elements
property (§7.3.2.2); variant (§7.4.2.33)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Empty"/>
```

7.4.2.11 error (Error Status Code)

The error element specifies a 32-bit error status code variant type of the form 0xHHHHHHHH. Each H represents a hexadecimal digit.

The possible values for this element are defined by the ST_Error simple type (§7.4.3.4).

Parent Elements
array (§7.4.2.1); property (§7.3.2.2); variant (§7.4.2.33); vector (§7.4.2.34)

7.4.2.12 filetime (File Time)

This element specifies a file-time variant type of type date-time as defined in RFC 3339.

The possible values for this element are defined by the XML Schema dateTime datatype.

Parent Elements
property (§7.3.2.2); variant (§7.4.2.33); vector (§7.4.2.34)

7.4.2.13 i1 (1-Byte Signed Integer)

This element specifies a 1-byte signed integer variant type.

The possible values for this element are defined by the XML Schema byte datatype.

Parent Elements
array (§7.4.2.1); property (§7.3.2.2); variant (§7.4.2.33); vector (§7.4.2.34)

7.4.2.14 i2 (2-Byte Signed Integer)

This element specifies a 2-byte signed integer variant type.

The possible values for this element are defined by the XML Schema short datatype.

Parent Elements
array (§7.4.2.1); property (§7.3.2.2); variant (§7.4.2.33); vector (§7.4.2.34)

7.4.2.15 i4 (4-Byte Signed Integer)

This element specifies a 4-byte signed integer variant type.

The possible values for this element are defined by the XML Schema int datatype.

Parent Elements
array (§7.4.2.1); property (§7.3.2.2); variant (§7.4.2.33); vector (§7.4.2.34)

7.4.2.16 i8 (8-Byte Signed Integer)

This element specifies a 8-byte signed integer variant type.

The possible values for this element are defined by the XML Schema long datatype.

Parent Elements
property (§7.3.2.2); variant (§7.4.2.33); vector (§7.4.2.34)

7.4.2.17 int (Integer)

This element specifies an integer variant type.

The possible values for this element are defined by the XML Schema int datatype.

Parent Elements
array (§7.4.2.1); property (§7.3.2.2); variant (§7.4.2.33)

7.4.2.18 lpstr (LPSTR)

This element specifies a string variant type. For all characters that cannot be represented in XML as defined by the XML 1.0 specification, the characters are escaped using the Unicode numerical character representation escape character format _xHHHH_, where H represents a hexadecimal character in the character's value. [*Example*: The Unicode character 8 is not permitted in an XML 1.0 document, so it shall be escaped as _x0008_. *end example*]

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
property (§7.3.2.2); variant (§7.4.2.33); vector (§7.4.2.34)

7.4.2.19 lpwstr (LPWSTR)

This element specifies a string variant type. For all characters that cannot be represented in XML as defined by the XML 1.0 specification, the characters are escaped using the Unicode numerical character representation escape character format _xHHHH_, where H represents a hexadecimal character in the character's value. [*Example*: The Unicode character 8 is not permitted in an XML 1.0 document, so it shall be escaped as _x0008_. *end example*]

The possible values for this element are defined by the XML Schema string datatype.

Parent Elements
property (§7.3.2.2); variant (§7.4.2.33); vector (§7.4.2.34)

7.4.2.20 null (Null)

This element specifies a null variant type.

Parent Elements
property (§7.3.2.2); variant (§7.4.2.33)

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Null"/>
```

7.4.2.21 oblob (Binary Blob Object)

This element specifies a base64 binary blob object variant type.

This type is defined as follows: A blob member that contains a serialized object in the same representation that would appear in the ostream element. That is, a DWORD byte count (where the byte count does not include the size of itself) which is in the format of a class identifier followed by initialization data for that class.

The possible values for this element are defined by the XML Schema base64Binary datatype.

Parent Elements
property (§7.3.2.2); variant (§7.4.2.33)

7.4.2.22 ostorage (Binary Storage Object)

This element specifies a base64 binary storage object variant type.

This type is defined as follows: Identical to the storage element, but indicates that the designated storage shall contain a loadable object.

The possible values for this element are defined by the XML Schema base64Binary datatype.

Parent Elements
property (§7.3.2.2); variant (§7.4.2.33)

7.4.2.23 ostream (Binary Stream Object)

This element specifies a binary stream object variant type.

This type is defined as follows: Identical to the definition of the stream element, but indicates that the stream contains a serialized object, which is a CLSID – see the ST_Clsid simple type (§7.4.3.2) – followed by initialization data for the specified class.

The possible values for this element are defined by the XML Schema base64Binary datatype.

Parent Elements
property (§7.3.2.2); variant (§7.4.2.33)

7.4.2.24 r4 (4-Byte Real Number)

This element specifies a 4-byte real number variant type.

The possible values for this element are defined by the XML Schema float datatype.

Parent Elements
array (§7.4.2.1); property (§7.3.2.2); variant (§7.4.2.33); vector (§7.4.2.34)

7.4.2.25 r8 (8-Byte Real Number)

This element specifies a 8-byte real number variant type.

The possible values for this element are defined by the XML Schema double datatype.

Parent Elements
array (§7.4.2.1); property (§7.3.2.2); variant (§7.4.2.33); vector (§7.4.2.34)

7.4.2.26 storage (Binary Storage)

This element specifies a binary storage variant type.

This type is defined as follows: Contains the base64-encoded data for a structured storage.

The possible values for this element are defined by the XML Schema base64Binary datatype.

Parent Elements
property (§7.3.2.2); variant (§7.4.2.33)

7.4.2.27 stream (Binary Stream)

This element specifies a binary stream variant type.

This type is defined as follows: Contains the base64-encoded data for a structured storage stream.

The possible values for this element are defined by the XML Schema base64Binary datatype.

Parent Elements
property (§7.3.2.2); variant (§7.4.2.33)

7.4.2.28 ui1 (1-Byte Unsigned Integer)

This element specifies a 1-byte unsigned integer variant type.

The possible values for this element are defined by the XML Schema unsignedByte datatype.

Parent Elements
array (§7.4.2.1); property (§7.3.2.2); variant (§7.4.2.33); vector (§7.4.2.34)

7.4.2.29 ui2 (2-Byte Unsigned Integer)

This element specifies a 2-byte unsigned integer variant type.

The possible values for this element are defined by the XML Schema unsignedShort datatype.

Parent Elements
array (§7.4.2.1); property (§7.3.2.2); variant (§7.4.2.33); vector (§7.4.2.34)

7.4.2.30 ui4 (4-Byte Unsigned Integer)

This element specifies a 4-byte unsigned integer variant type.

The possible values for this element are defined by the XML Schema unsignedInt datatype.

Parent Elements
array (§7.4.2.1); property (§7.3.2.2); variant (§7.4.2.33); vector (§7.4.2.34)

7.4.2.31 ui8 (8-Byte Unsigned Integer)

This element specifies a 8-byte unsigned integer variant type.

The possible values for this element are defined by the XML Schema unsignedLong datatype.

Parent Elements
property (§7.3.2.2); variant (§7.4.2.33); vector (§7.4.2.34)

7.4.2.32 uint (Unsigned Integer)

This element specifies an unsigned integer variant type.

The possible values for this element are defined by the XML Schema unsignedInt datatype.

Parent Elements
array (§7.4.2.1); property (§7.3.2.2); variant (§7.4.2.33)

7.4.2.33 variant (Variant)

This element can contain exactly 1 child element of any variant type. This element is only valid as a child element of a vector or array variant type.

[*Example*: A vector of variant types:

```
<vt:vector baseType="variant">
  <vt:variant>
    <vt:i4>12</vt:i4>
  </vt:variant>
  <vt:variant>
    <vt:lpstr>WorkSheets</vt:lpstr>
```

```
  </vt:variant>
</vt:vector>
```

end example]

Parent Elements
array (§7.4.2.1); variant (§7.4.2.33); vector (§7.4.2.34)

Child Elements	Subclause
array (Array)	§7.4.2.1
blob (Binary Blob)	§7.4.2.2
bool (Boolean)	§7.4.2.3
bstr (Basic String)	§7.4.2.4
cf (Clipboard Data)	§7.4.2.5
clsid (Class ID)	§7.4.2.6
cy (Currency)	§7.4.2.7
date (Date and Time)	§7.4.2.8
decimal (Decimal)	§7.4.2.9
empty (Empty)	§7.4.2.10
error (Error Status Code)	§7.4.2.11
filetime (File Time)	§7.4.2.12
i1 (1-Byte Signed Integer)	§7.4.2.13
i2 (2-Byte Signed Integer)	§7.4.2.14
i4 (4-Byte Signed Integer)	§7.4.2.15
i8 (8-Byte Signed Integer)	§7.4.2.16
int (Integer)	§7.4.2.17
lpstr (LPSTR)	§7.4.2.18
lpwstr (LPWSTR)	§7.4.2.19
null (Null)	§7.4.2.20
oblob (Binary Blob Object)	§7.4.2.21
ostorage (Binary Storage Object)	§7.4.2.22
ostream (Binary Stream Object)	§7.4.2.23
r4 (4-Byte Real Number)	§7.4.2.24
r8 (8-Byte Real Number)	§7.4.2.25
storage (Binary Storage)	§7.4.2.26
stream (Binary Stream)	§7.4.2.27

Child Elements	Subclause
ui1 (1-Byte Unsigned Integer)	§7.4.2.28
ui2 (2-Byte Unsigned Integer)	§7.4.2.29
ui4 (4-Byte Unsigned Integer)	§7.4.2.30
ui8 (8-Byte Unsigned Integer)	§7.4.2.31
uint (Unsigned Integer)	§7.4.2.32
variant (Variant)	§7.4.2.33
vector (Vector)	§7.4.2.34
vstream (Binary Versioned Stream)	§7.4.2.35

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Variant">
   <choice minOccurs="1" maxOccurs="1">
      <element ref="variant"/>
      <element ref="vector"/>
      <element ref="array"/>
      <element ref="blob"/>
      <element ref="oblob"/>
      <element ref="empty"/>
      <element ref="null"/>
      <element ref="i1"/>
      <element ref="i2"/>
      <element ref="i4"/>
      <element ref="i8"/>
      <element ref="int"/>
      <element ref="ui1"/>
      <element ref="ui2"/>
      <element ref="ui4"/>
      <element ref="ui8"/>
      <element ref="uint"/>
      <element ref="r4"/>
      <element ref="r8"/>
      <element ref="decimal"/>
      <element ref="lpstr"/>
      <element ref="lpwstr"/>
      <element ref="bstr"/>
      <element ref="date"/>
      <element ref="filetime"/>
      <element ref="bool"/>
      <element ref="cy"/>
      <element ref="error"/>
      <element ref="stream"/>
      <element ref="ostream"/>
      <element ref="storage"/>
      <element ref="ostorage"/>
      <element ref="vstream"/>
      <element ref="clsid"/>
      <element ref="cf"/>
   </choice>
</complexType>
```

7.4.2.34 vector (Vector)

This element defines the vector variant type. Vector contents must be of uniform type as specified by the baseType attribute. The contents of a vector are defined using repeated child elements of the appropriate variant type.

[*Example*: A vector of lpstr variant types:

```
<vt:vector baseType="lpstr">
  <vt:lpstr>One</vt:lpstr>
  <vt:lpstr>Two</vt:lpstr>
  <vt:lpstr>Three</vt:lpstr>
</vt:vector>
```

end example]

Parent Elements
HeadingPairs (§7.2.2.8); HLinks (§7.2.2.10); property (§7.3.2.2); TitlesOfParts (§7.2.2.26); variant (§7.4.2.33)

Child Elements	Subclause
bool (Boolean)	§7.4.2.3
bstr (Basic String)	§7.4.2.4
cf (Clipboard Data)	§7.4.2.5
clsid (Class ID)	§7.4.2.6
cy (Currency)	§7.4.2.7
date (Date and Time)	§7.4.2.8
error (Error Status Code)	§7.4.2.11
filetime (File Time)	§7.4.2.12
i1 (1-Byte Signed Integer)	§7.4.2.13
i2 (2-Byte Signed Integer)	§7.4.2.14
i4 (4-Byte Signed Integer)	§7.4.2.15
i8 (8-Byte Signed Integer)	§7.4.2.16
lpstr (LPSTR)	§7.4.2.18
lpwstr (LPWSTR)	§7.4.2.19
r4 (4-Byte Real Number)	§7.4.2.24
r8 (8-Byte Real Number)	§7.4.2.25
ui1 (1-Byte Unsigned Integer)	§7.4.2.28
ui2 (2-Byte Unsigned Integer)	§7.4.2.29
ui4 (4-Byte Unsigned Integer)	§7.4.2.30
ui8 (8-Byte Unsigned Integer)	§7.4.2.31
variant (Variant)	§7.4.2.33

Attributes	Description
baseType (Vector	The baseType attribute specifies the base variant type of a vector.

Attributes	Description
Base Type)	The allowed values are: variant, i1, i2, i4, i8, ui1, ui2, ui4, ui8, r4, r8, lpstr, lpwstr, bstr, date, filetime, bool, cy, error, clsid, and cf. The possible values for this attribute are defined by the ST_VectorBaseType simple type (§7.4.3.6).
size (Vector Size)	Specifies the number of elements in the vector. The possible values for this attribute are defined by the XML Schema unsignedInt datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Vector">
   <choice minOccurs="1" maxOccurs="unbounded">
      <element ref="variant"/>
      <element ref="i1"/>
      <element ref="i2"/>
      <element ref="i4"/>
      <element ref="i8"/>
      <element ref="ui1"/>
      <element ref="ui2"/>
      <element ref="ui4"/>
      <element ref="ui8"/>
      <element ref="r4"/>
      <element ref="r8"/>
      <element ref="lpstr"/>
      <element ref="lpwstr"/>
      <element ref="bstr"/>
      <element ref="date"/>
      <element ref="filetime"/>
      <element ref="bool"/>
      <element ref="cy"/>
      <element ref="error"/>
      <element ref="clsid"/>
      <element ref="cf"/>
   </choice>
   <attribute name="baseType" type="ST_VectorBaseType" use="required"/>
   <attribute name="size" type="xsd:unsignedInt" use="required"/>
</complexType>
```

7.4.2.35 vstream (Binary Versioned Stream)

This element specifies a binary versioned stream variant type.

This type is defined as follows: A stream element's content with a GUID version (the version attribute).

Parent Elements
property (§7.3.2.2); variant (§7.4.2.33)

Attributes	Description
version (VSTREAM Version Attribute)	The version attribute of the vstream element specifies the version as a Globally Unique Identifiers with format: {HHHHHHHH-HHHH-HHHH-HHHH-HHHHHHHH}. The possible values for this attribute are defined by the ST_Clsid simple type (§7.4.3.2).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Vstream">
   <simpleContent>
      <extension base="xsd:base64Binary">
         <attribute name="version" type="ST_Clsid"/>
      </extension>
   </simpleContent>
</complexType>
```

7.4.3 Simple Types

This is the complete list of simple types in the http://schemas.openxmlformats.org/officeDocument/2006/docPropsVTypes namespace.

7.4.3.1 ST_ArrayBaseType (Array Base Type Simple Type)

The ST_ArrayBaseType simple type defines the allowed values for an array's baseType attribute as: variant, i1, i2, i4, int, ui1,ui2, ui4, uint, r4, r8, decimal, bstr, date, bool, cy, and error.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
bool (Boolean Base Type)	Specifies that the variant type for the contents of a array shall be `bool`.
bstr (Basic String Base Type)	Specifies that the variant type for the contents of a array shall be `bstr`.
cy (Curency Base Type)	Specifies that the variant type for the contents of a array shall be `cy`.
date (Date and Time Base Type)	Specifies that the variant type for the contents of a array shall be `date`.
decimal (Decimal Base Type)	Specifies that the variant type for the contents of a array shall be `decimal`.
error (Error Status Code Base Type)	Specifies that the variant type for the contents of a array shall be `error`.
i1 (1-Byte Signed Integer Base Type)	Specifies that the variant type for the contents of a array shall be `i1`.

Enumeration Value	Description
i2 (2-Byte Signed Integer Base Type)	Specifies that the variant type for the contents of a array shall be `i2`.
i4 (4-Byte Signed Integer Base Type)	Specifies that the variant type for the contents of a array shall be `i4`.
int (Integer Base Type)	Specifies that the variant type for the contents of a array shall be `int`.
r4 (4-Byte Real Number Base Type)	Specifies that the variant type for the contents of a array shall be `r4`.
r8 (8-Byte Real Number Base Type)	Specifies that the variant type for the contents of a array shall be `r8`.
ui1 (1-Byte Unsigned Integer Base Type)	Specifies that the variant type for the contents of a array shall be `ui1`.
ui2 (2-Byte Unsigned Integer Base Type)	Specifies that the variant type for the contents of a array shall be `ui2`.
ui4 (4-Byte Unsigned Integer Base Type)	Specifies that the variant type for the contents of a array shall be `ui4`.
uint (Unsigned Integer Base Type)	Specifies that the variant type for the contents of a array shall be `uint`.
variant (Variant Base Type)	Specifies that the variant type for the contents of a array shall be `variant`.

Referenced By
array@baseType (§7.4.2.1)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_ArrayBaseType">
   <restriction base="xsd:string">
      <enumeration value="variant"/>
      <enumeration value="i1"/>
      <enumeration value="i2"/>
      <enumeration value="i4"/>
      <enumeration value="int"/>
      <enumeration value="ui1"/>
      <enumeration value="ui2"/>
      <enumeration value="ui4"/>
      <enumeration value="uint"/>
      <enumeration value="r4"/>
      <enumeration value="r8"/>
      <enumeration value="decimal"/>
      <enumeration value="bstr"/>
      <enumeration value="date"/>
      <enumeration value="bool"/>
      <enumeration value="cy"/>
      <enumeration value="error"/>
   </restriction>
</simpleType>
```

7.4.3.2 ST_Clsid (Class ID Simple Type)

The ST_Clsid simple type specifies the type for the clsid element. The allowed values must be Globally Unique Identifiers with format: {HHHHHHHH-HHHH-HHHH-HHHH-HHHHHHHH}.

This simple type's contents are a restriction of the XML Schema string datatype.

This simple type also specifies the following restrictions:

- This simple type's contents must match the following regular expression pattern: \s*\{[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}\}\s*.

Referenced By
clsid (§7.4.2.6); property@fmtid (§7.3.2.2)

7.4.3.3 ST_Cy (Currency Simple Type)

The ST_Cy simple type defines the cy element as a currency variant type with exactly four digits after the decimal point.

This simple type's contents are a restriction of the XML Schema string datatype.

This simple type also specifies the following restrictions:

- This simple type's contents must match the following regular expression pattern: \s*[0-9]*\.[0-9]{4}\s*.

Referenced By

Referenced By
cy (§7.4.2.7)

7.4.3.4 ST_Error (Error Status Code Simple Type)

The ST_Error simple type defines a 32-bit error status code variant type of the form 0xHHHHHHHH. Each H represents a hexadecimal.

This simple type's contents are a restriction of the XML Schema string datatype.

This simple type also specifies the following restrictions:

- This simple type's contents must match the following regular expression pattern: \s*0x[0-9A-Za-z]{8}\s*.

Referenced By
error (§7.4.2.11)

7.4.3.5 ST_Format (Format Simple Type)

The ST_Format simple type defines the format attribute of the cf element. The format attribute can be the following values: -3, -2, 1, 0, and any positive integer.

This simple type's contents are a restriction of the XML Schema string datatype.

This simple type also specifies the following restrictions:

- This simple type's contents must match the following regular expression pattern: \-1.
- This simple type's contents must match the following regular expression pattern: \-2.
- This simple type's contents must match the following regular expression pattern: \-3.
- This simple type's contents must match the following regular expression pattern: [1-9]+.
- This simple type's contents must match the following regular expression pattern: 0.

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Format">
    <restriction base="xsd:string">
        <pattern value="\-1"/>
        <pattern value="\-2"/>
        <pattern value="\-3"/>
        <pattern value="[1-9]+"/>
        <pattern value="0"/>
    </restriction>
</simpleType>
```

7.4.3.6 ST_VectorBaseType (Vector Base Type Simple Type)

The ST_VectorBaseType simple type defines the allowed values for a vector's baseType attribute as: variant, i1, i2, i4, i8, ui1, ui2, ui4, ui8, r4, r8, lpstr, lpwstr, bstr, date, filetime, bool, cy, error, clsid, and cf.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
bool (Boolean Base Type)	Specifies that the variant type for the contents of a vector shall be `bool`.
bstr (Basic String Base Type)	Specifies that the variant type for the contents of a vector shall be `bstr`.
cf (Clipboard Data Base Type)	Specifies that the variant type for the contents of a vector shall be `cf`.
clsid (Class ID Base Type)	Specifies that the variant type for the contents of a vector shall be `clsid`.
cy (Currency Base Type)	Specifies that the variant type for the contents of a vector shall be `cy`.
date (Date and Time Base Type)	Specifies that the variant type for the contents of a vector shall be `date`.
error (Error Status Code Base Type)	Specifies that the variant type for the contents of a vector shall be `error`.
filetime (File Time Base Type)	Specifies that the variant type for the contents of a vector shall be `filetime`.
i1 (Vector Base Type Enumeration Value)	Specifies that the variant type for the contents of a vector shall be `i1`.
i2 (2-Byte Signed Integer Base Type)	Specifies that the variant type for the contents of a vector shall be `i2`.
i4 (4-Byte Signed Integer Base Type)	Specifies that the variant type for the contents of a vector shall be `i4`.
i8 (8-Byte Signed Integer Base Type)	Specifies that the variant type for the contents of a vector shall be `i8`.
lpstr (LPSTR Base Type)	Specifies that the variant type for the contents of a vector shall be `lpstr`.
lpwstr (LPWSTR Base Type)	Specifies that the variant type for the contents of a vector shall be `lpwstr`.
r4 (4-Byte Real Number Base Type)	Specifies that the variant type for the contents of a vector shall be `r4`.
r8 (8-Byte Real Number Base Type)	Specifies that the variant type for the contents of a vector shall be `r8`.
ui1 (1-Byte Unsigned Integer Base Type)	Specifies that the variant type for the contents of a vector shall be `ui1`.
ui2 (2-Byte Unisigned Integer Base Type)	Specifies that the variant type for the contents of a vector shall be `ui2`.
ui4 (4-Byte Unsigned Integer Base Type)	Specifies that the variant type for the contents of a vector shall be `ui4`.

Enumeration Value	Description
ui8 (8-Byte Unsigned Integer Base Type)	Specifies that the variant type for the contents of a vector shall be `ui8`.
variant (Variant Base Type)	Specifies that the variant type for the contents of a vector shall be `variant`.

Referenced By
vector@baseType (§7.4.2.34)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_VectorBaseType">
   <restriction base="xsd:string">
      <enumeration value="variant"/>
      <enumeration value="i1"/>
      <enumeration value="i2"/>
      <enumeration value="i4"/>
      <enumeration value="i8"/>
      <enumeration value="ui1"/>
      <enumeration value="ui2"/>
      <enumeration value="ui4"/>
      <enumeration value="ui8"/>
      <enumeration value="r4"/>
      <enumeration value="r8"/>
      <enumeration value="lpstr"/>
      <enumeration value="lpwstr"/>
      <enumeration value="bstr"/>
      <enumeration value="date"/>
      <enumeration value="filetime"/>
      <enumeration value="bool"/>
      <enumeration value="cy"/>
      <enumeration value="error"/>
      <enumeration value="clsid"/>
      <enumeration value="cf"/>
   </restriction>
</simpleType>
```

7.5 Custom XML Data Properties

This namespace defines the set of properties that may be associated with one or more custom XML parts within an Office Open XML document. A *custom XML part* is a part within an Office Open XML document, that contains arbitrary custom XML markup not necessarily defined by this Office Open XML Standard, and which is kept independent from the presentation-specific markup within the package.

[*Rationale*: It is often necessary to include custom XML semantics with an Office Open XML document, to store a complex set of properties (e.g., a document management system's metadata) along with the file. This mechanism allows this custom XML to be stored in the document in a way that is independent of the type of

document and separate from the presentation markup—only the custom XML is stored in this part. *end rationale*]

The properties that can be applied to a custom XML part are:

- A part ID
- (optionally) One or more associated custom XML schemas

7.5.1 Table of Contents

This subclause is informative.

End of informative text.

7.5.2 Elements

The following information describes the elements in this namespace:

7.5.2.1 datastoreItem (Custom XML Data Properties)

This element specifies the properties for a single custom XML part inside of an Office Open XML document. The set of properties specified within this element are attached to the custom XML part that specifies a relationship to this part.

[*Example*: Consider the following content for a custom XML part properties part:

```
<w:datastoreItem w:itemID="{A67AC88A-A164-4ADE-8889-8826CE44DE6E}">
  <w:schemaRefs>
    <w:schemaRef w:uri="http://www.contoso.com/exampleSchema" />
  </w:schemaRefs>
</w:datastoreItem>
```

The datastoreItem element contains the properties for the custom XML part that referenced it; specifically, a part ID of `A67AC88A-A164-4ADE-8889-8826CE44DE6E`, and a single XML Schema reference to a schema with a target namespace of `http://www.contoso.com/exampleSchema`. *end example*]

Parent Elements
Root element of Shared Custom XML Data Storage Properties part

Child Elements	Subclause
schemaRefs (Set of Associated XML Schemas)	§7.5.2.3

Attributes	Description

itemID (Custom XML Data ID)	Specifies a globally unique identifier (GUID) that uniquely identifies a single custom XML part within an Office Open XML document.
	
	Each itemID value shall be unique among all custom XML data parts in this document. If a document contains duplicate itemID values, then the first value should be persisted, and subsequent values should be reassigned.
	
	[*Example*: Consider the following content for a custom XML part properties part:
	
	`<w:datastoreItem w:itemID="{A67AC88A-A164-4ADE-8889-8826CE44DE6E}">`
	`…`
	`</w:datastoreItem>`
	
	The itemID attribute specifies that the ID associated with the parent custom XML part is `A67AC88A-A164-4ADE-8889-8826CE44DE6E`. *end example*]
	
	The possible values for this attribute are defined by the ST_Guid simple type (§7.5.3.1).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DatastoreItem">
    <sequence>
        <element name="schemaRefs" type="CT_DatastoreSchemaRefs" minOccurs="0"/>
    </sequence>
    <attribute name="itemID" type="ST_Guid" use="required"/>
</complexType>
```

7.5.2.2 schemaRef (Associated XML Schema)

This element specifies a single XML schema that is associated with the custom XML data part. This XML schema is identified using its target namespace, and may be located via any means available to an application processing the contents of this file.

If the custom XML part cannot be validated using the specified XML schema when it is opened, then this reference may be omitted when the file is subsequently saved.

[*Example*: Consider the following content for a custom XML part properties part:

```
<w:datastoreItem w:itemID="{A67AC88A-A164-4ADE-8889-8826CE44DE6E}">
  <w:schemaRefs>
    <w:schemaRef w:uri="http://www.contoso.com/exampleSchema" />
  </w:schemaRefs>
</w:datastoreItem>
```

The schemaRef element contains a single XML Schema reference to a schema with a target namespace of `http://www.contoso.com/exampleSchema`. Applications may then locate and utilize a schema for this namespace using any means available. *end example*]

Parent Elements
schemaRefs (§7.5.2.3)

Attributes	Description
uri (Target Namespace of Associated XML Schema)	Specifies the target namespace for the XML Schema associated with this schema reference. [*Example*: Consider the following content for a custom XML part properties part: … <w:schemaRef w:uri="http://www.contoso.com/schema1" /> <w:schemaRef w:uri="http://www.contoso.com/schema2" /> … The uri attribute specifies the target namespace of each XML schema reference: • http://www.contoso.com/schema1 • http://www.contoso.com/schema2 Applications may then locate and utilize a schema for these namespaces using any means available. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DatastoreSchemaRef">
   <attribute name="uri" type="xsd:string" use="required"/>
</complexType>
```

7.5.2.3 schemaRefs (Set of Associated XML Schemas)

This element specifies the set of XML schemas that are associated with the parent custom XML part. Any number of XML schemas may be referenced, and this collection of schemas shall then be used to validate the contents of the corresponding custom XML part. If this element is present, then the set of XML schemas provided within should be used to validate the contents of the corresponding custom XML part (including the

explicit presence of no child elements to specify that no custom XML schemas should be used even if one is present).

If this element is omitted, then applications may determine the set of XML schemas to be used to validate the contents of this part using any desired means.

[*Example*: Consider the following content for a custom XML part properties part:

```
<w:datastoreItem w:itemID="{A67AC88A-A164-4ADE-8889-8826CE44DE6E}">
  <w:schemaRefs>
    <w:schemaRef w:uri="http://www.contoso.com/exampleSchema" />
  </w:schemaRefs>
</w:datastoreItem>
```

The schemaRefs element contains the set of XML Schema references that may be used to validate the contents of this part. *end example*]

Parent Elements
datastoreItem (§7.5.2.1)

Child Elements	Subclause
schemaRef (Associated XML Schema)	§7.5.2.2

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_DatastoreSchemaRefs">
   <sequence>
      <element name="schemaRef" type="CT_DatastoreSchemaRef" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

7.5.3 Simple Types

This is the complete list of simple types in the http://schemas.openxmlformats.org/officeDocument/2006/customXml namespace.

7.5.3.1 ST_Guid (128-Bit GUID Value)

This simple type specifies that its values shall be a 128-bit globally unique identifier (GUID) value.

[*Example*: Consider the following content for a custom XML part properties part:

```
<w:datastoreItem w:itemID="{A67AC88A-A164-4ADE-8889-8826CE44DE6E}">
  …
</w:datastoreItem>
```

The itemID attribute specifies that the ID associated with the parent custom XML part is A67AC88A-A164-4ADE-8889-8826CE44DE6E. *end example*]

This simple type's contents are a restriction of the XML Schema token datatype.

This simple type also specifies the following restrictions:

- This simple type's contents must match the following regular expression pattern: \{[0-9A-F]{8}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{12}\}.

Referenced By
datastoreItem@itemID (§7.5.2.1)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Guid">
   <restriction base="xsd:token">
      <pattern value="\{[0-9A-F]{8}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{12}\}"/>
   </restriction>
</simpleType>
```

7.6 Bibliography

Within an Office Open XML document, it is possible to store an arbitrary amount of bibliographic data, the use of which can be determined by the application reading the content. This subclause defines the format and structure of that bibliographic data.

The outermost element of bibliographic data is Sources, which represents the collection of individual reference materials (Source) in the document.

7.6.1 Table of Contents

This subclause is informative.

End of informative text.

7.6.2 Elements

The following elements define the contents of the Bibliography schema:

7.6.2.1 AbbreviatedCaseNumber (Abbreviated Case Number)

This element describes the abbreviated form of a case number. Typically, this field is used in the Case source type.

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.2 AlbumTitle (Album Title)

This element specifies the title of an album. Typically, this field is used in the Sound recording source type. [*Example*:

```
<b:Source>
  <b:Tag>Bon96</b:Tag>
  <b:SourceType>SoundRecording</b:SourceType>
  <b:Author>
    <b:Performer>
      <b:NameList>
        <b:Person>
          <b:Last>Villaron</b:Last>
          <b:First>Shawn</b:First>
          <b:Middle>Alan</b:Middle>
        </b:Person>
      </b:NameList>
    </b:Performer>
  </b:Author>
  <b:Title>Title</b:Title>
  <b:Year>2004</b:Year>
  <b:City>London</b:City>
  <b:AlbumTitle>Album Title</b:AlbumTitle>
  <b:RefOrder>15</b:RefOrder>
  <b:Guid>{17722923-790D-47E7-BB5D-C5DC67FA83D6}</b:Guid>
  <b:LCID>0</b:LCID>
  <b:Comments>Comments</b:Comments>
</b:Source>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.3 Artist (Artist)

This element specifies the artist of the source. Typically, this field is used in the Art and Sound Recording source types. [*Example*:

```
<b:Author>
  <b:Artist>
    <b:NameList>
      <b:Person>
        <b:Last>Jones</b:Last>
        <b:First>Brian</b:First>
      </b:Person>
    </b:NameList>
  </b:Artist>
</b:Author>
```

end example]

Parent Elements
Author (§7.6.2.5)

Child Elements	Subclause
NameList (Name List)	§7.6.2.44

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NameType">
   <sequence>
      <element name="NameList" type="CT_NameListType" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

7.6.2.4 Author (Author)

This element specifies the author of the source. [*Example*:

```
<b:Author>
  <b:Author>
    <b:NameList>
      <b:Person>
        <b:Last>Jones</b:Last>
        <b:First>Brian</b:First>
      </b:Person>
    </b:NameList>
  </b:Author>
</b:Author>
```

end example]

Parent Elements

Parent Elements
Author (§7.6.2.5)

Child Elements	Subclause
Corporate (Corporate Author)	§7.6.2.18
NameList (Name List)	§7.6.2.44

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NameOrCorporateType">
   <sequence>
      <choice minOccurs="0" maxOccurs="1">
         <element name="NameList" type="CT_NameListType" minOccurs="1" maxOccurs="1"/>
         <element name="Corporate" minOccurs="1" maxOccurs="1" type="ST_String255"/>
      </choice>
   </sequence>
</complexType>
```

7.6.2.5 Author (Contributors List)

This element specifies the contributors to the source. [*Example*:

```
<b:Author>
  <b:Author>
    <b:NameList>
      <b:Person>
        <b:Last>Rothschiller</b:Last>
        <b:First>Chad</b:First>
      </b:Person>
    </b:NameList>
  </b:Author>
  <b:Editor>
    <b:NameList>
      <b:Person>
        <b:Last>Jaeschke</b:Last>
        <b:First>Rex</b:First>
      </b:Person>
    </b:NameList>
  </b:Editor>
```

```
    <b:Translator>
      <b:NameList>
        <b:Person>
          <b:Last>Davis</b:Last>
          <b:First>Tristan</b:First>
        </b:Person>
      </b:NameList>
    </b:Translator>
  </b:Author>
```

end example]

Parent Elements
Source (§7.6.2.59)

Child Elements	Subclause
Artist (Artist)	§7.6.2.3
Author (Author)	§7.6.2.4
BookAuthor (Book Author)	§7.6.2.6
Compiler (Compiler)	§7.6.2.14
Composer (Composer)	§7.6.2.15
Conductor (Conductor)	§7.6.2.16
Counsel (Counsel)	§7.6.2.19
Director (Director)	§7.6.2.25
Editor (Editor)	§7.6.2.28
Interviewee (Interviewee)	§7.6.2.33
Interviewer (Interviewer)	§7.6.2.34
Inventor (Inventor)	§7.6.2.35
Performer (Performer)	§7.6.2.48
ProducerName (Producer Name)	§7.6.2.51
Translator (Translator)	§7.6.2.69
Writer (Writer)	§7.6.2.74

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AuthorType">
   <sequence>
      <choice minOccurs="0" maxOccurs="unbounded">
         <element name="Artist" type="CT_NameType"/>
         <element name="Author" type="CT_NameOrCorporateType"/>
         <element name="BookAuthor" type="CT_NameType"/>
         <element name="Compiler" type="CT_NameType"/>
         <element name="Composer" type="CT_NameType"/>
         <element name="Conductor" type="CT_NameType"/>
         <element name="Counsel" type="CT_NameType"/>
         <element name="Director" type="CT_NameType"/>
         <element name="Editor" type="CT_NameType"/>
         <element name="Interviewee" type="CT_NameType"/>
         <element name="Interviewer" type="CT_NameType"/>
         <element name="Inventor" type="CT_NameType"/>
         <element name="Performer" type="CT_NameOrCorporateType"/>
         <element name="ProducerName" type="CT_NameType"/>
         <element name="Translator" type="CT_NameType"/>
         <element name="Writer" type="CT_NameType"/>
      </choice>
   </sequence>
</complexType>
```

7.6.2.6 BookAuthor (Book Author)

This element specifies the author of a book, when the primary author has authored the book section. For example, if person X writes a chapter in a book by person Y, person X is the Author and person Y is the BookAuthor. [*Example*:

```
<b:Author>
  <b:BookAuthor>
    <b:NameList>
      <b:Person>
        <b:Last>Rothschiller</b:Last>
        <b:First>Chad</b:First>
      </b:Person>
    </b:NameList>
  </b:BookAuthor>
</b:Author>
```

end example]

Parent Elements
Author (§7.6.2.5)

Child Elements	Subclause

Child Elements	Subclause
NameList (Name List)	§7.6.2.44

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NameType">
   <sequence>
      <element name="NameList" type="CT_NameListType" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

7.6.2.7 BookTitle (Book Title)

This element specifies the title of a book, when the source is a book section. In this case, the title of the book section is the primary title. For example, if X is the title of a chapter in a book entitled , X is the Title and Y is the BookTitle. [*Example*:

```
    …
    <b:Year>1992</b:Year>
    <b:City>Paris</b:City>
    <b:Publisher>Publisher</b:Publisher>
    <b:Pages>51-84</b:Pages>
    <b:Comments>Comments</b:Comments>
    <b:BookTitle>Book Title</b:BookTitle>
  </b:Source>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.8 Broadcaster (Broadcaster)

This element specifies the broadcaster of a source. Typically, this field is used in the Interview source type. [*Example*:

```
    …
    <b:ProgramTitle>Program Title</b:ProgramTitle>
    <b:Broadcaster>Broadcaster</b:Broadcaster>
    <b:Station>Station</b:Station>
    <b:RefOrder>1</b:RefOrder>
    <b:Title>Title (Interview)</b:Title>
    <b:BroadcastTitle>Broadcast Title</b:BroadcastTitle>
    <b:StateProvince>State or Province</b:StateProvince>
    <b:CountryRegion>Country or Region</b:CountryRegion>
  </b:Source>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.9 BroadcastTitle (Broadcast Title)

This element specifies the broadcast title of a source. Typically, this field is used in the Interview source type. [*Example*:

```
    …
    <b:ProgramTitle>Program Title</b:ProgramTitle>
    <b:Broadcaster>Broadcaster</b:Broadcaster>
    <b:Station>Station</b:Station>
    <b:RefOrder>1</b:RefOrder>
    <b:Title>Title (Interview)</b:Title>
    <b:BroadcastTitle>Broadcast Title</b:BroadcastTitle>
    <b:StateProvince>State/Province</b:StateProvince>
    <b:CountryRegion>Country/Region</b:CountryRegion>
</b:Source>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.10 CaseNumber (Case Number)

This element specifies the case number of a source. Typically, this field is used in the Case source type. [*Example*:

```
…
<b:Title>Title (Case)</b:Title>
<b:Year>Year</b:Year>
<b:City>Place Published</b:City>
<b:ShortTitle>Short Title</b:ShortTitle>
<b:CaseNumber>Case Number</b:CaseNumber>
<b:Court>Court</b:Court>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.11 ChapterNumber (Chapter Number)

This element specifies the number or index of the chapter being referenced. [*Example*:

```
…
<b:BookTitle>Title</b:BookTitle>
<b:Pages>23-65</b:Pages>
<b:Comments>Comments</b:Comments>
<b:ChapterNumber>6</b:ChapterNumber>
<b:RefOrder>1</b:RefOrder>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.12 City (City)

This element specifies the city in which the source was published, printed, or manufactured. [*Example*:

```
…
<b:Title>Title</b:Title>
<b:Year>1997</b:Year>
<b:City>London</b:City>
<b:Publisher>Publihser</b:Publisher>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.13 Comments (Comments)

This element specifies any additional comments about the source. The documentation style determines whether the comments appear in the bibliography. [*Example*:

```
…
<b:ShortTitle>Short Title</b:ShortTitle>
<b:Comments>Comments</b:Comments>
<b:RefOrder>2</b:RefOrder>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.14 Compiler (Compiler)

This element specifies the person who compiled the information in a source. [*Example*:

```
<b:Author>
  <b:Compiler>
    <b:NameList>
      <b:Person>
        <b:Last>Jones</b:Last>
        <b:First>Brian</b:First>
      </b:Person>
    </b:NameList>
  </b:Compiler>
</b:Author>
```

end example]

Parent Elements
Author (§7.6.2.5)

Child Elements	Subclause
NameList (Name List)	§7.6.2.44

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NameType">
   <sequence>
      <element name="NameList" type="CT_NameListType" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

7.6.2.15 Composer (Composer)

This element specifies the composer of a sound recording. [*Example*:

```
<b:Author>
  <b:Composer>
    <b:NameList>
      <b:Person>
        <b:Last>Davis</b:Last>
        <b:First>Tristan</b:First>
      </b:Person>
    </b:NameList>
  </b:Composer>
</b:Author>
```

end example]

Parent Elements
Author (§7.6.2.5)

Child Elements	Subclause
NameList (Name List)	§7.6.2.44

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NameType">
   <sequence>
      <element name="NameList" type="CT_NameListType" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

7.6.2.16 Conductor (Conductor)

This element specifies the conductor of a source. Typically, this field is used in the sound recording source type. [*Example*:

```
<b:Author>
  <b:Conductor>
    <b:NameList>
      <b:Person>
        <b:Last>Jones</b:Last>
        <b:First>Brian</b:First>
      </b:Person>
    </b:NameList>
  </b:Conductor>
</b:Author>
```

end example]

Parent Elements

Parent Elements
Author (§7.6.2.5)

Child Elements	Subclause
NameList (Name List)	§7.6.2.44

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NameType">
   <sequence>
      <element name="NameList" type="CT_NameListType" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

7.6.2.17 ConferenceName (Conference or Proceedings Name)

This element specifies the title of the proceedings from a conference. [*Example*:

```
…
<b:Comments>Comments</b:Comments>
<b:ConferenceName>Conference Name</b:ConferenceName>
<b:RefOrder>9</b:RefOrder>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.18 Corporate (Corporate Author)

This element specifies the corporate author, performer, or any field that can be a name. The element is used when an organization, rather than a person, is used. [*Example*:

```
<b:Author>
  <b:Author>
    <b:Corporate>Corporate Author</b:Corporate>
  </b:Author>
</b:Author>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements

Parent Elements
Author (§7.6.2.4); Performer (§7.6.2.48)

7.6.2.19 Counsel (Counsel)

This element specifies the counsel, attorney, or attorneys in a case.

Parent Elements
Author (§7.6.2.5)

Child Elements	Subclause
NameList (Name List)	§7.6.2.44

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NameType">
   <sequence>
      <element name="NameList" type="CT_NameListType" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

7.6.2.20 CountryRegion (Country or Region)

This element specifies the country or region of a source. [*Example*:

```
…
  <b:ProgramTitle>Program Title</b:ProgramTitle>
  <b:Broadcaster>Broadcaster</b:Broadcaster>
  <b:Station>Station</b:Station>
  <b:RefOrder>1</b:RefOrder>
  <b:Title>Title (Interview)</b:Title>
  <b:BroadcastTitle>Broadcast Title</b:BroadcastTitle>
  <b:StateProvince>State or Province</b:StateProvince>
  <b:CountryRegion>Country or Region</b:CountryRegion>
</b:Source>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.21 Court (Court)

This element specifies the court in which the case was presented. [*Example*:

```
…
<b:Year>1972</b:Year>
<b:CaseNumber>339 1018</b:CaseNumber>
<b:Court>Supreme Court</b:Court>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.22 Day (Day)

This element specifies the day on which a source was created or published. [*Example*:

```
…
<b:PeriodicalTitle>Periodical Title</b:PeriodicalTitle>
<b:Month>November</b:Month>
<b:Day>10</b:Day>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.23 DayAccessed (Day Accessed)

This element specifies the day of the month a source was accessed. [*Example*:

```
<b:MonthAccessed>October</b:MonthAccessed>
<b:DayAccessed>5</b:DayAccessed>
<b:YearAccessed>2000</b:YearAccessed>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.24 Department (Department)

This element specifies the department in which a source originated, or to which a source was submitted. Typically, this field is used in the Report source type, which includes theses and dissertations. [*Example*:

```
…
<b:Institution>Harvard University</b:Institution>
<b:ThesisType>Doctoral Dissertation</b:ThesisType>
<b:Department>Department of Mathematics</b:Department>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.25 Director (Director)

This element specifies the director of a source. Typically, this field is used in the Film source type. [*Example*:

```
<b:Author>
  <b:Director>
    <b:NameList>
      <b:Person>
        <b:Last>Jones</b:Last>
        <b:First>Brian</b:First>
      </b:Person>
    </b:NameList>
  </b:Director>
</b:Author>
```

end example]

Parent Elements
Author (§7.6.2.5)

Child Elements	Subclause
NameList (Name List)	§7.6.2.44

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NameType">
    <sequence>
        <element name="NameList" type="CT_NameListType" minOccurs="1" maxOccurs="1"/>
    </sequence>
</complexType>
```

7.6.2.26 Distributor (Distributor)

This element specifies the distributor of a source. Typically, this field is used in the Performance and Film source types. [*Example*:

```
…
<b:Distributor>Distributor</b:Distributor>
<b:Country>United States</b:Country>
<b:RefOrder>19</b:RefOrder>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.27 Edition (Editor)

This element specifies the edition of a source. [*Example*:

```
…
<b:Pages>1-34</b:Pages>
<b:Edition>Edition</b:Edition>
<b:Issue>Issue</b:Issue>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.28 Editor (Editor)

This element specifies the editor of a source. [*Example*:

```
<b:Author>
      <b:Editor>
        <b:NameList>
          <b:Person>
            <b:Last>Jaeschke</b:Last>
            <b:First>Rex</b:First>
          </b:Person>
        </b:NameList>
      </b:Editor>
</b:Author>
```

end example]

Parent Elements
Author (§7.6.2.5)

Child Elements	Subclause
NameList (Name List)	§7.6.2.44

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NameType">
   <sequence>
      <element name="NameList" type="CT_NameListType" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

7.6.2.29 First (Person's First, or Given, Name)

This element specifies a person's first name. [*Example*:

```
<b:Author>
  <b:Editor>
    <b:NameList>
      <b:Person>
        <b:Last>Jaeschke</b:Last>
        <b:First>Rex</b:First>
      </b:Person>
    </b:NameList>
  </b:Editor>
</b:Author>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Person (§7.6.2.50)

7.6.2.30 Guid (GUID)

This element specifies the GUID of a source. [*Example*:

```
…
  <b:RefOrder>2</b:RefOrder>
  <b:Guid>{EE06CBFE-1989-4533-A274-D81DFA436D79}</b:Guid>
  <b:LCID>0</b:LCID>
</b:Source>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.31 Institution (Institution)

This element specifies the institution of the source. Typically, this field is used in the Report source type, where it signifies the university or institute, and in the Art source type, where it signifies the museum or institution where the art is housed. [*Example*:

```
…
  <b:Institution>Harvard University</b:Institution>
  <b:ThesisType>Dissertation</b:ThesisType>
  <b:RefOrder>12</b:RefOrder>
  <b:Guid>{6CB80970-81D3-476D-90D5-5C9D64E77FAF}</b:Guid>
  <b:LCID>0</b:LCID>
</b:Source>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.32 InternetSiteTitle (Internet Site Title)

This element specifies the title of an internet site. Typically, this field is used in the Internet Site and Document from Internet Site source types. [*Example*:

```
…
<b:Title>Title</b:Title>
<b:InternetSiteTitle>Internet Site Title</b:InternetSiteTitle>
<b:Month>July</b:Month>
<b:Day>1</b:Day>
<b:Year>2001</b:Year>
<b:MonthAccessed>Sept.</b:MonthAccessed>
<b:DayAccessed>22</b:DayAccessed>
<b:YearAccessed>1999</b:YearAccessed>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.33 Interviewee (Interviewee)

This element specifies the person being interviewed. Typically, this field is used in the Interview source type.

[*Example*:

```
<b:Interviewee>
  <b:NameList>
    <b:Person>
      <b:Last>Rothschiller</b:Last>
      <b:First>Chad</b:First>
    </b:Person>
  </b:NameList>
</b:Interviewee>
```

end example]

Parent Elements
Author (§7.6.2.5)

Child Elements	Subclause
NameList (Name List)	§7.6.2.44

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NameType">
   <sequence>
      <element name="NameList" type="CT_NameListType" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

7.6.2.34 Interviewer (Interviewer)

This element specifies the person conducting an interview. Typically, this field is used in the Interview source type. [*Example*:

```
<b:Interviewer>
  <b:NameList>
    <b:Person>
      <b:Last>Davis</b:Last>
      <b:First>Tristan</b:First>
     </b:Person>
  </b:NameList>
</b:Interviewer>
```

end example]

Parent Elements
Author (§7.6.2.5)

Child Elements	Subclause
NameList (Name List)	§7.6.2.44

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NameType">
   <sequence>
      <element name="NameList" type="CT_NameListType" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

7.6.2.35 Inventor (Inventor)

This element specifies the inventor of a source. Typically, this field is used in the Patent source type. [*Example*:

```
<b:Author>
  <b:Inventor>
    <b:NameList>
      <b:Person>
        <b:Last>Jones</b:Last>
        <b:First>Brian</b:First>
      </b:Person>
    </b:NameList>
  </b:Inventor>
</b:Author>
```

end example]

Parent Elements
Author (§7.6.2.5)

Child Elements	Subclause
NameList (Name List)	§7.6.2.44

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NameType">
   <sequence>
      <element name="NameList" type="CT_NameListType" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

7.6.2.36 Issue (Issue)

This element specifies the issue of a source. Typically, this field is used in the Journal Article and Article in Periodical source types. [*Example*:

```
…
  <b:Edition>Edition</b:Edition>
  <b:Issue>Issue</b:Issue>
  <b:RefOrder>28</b:RefOrder>
</b:Source>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.37 JournalName (Journal Name)

This element specifies the name of the journal. Typically, this field is used in the Journal Article source type. [*Example*:

```
…
<b:Title>Article Title</b:Title>
<b:Year>2000</b:Year>
<b:ShortTitle>Short Title</b:ShortTitle>
<b:Volume>100</b:Volume>
<b:Comments>Comments</b:Comments>
<b:JournalName>Journal Name</b:JournalName>
<b:Pages>91-160</b:Pages>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.38 Last (Person's Last, or Family, Name)

This element specifies a person's last name. [*Example*:

```
<b:Author>
  <b:Editor>
    <b:NameList>
      <b:Person>
        <b:Last>Jaeschke</b:Last>
        <b:First>Tristan</b:First>
      </b:Person>
    </b:NameList>
  </b:Editor>
</b:Author>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Person (§7.6.2.50)

7.6.2.39 LCID (Locale ID)

This element specifies the locale ID of a source, representing the source's language. The set of locale IDs shall be as specified in §2.18.52. [*Example*:

```
    …
    <b:RefOrder>2</b:RefOrder>
    <b:Guid>{EE06CBFE-1989-4533-A274-D81DFA436D79}</b:Guid>
    <b:LCID>0</b:LCID>
  </b:Source>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.40 Medium (Medium)

This element specifies the medium on or in which a source was created. Typically, this field is used in the Electronic source, sound recording, and film source types. [*Example*:

```
  <b:Source>
    …
    <b:LCID>0</b:LCID>
    <b:Medium>DVD</b:Medium>
  </b:Source>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.41 Middle (Person's Middle, or Other, Name)

This element specifies a person's middle name. [*Example*:

```
<b:Author>
  <b:Editor>
    <b:NameList>
      <b:Person>
        <b:Last>Villaron</b:Last>
        <b:First>Shawn</b:First>
        <b:Middle>Alan</b:Middle>
      </b:Person>
    </b:NameList>
  </b:Editor>
</b:Author>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Person (§7.6.2.50)

7.6.2.42 Month (Month)

This element specifies the month in which a source was created or published. [*Example*:

```
…
<b:PeriodicalTitle>Time</b:PeriodicalTitle>
<b:Month>November</b:Month>
<b:Day>10</b:Day>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.43 MonthAccessed (Month Accessed)

This element specifies the month during which the source was accessed. [*Example*:

```
…
<b:MonthAccessed>October</b:MonthAccessed>
<b:DayAccessed>5</b:DayAccessed>
<b:YearAccessed>2000</b:YearAccessed>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.44 NameList (Name List)

This element specifies a list containing one or more names of a type of contributor to a source, such as a list of authors, editors, or translators. [*Example*:

```
<b:Author>
  <b:Author>
    <b:NameList>
      <b:Person>
        <b:Last>Davis</b:Last>
        <b:First>Tristan</b:First>
      </b:Person>
    </b:NameList>
  </b:Author>
</b:Author>
```

end example]

Parent Elements
Artist (§7.6.2.3); Author (§7.6.2.4); BookAuthor (§7.6.2.6); Compiler (§7.6.2.14); Composer (§7.6.2.15); Conductor (§7.6.2.16); Counsel (§7.6.2.19); Director (§7.6.2.25); Editor (§7.6.2.28); Interviewee (§7.6.2.33); Interviewer (§7.6.2.34); Inventor (§7.6.2.35); Performer (§7.6.2.48); ProducerName (§7.6.2.51); Translator (§7.6.2.69); Writer (§7.6.2.74)

Child Elements	Subclause
Person (Person)	§7.6.2.50

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NameListType">
   <sequence>
      <element name="Person" type="CT_PersonType" minOccurs="1" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

7.6.2.45 NumberVolumes (Number of Volumes)

This element specifies the number of volumes a source contains. [*Example*:

```
…
  <b:NumberVolumes>10</b:NumberVolumes>
  <b:Comments>Comments</b:Comments>
</b:Source>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.46 Pages (Pages)

This element specifies the page range being cited in a source. [*Example*:

```
…
<b:Title>Article Title</b:Title>
<b:Year>2000</b:Year>
<b:ShortTitle>Short Title</b:ShortTitle>
<b:Volume>100</b:Volume>
<b:Comments>Comments</b:Comments>
<b:JournalName>Journal Name</b:JournalName>
<b:Pages>91-160</b:Pages>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.47 PatentNumber (Patent Number)

This element specifies the patent number of a source. Typically, this field is used in the Patent source type. [*Example*:

```
<b:Source>
  …
  <b:PatentNumber>1,000,000</b:PatentNumber>
  <b:RefOrder>26</b:RefOrder>
  <b:Guid>{8295ABC5-2DFD-4FA7-A2A7-A748917C1755}</b:Guid>
  <b:LCID>0</b:LCID>
</b:Source>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.48 Performer (Performer)

This element specifies the performer. Typically, this field is used in the sound recording, performance, and film source types. [*Example*:

```
<b:Performer>
  <b:NameList>
    <b:Person>
      <b:Last>Rothschiller</b:Last>
      <b:First>Chad</b:First>
    </b:Person>
  </b:NameList>
</b:Performer>
```

end example]

Parent Elements
Author (§7.6.2.5)

Child Elements	Subclause
Corporate (Corporate Author)	§7.6.2.18
NameList (Name List)	§7.6.2.44

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NameOrCorporateType">
   <sequence>
      <choice minOccurs="0" maxOccurs="1">
         <element name="NameList" type="CT_NameListType" minOccurs="1" maxOccurs="1"/>
         <element name="Corporate" minOccurs="1" maxOccurs="1" type="ST_String255"/>
      </choice>
   </sequence>
</complexType>
```

7.6.2.49 PeriodicalTitle (Periodical Title)

This element specifies the title of a periodical. [*Example*:

```
…
<b:PeriodicalTitle>Periodical Title</b:PeriodicalTitle>
<b:Month>July</b:Month>
<b:Day>1</b:Day>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.50 Person (Person)

This element specifies a person who contributed to a source. [*Example*:

```
<b:Author>
  <b:NameList>
    <b:Person>
      <b:Last>Villaron</b:Last>
      <b:First>Shawn</b:First>
      <b:Middle>Alan</b:Middle>
    </b:Person>
  </b:NameList>
</b:Author>
```

end example]

Parent Elements
NameList (§7.6.2.44)

Child Elements	Subclause
First (Person's First, or Given, Name)	§7.6.2.29
Last (Person's Last, or Family, Name)	§7.6.2.38
Middle (Person's Middle, or Other, Name)	§7.6.2.41

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_PersonType">
   <sequence>
      <element name="Last" type="ST_String255" minOccurs="0" maxOccurs="unbounded"/>
      <element name="First" type="ST_String255" minOccurs="0" maxOccurs="unbounded"/>
      <element name="Middle" type="ST_String255" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

7.6.2.51 ProducerName (Producer Name)

This element specifies the person who produced a source. Typically, this field is used in the Internet site, Doc from internet site, Electronic source, Sound recording, Performance, and Film source types. [*Example*:

```
<b:ProducerName>
  <b:NameList>
    <b:Person>
      <b:Last>Rothschiller</b:Last>
      <b:First>Chad</b:First>
    </b:Person>
  </b:NameList>
</b:ProducerName>
```

end example]

Parent Elements
Author (§7.6.2.5)

Child Elements	Subclause
NameList (Name List)	§7.6.2.44

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NameType">
   <sequence>
      <element name="NameList" type="CT_NameListType" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

7.6.2.52 ProductionCompany (Production Company)

This element specifies the company that produced a source. Typically, this field is used in the Internet site, Document from internet site, Electronic source, Sound recording, Performance, and Film source types. [*Example*:

```
…
  <b:City>Chicago</b:City>
  <b:ProductionCompany>Production Company</b:ProductionCompany>
  <b:Medium>CD</b:Medium>
  <b:RefOrder>16</b:RefOrder>
</b:Source>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements

Parent Elements
Source (§7.6.2.59)

7.6.2.53 PublicationTitle (Publication Title)

This element specifies the title of the publication that contains the source. Typically, this field is used in the Electronic Source source type. [*Example*:

```
…
<b:Volume>Volume</b:Volume>
<b:PublicationTitle>Publication Title</b:PublicationTitle>
<b:Month>June</b:Month>
<b:Day>2</b:Day>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.54 Publisher (Publisher)

This element specifies the publisher of a source.

```
…
<b:City>London</b:City>
<b:Publisher>Publisher</b:Publisher>
<b:ShortTitle>Short Title</b:ShortTitle>
<b:Volume>Volume</b:Volume>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.55 RecordingNumber (Recording Number)

This element specifies the recording number of a source. Typically, this field is used in the sound recording source type.

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements

Parent Elements
Source (§7.6.2.59)

7.6.2.56 RefOrder (Reference Order)

This element specifies the reference order of a source. [*Example*:

```
…
  <b:City>Chicago</b:City>
  <b:ProductionCompany>Production Company</b:ProductionCompany>
  <b:Medium>CD</b:Medium>
  <b:RefOrder>16</b:RefOrder>
</b:Source>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.57 Reporter (Reporter)

This element specifies the reporter of a source. Typically, this field is used in the Case source type. [*Example*:

```
…
  <b:Reporter>Reporter</b:Reporter>
  <b:RefOrder>27</b:RefOrder>
  <b:Guid>{CE314AB7-E824-4D10-B295-044C68EBED27}</b:Guid>
  <b:LCID>0</b:LCID>
</b:Source>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.58 ShortTitle (Short Title)

This element specifies the short title of a source. [*Example*:

```
…
<b:City>London</b:City>
<b:Publisher>Publisher</b:Publisher>
<b:ShortTitle>Short Title</b:ShortTitle>
<b:Volume>Volume</b:Volume>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.59 Source (Source)

This element specifies the bibliography entry for a source or reference work. [*Example*:

```
…
  <b:NumberVolumes>10</b:NumberVolumes>
  <b:Comments>Comments</b:Comments>
</b:Source>
```

end example]

Parent Elements
Sources (§7.6.2.60)

Child Elements	Subclause
AbbreviatedCaseNumber (Abbreviated Case Number)	§7.6.2.1
AlbumTitle (Album Title)	§7.6.2.2
Author (Contributors List)	§7.6.2.5
BookTitle (Book Title)	§7.6.2.7
Broadcaster (Broadcaster)	§7.6.2.8
BroadcastTitle (Broadcast Title)	§7.6.2.9
CaseNumber (Case Number)	§7.6.2.10
ChapterNumber (Chapter Number)	§7.6.2.11
City (City)	§7.6.2.12
Comments (Comments)	§7.6.2.13
ConferenceName (Conference or Proceedings Name)	§7.6.2.17
CountryRegion (Country or Region)	§7.6.2.20

Child Elements	Subclause
Court (Court)	§7.6.2.21
Day (Day)	§7.6.2.22
DayAccessed (Day Accessed)	§7.6.2.23
Department (Department)	§7.6.2.24
Distributor (Distributor)	§7.6.2.26
Edition (Editor)	§7.6.2.27
Guid (GUID)	§7.6.2.30
Institution (Institution)	§7.6.2.31
InternetSiteTitle (Internet Site Title)	§7.6.2.32
Issue (Issue)	§7.6.2.36
JournalName (Journal Name)	§7.6.2.37
LCID (Locale ID)	§7.6.2.39
Medium (Medium)	§7.6.2.40
Month (Month)	§7.6.2.42
MonthAccessed (Month Accessed)	§7.6.2.43
NumberVolumes (Number of Volumes)	§7.6.2.45
Pages (Pages)	§7.6.2.46
PatentNumber (Patent Number)	§7.6.2.47
PeriodicalTitle (Periodical Title)	§7.6.2.49
ProductionCompany (Production Company)	§7.6.2.52
PublicationTitle (Publication Title)	§7.6.2.53
Publisher (Publisher)	§7.6.2.54
RecordingNumber (Recording Number)	§7.6.2.55
RefOrder (Reference Order)	§7.6.2.56
Reporter (Reporter)	§7.6.2.57
ShortTitle (Short Title)	§7.6.2.58
SourceType (Source Type)	§7.6.2.61
StandardNumber (Standard Number)	§7.6.2.62
StateProvince (State or Province)	§7.6.2.63
Station (Station)	§7.6.2.64
Tag (Tag)	§7.6.2.65
Theater (Theater)	§7.6.2.66
ThesisType (Thesis Type)	§7.6.2.67
Title (Title)	§7.6.2.68

Child Elements	Subclause
Type (Type)	§7.6.2.70
URL (URL)	§7.6.2.71
Version (Version)	§7.6.2.72
Volume (Volume)	§7.6.2.73
Year (Year)	§7.6.2.75
YearAccessed (Year Accessed)	§7.6.2.76

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SourceType">
    <sequence>
        <choice minOccurs="0" maxOccurs="unbounded">
            <element name="AbbreviatedCaseNumber" type="ST_String255"/>
            <element name="AlbumTitle" type="ST_String255"/>
            <element name="Author" type="CT_AuthorType"/>
            <element name="BookTitle" type="ST_String255"/>
            <element name="Broadcaster" type="ST_String255"/>
            <element name="BroadcastTitle" type="ST_String255"/>
            <element name="CaseNumber" type="ST_String255"/>
            <element name="ChapterNumber" type="ST_String255"/>
            <element name="City" type="ST_String255"/>
            <element name="Comments" type="ST_String255"/>
            <element name="ConferenceName" type="ST_String255"/>
            <element name="CountryRegion" type="ST_String255"/>
            <element name="Court" type="ST_String255"/>
            <element name="Day" type="ST_String255"/>
            <element name="DayAccessed" type="ST_String255"/>
            <element name="Department" type="ST_String255"/>
            <element name="Distributor" type="ST_String255"/>
            <element name="Edition" type="ST_String255"/>
            <element name="Guid" type="ST_String255"/>
            <element name="Institution" type="ST_String255"/>
            <element name="InternetSiteTitle" type="ST_String255"/>
            <element name="Issue" type="ST_String255"/>
            <element name="JournalName" type="ST_String255"/>
            <element name="LCID" type="ST_String255"/>
            <element name="Medium" type="ST_String255"/>
            <element name="Month" type="ST_String255"/>
            <element name="MonthAccessed" type="ST_String255"/>
            <element name="NumberVolumes" type="ST_String255"/>
            <element name="Pages" type="ST_String255"/>
            <element name="PatentNumber" type="ST_String255"/>
            <element name="PeriodicalTitle" type="ST_String255"/>
            <element name="ProductionCompany" type="ST_String255"/>
            <element name="PublicationTitle" type="ST_String255"/>
            <element name="Publisher" type="ST_String255"/>
            <element name="RecordingNumber" type="ST_String255"/>
            <element name="RefOrder" type="ST_String255"/>
            <element name="Reporter" type="ST_String255"/>
            <element name="SourceType" type="ST_SourceType"/>
            <element name="ShortTitle" type="ST_String255"/>
            <element name="StandardNumber" type="ST_String255"/>
            <element name="StateProvince" type="ST_String255"/>
            <element name="Station" type="ST_String255"/>
            <element name="Tag" type="ST_String255"/>
            <element name="Theater" type="ST_String255"/>
            <element name="ThesisType" type="ST_String255"/>
            <element name="Title" type="ST_String255"/>
            <element name="Type" type="ST_String255"/>
            <element name="URL" type="ST_String255"/>
            <element name="Version" type="ST_String255"/>
```

```
          <element name="Volume" type="ST_String255"/>
          <element name="Year" type="ST_String255"/>
          <element name="YearAccessed" type="ST_String255"/>
       </choice>
    </sequence>
</complexType>
```

7.6.2.60 Sources (Sources)

This element specifies the sources in a collection.

Parent Elements
Root element of Shared Bibliography part

Child Elements	Subclause
Source (Source)	§7.6.2.59

Attributes	Description
SelectedStyle (Selected Style)	Specifies the filename of a file which may be used to format the bibliographies and citations within this document. If this file is of an unknown form or cannot be located, then the other attributes on this element may be used to determine the format to use. [*Example*: `<b:Sources SelectedStyle="\APA.XSL" StyleName="APA" URI="http://schemas.openxmlformats.org/bibliographicStyle/APA">` *end example*] The possible values for this attribute are defined by the ST_String255 simple type (§7.6.3.2).
StyleName (Documentation Style Name)	Specifies the name of the documentation style in which the bibliography and citations are formatted. The following values shall be well-defined: **Value** / **Reference** APA — American Psychological Association. http://www.apa.org/. Publication Manual of the American Psychological Association, Fifth Edition. Chicago — Chicago Manual of Style, 15th Edition GB7714: GB7714-1987, Standardization Administration of China,

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>1987-05-05 (http://www.sac.gov.cn)</td></tr>
<tr><td>GOST - Name Soft</td><td>ГОСТ 7.1-2003 (GOST 7.1-2003) - The Federal Agency of the Russian Federation on Technical Regulating and Metrology - published by ИПК Издательство стандартов 2004</td></tr>
<tr><td>GOST - Title Sort</td><td>ГОСТ 7.1-2003 (GOST 7.1-2003) - The Federal Agency of the Russian Federation on Technical Regulating and Metrology - published by ИПК Издательство стандартов 2004</td></tr>
<tr><td>ISO 690 - First Element and Date</td><td>ISO 690-1987(E)-International Organization for Standardization-Second Edition 1987-08-15 (http://www.iso.org)
ISO 690-2:1997(E)-International Organization for Standardization-First Edition 1997-11-15 (http://www.iso.org)</td></tr>
<tr><td>ISO 690 - Numerical Reference</td><td>ISO 690-1987(E)-International Organization for Standardization-Second Edition 1987-08-15 (http://www.iso.org)
ISO 690-2:1997(E)-International Organization for Standardization-First Edition 1997-11-15 (http://www.iso.org)</td></tr>
<tr><td>MLA</td><td>Modern Language Association. http://www.mla.org/. MLA Handbook for Writers of Research Papers, Sixth Edition.</td></tr>
<tr><td>SIST02</td><td>Standard for Information of Science and Technology by Japan Science and Technology Agency, 2003(http://www.jst.go.jp/SIST/handbook/sist02sup/index.htm.</td></tr>
<tr><td>Turabian</td><td>A Manual for Writers of Term Papers, Theses, and Dissertations (Chicago Guides to Writing, Editing, and Publishing), by Kate L. Turabian, 1996.</td></tr>
<tr><td>Any other value</td><td>Implementation-defined.</td></tr>
</table>

[Example:

<code><b:Sources SelectedStyle="\APA.XSL" StyleName="APA" URI="123"></code>

end example]

The possible values for this attribute are defined by the ST_String255 simple type (§7.6.3.2).
</td></tr>
<tr><td>URI (Uniform Resource Identifier)</td><td>Specifies a URI or unique identifier with which a documentation style is associated; may be used to uniquely identify versions of styles that share a StyleName.

[Example:

<code><b:Sources SelectedStyle="\APA.XSL" StyleName="APA" URI="123"></code>

end example]</td></tr>
</table>

Attributes	Description
	The possible values for this attribute are defined by the ST_String255 simple type (§7.6.3.2).

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Sources">
   <sequence>
      <element name="Source" type="CT_SourceType" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="SelectedStyle" type="ST_String255"/>
   <attribute name="StyleName" type="ST_String255"/>
   <attribute name="URI" type="ST_String255"/>
</complexType>
```

7.6.2.61 SourceType (Source Type)

This element specifies the type of source being cited.

The possible values for this element are defined by the ST_SourceType simple type (§7.6.3.1).

Parent Elements
Source (§7.6.2.59)

7.6.2.62 StandardNumber (Standard Number)

This element specifies the standard number, such as ISBN or ISSN, of a source. [*Example*:

```
…
  <b:NumberVolumes>10</b:NumberVolumes>
  <b:StandardNumber>ISBN or ISSN</b:StandardNumber>
  <b:Comments>Comments</b:Comments>
</b:Source>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.63 StateProvince (State or Province)

This element specifies the state or province in which a source was created or published. [*Example*:

```
…
  <b:ProgramTitle>Program Title</b:ProgramTitle>
  <b:Broadcaster>Broadcaster</b:Broadcaster>
  <b:Station>Station</b:Station>
  <b:RefOrder>1</b:RefOrder>
```

```
    <b:Title>Title (Interview)</b:Title>
    <b:BroadcastTitle>Broadcast Title</b:BroadcastTitle>
    <b:StateProvince>State/Province</b:StateProvince>
    <b:CountryRegion>Country/Region</b:CountryRegion>
  </b:Source>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.64 Station (Station)

This element specifies the station on which an interview was broadcasted. Typically, this field is used in the Interview source type. [*Example*:

```
  …
  <b:Month>November</b:Month>
  <b:Day>18</b:Day>
  <b:Broadcaster>ABC</b:Broadcaster>
  <b:Station>WABC</b:Station>
  …
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.65 Tag (Tag)

This element specifies the tag name of a source. [*Example*:

```
  <b:Source>
    <b:Tag>New01</b:Tag>
    …
  </b:Source>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.66 Theater (Theater)

This element specifies the theater in which a source was performed or viewed. Typically, this field is used in the Performer source type. [*Example*:

```
…
<b:Theater>Theater Name</b:Theater>
<b:Month>October</b:Month>
<b:Day>25</b:Day>
<b:RefOrder>19</b:RefOrder>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.67 ThesisType (Thesis Type)

This element specifies the type of report being cited, such as Thesis, Dissertation, or Book Report. Typically, this field is used in the Report source type. [*Example*:

```
…
<b:Institution>Harvard University</b:Institution>
<b:ThesisType>Doctoral Dissertation</b:ThesisType>
<b:Department>Department of Mathematics</b:Department>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.68 Title (Title)

This element specifies the title of a source. [*Example*:

```
…
</b:Author>
<b:Title>Title</b:Title>
<b:Year>2005</b:Year>
<b:City>Seattle</b:City>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.69 Translator (Translator)

This element specifies the translator of a source. [*Example*:

```
<b:Author>
  <b:Translator>
    <b:NameList>
      <b:Person>
        <b:Last>Davis</b:Last>
        <b:First>Tristan</b:First>
      </b:Person>
    </b:NameList>
  </b:Translator>
</b:Author>
```

end example]

Parent Elements
Author (§7.6.2.5)

Child Elements	Subclause
NameList (Name List)	§7.6.2.44

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NameType">
   <sequence>
      <element name="NameList" type="CT_NameListType" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

7.6.2.70 Type (Type)

This element specifies the type of patent. Typically, this field is used in the Patent source type.

[*Example*:

```
    <b:Type>Patent Type</b:Type>
    <b:Guid>{8295ABC5-2DFD-4FA7-A2A7-A748917C1755}</b:Guid>
    <b:LCID>0</b:LCID>
  </Source>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.71 URL (URL)

This element specifies the URL of the source. Typically, this field is used in the Internet Site and Document from Internet Site source types. [*Example*:

```
…
<b:MonthAccessed>September</b:MonthAccessed>
<b:DayAccessed>1</b:DayAccessed>
<b:YearAccessed>1998</b:YearAccessed>
<b:URL>URL</b:URL>
<b:RefOrder>29</b:RefOrder>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.72 Version (Version)

This element specifies the version of the source. Typically, this field is used in the Internet Site and Document from Internet Site source types. [*Example*:

```
…
    <b:Version>3.0</b:Version>
    <b:RefOrder>31</b:RefOrder>
    <b:Guid>{F06D8D48-7FD7-4515-88E9-EC70AB9BE792}</b:Guid>
    <b:LCID>0</b:LCID>
  </b:Source>
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.73 Volume (Volume)

This element specifies the volume of the source. [*Example*:

```
…
<b:Title>Article Title</b:Title>
<b:Year>2000</b:Year>
<b:ShortTitle>Short Title</b:ShortTitle>
<b:Volume>100</b:Volume>
<b:Comments>Comments</b:Comments>
<b:JournalName>Journal Name</b:JournalName>
<b:Pages>91-160</b:Pages>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.74 Writer (Writer)

This element specifies the writer of the source. Typically, this field is used in the Performance and Film source types. [*Example*:

```
<b:Author>
  <b:Writer>
    <b:NameList>
      <b:Person>
        <b:Last>Jones</b:Last>
        <b:First>Brian</b:First>
      </b:Person>
    </b:NameList>
  </b:Writer>
</b:Author>
```

end example]

Parent Elements
Author (§7.6.2.5)

Child Elements	Subclause
NameList (Name List)	§7.6.2.44

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_NameType">
   <sequence>
      <element name="NameList" type="CT_NameListType" minOccurs="1" maxOccurs="1"/>
   </sequence>
</complexType>
```

7.6.2.75 Year (Year)

This element specifies the year in which a source was created or published.

[*Example*:

```
…
<b:Title>Title</b:Title>
<b:InternetSiteTitle>Internet Site Title</b:InternetSiteTitle>
<b:Month>July</b:Month>
<b:Day>1</b:Day>
<b:Year>2001</b:Year>
<b:MonthAccessed>Sept.</b:MonthAccessed>
<b:DayAccessed>22</b:DayAccessed>
<b:YearAccessed>1999</b:YearAccessed>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.2.76 YearAccessed (Year Accessed)

This element specifies the month during which the source was accessed. [*Example*:

```
…
<b:Title>Title</b:Title>
<b:InternetSiteTitle>Internet Site Title</b:InternetSiteTitle>
<b:Month>July</b:Month>
<b:Day>1</b:Day>
<b:Year>2001</b:Year>
<b:MonthAccessed>Sept.</b:MonthAccessed>
<b:DayAccessed>22</b:DayAccessed>
<b:YearAccessed>1999</b:YearAccessed>
…
```

end example]

The possible values for this element are defined by the ST_String255 simple type (§7.6.3.2).

Parent Elements
Source (§7.6.2.59)

7.6.3 Simple Types

This is the complete list of simple types in the http://schemas.openxmlformats.org/officeDocument/2006/bibliography namespace.

7.6.3.1 ST_SourceType (Bibliographic Data Source Types)

This simple type specifies the possible types of sources that may be used within bibliographic data in an Office Open XML document.

[*Note:* The Office Open XML formats support a collection of predefined source types based on the categories most commonly used in various citation and bibliography style guidelines . The set of predefined source types can be extended as needed. The recommended approach for extending this set is to use the `Misc` type, and then leverage the methods described in Part 5 of this standard for extending the format with new attributes or elements. *end note.*]

This simple type's contents are a restriction of the ST_String255 simple type (§7.6.3.2).

The following are possible enumeration values for this type:

Enumeration Value	Description
Art (Art)	Art
ArticleInAPeriodical (Article in a Periodical)	Article in a Periodical
Book (Book)	Book
BookSection (Book Section)	Book Section
Case (Case)	Case
ConferenceProceedings (Conference Proceedings)	Conference Proceedings

Enumeration Value	Description
DocumentFromInternetSite (Document from Internet Site)	Document from Internet Site
ElectronicSource (Electronic Source)	Electronic Source
Film (Film)	Film
InternetSite (Internet Site)	Internet Site
Interview (Interview)	Interview
JournalArticle (Journal Article)	Journal Article
Misc (Miscellaneous)	Miscellaneous
Patent (Patent)	Patent
Performance (Performance)	Performance
Report (Reporter)	Report
SoundRecording (Sound Recording)	Sound Recording

Referenced By
SourceType (§7.6.2.61)

7.6.3.2 ST_String255 (String Value)

String whose maximum length is 255 characters.

This simple type's contents are a restriction of the XML Schema string datatype.

This simple type also specifies the following restrictions:

- This simple type's contents have a minimum length of 0 characters.
- This simple type's contents have a maximum length of 255 characters.

Referenced By
AbbreviatedCaseNumber (§7.6.2.1); AlbumTitle (§7.6.2.2); BookTitle (§7.6.2.7); Broadcaster (§7.6.2.8); BroadcastTitle (§7.6.2.9); CaseNumber (§7.6.2.10); ChapterNumber (§7.6.2.11); City (§7.6.2.12); Comments (§7.6.2.13); ConferenceName (§7.6.2.17); Corporate (§7.6.2.18); CountryRegion (§7.6.2.20); Court (§7.6.2.21); Day (§7.6.2.22); DayAccessed (§7.6.2.23); Department (§7.6.2.24); Distributor (§7.6.2.26); Edition (§7.6.2.27); First (§7.6.2.29); Guid (§7.6.2.30); Institution (§7.6.2.31); InternetSiteTitle (§7.6.2.32); Issue (§7.6.2.36); JournalName (§7.6.2.37); Last (§7.6.2.38); LCID (§7.6.2.39); Medium (§7.6.2.40); Middle (§7.6.2.41); Month (§7.6.2.42); MonthAccessed (§7.6.2.43); NumberVolumes (§7.6.2.45); Pages (§7.6.2.46); PatentNumber (§7.6.2.47); PeriodicalTitle (§7.6.2.49); ProductionCompany (§7.6.2.52); PublicationTitle (§7.6.2.53); Publisher (§7.6.2.54); RecordingNumber (§7.6.2.55); RefOrder (§7.6.2.56); Reporter (§7.6.2.57); ShortTitle (§7.6.2.58); Sources@SelectedStyle (§7.6.2.60); Sources@StyleName (§7.6.2.60); Sources@URI (§7.6.2.60); ST_SourceType (§7.6.3.1); StandardNumber (§7.6.2.62); StateProvince (§7.6.2.63); Station (§7.6.2.64); Tag (§7.6.2.65); Theater (§7.6.2.66); ThesisType (§7.6.2.67); Title (§7.6.2.68); Type (§7.6.2.70); URL (§7.6.2.71); Version (§7.6.2.72); Volume (§7.6.2.73); Year (§7.6.2.75); YearAccessed (§7.6.2.76)

7.7 Additional Characteristics

In order to allow producers of Office Open XML to describe specific contextual conditions under which the document was created, additional characteristics can be provided within the Additional Characteristics part using the syntax defined below.

The set of additional characteristics is designed to be an extensible list, and can provide a consumer with more information on how to interpret the file. This Standard defines one set of characteristics; however, additional grammars can be created and associated with a unique URI via the vocabulary attribute.

7.7.1 Table of Contents

This subclause is informative.

End of informative text.

7.7.2 Elements

The following elements define the contents of the Additional Characteristics schema:

7.7.2.1 additionalCharacteristics (Set of Additional Characteristics)

This element is the root element of the Additional Characteristics part and contains the list of additional characteristics for an Office Open XML document.

[*Example:* The following content in an Additional Characteristics part would specify that the producing spreadsheet application supports from 0 to 10,000 columns, and that column ranges should be interpreted accordingly:

```
<additionalCharacteristics>
  <characteristic name="numColumns" relation="le" val="10000"/>
  <characteristic name="numColumns" relation="ge" val="0"/>
</additionalCharacteristics>
```

end example]

Parent Elements
Root element of Shared Additional Characteristics part

Child Elements	Subclause
characteristic (Single Characteristic)	§7.7.2.2

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_AdditionalCharacteristics">
   <sequence>
      <element name="characteristic" type="CT_Characteristic" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
</complexType>
```

7.7.2.2 characteristic (Single Characteristic)

This element specifies a single characteristic. The type of characteristic is defined by the name attribute.

[*Example:* A producer can inform the consumer that the computations used to calculate the stored numbers in the formulas belong to a value space expressed by ranges of the binary mantissa and exponent. A consumer can optionally check those characteristics to determine whether, for example, the values should be recalculated. The XML for this would be:

```
<additionalCharacteristics>
  <characteristic name='precisionMantissa' relation='gt'
     val='-9007199254740992'/>
  <characteristic name='precisionMantissa' relation='lt'
     val='9007199254740992'/>
  <characteristic name='precisionExponent' relation='ge' val='-1075'/>
  <characteristic name='precisionExponent' relation='le' val='970'/>
</additionalCharacteristics>
```

end example]

Parent Elements
additionalCharacteristics (§7.7.2.1)

Attributes	Description
name (Name of Characteristic)	Specifies the name of the characteristic. There are no constraints on the value of the name attribute, but each name shall be associated with a specific vocabulary via the vocabulary attribute. The values defined by this Standard shall be associated with a null vocabulary value, and are as follows: **Name Value** / **Property Specified** numColumns / Number of Columns supported by the spreadsheet producer.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>numRows</td><td>Number of Rows supported by the spreadsheet producer.</td></tr>
<tr><td>functionVersion</td><td>Version of the function specification used</td></tr>
<tr><td>precisionMantissa</td><td>Allowed values of the mantissa of numbers within spreadsheet cells/formulas when expressed in base 2.</td></tr>
<tr><td>precisionExponent</td><td>Allowed values of the exponent of numbers within spreadsheet cells/formulas when expressed in base 2.</td></tr>
<tr><td>numWorkbookColors</td><td>Number of Workbook colors</td></tr>
<tr><td>numConditionalFormatConditions</td><td>Number of condition format conditions on a workbook cell</td></tr>
<tr><td>nummaxSortLevels</td><td>Number of level of sorting on a range or table</td></tr>
<tr><td>numAutoFilterItems</td><td>Number of items shown in the Auto-filter dropdown</td></tr>
<tr><td>numDisplayCellChars</td><td>Number of characters that can display in a cell</td></tr>
<tr><td>numPrintCellChars</td><td>Number of characters per cell that Excel can print</td></tr>
<tr><td>numUnqiueCellStyles</td><td>Number of unique cell styles in a workbook (combinations of all cell formatting)</td></tr>
<tr><td>numFormulaLengthChars</td><td>Length of formulas in characters</td></tr>
<tr><td>numFormulaNestingLevel</td><td>Number of levels of formula nesting</td></tr>
<tr><td>numFunctionArguments</td><td>Number of arguments to a function</td></tr>
<tr><td>numPivotTableRows</td><td>Number of rows in a pivot table</td></tr>
<tr><td>numPivotTableColumns</td><td>Number of columns in a pivot table</td></tr>
<tr><td>numUniquePivotFieldItems</td><td>Number of unique items in a pivot field</td></tr>
<tr><td>numPivotTableMDXNameChars</td><td>Number of characters in a MDX name for a pivot table item</td></tr>
<tr><td>numPivotTableRelationChars</td><td>String length for a relationship pivot table</td></tr>
<tr><td>numPivotTableFieldLabelChars</td><td>Length of field labels in PivotTable including caption length limitations</td></tr>
<tr><td>numPivotTableFields</td><td>Number of fields in a pivot table</td></tr>
<tr><td>numSheetXRefArrayFormulas</td><td>The number of array formulas in a worksheet that can refer to another (given) worksheet</td></tr>
</table>

The possible values for this attribute are defined by the XML Schema string datatype.
</td></tr>
<tr><td>relation (Relationship of Value to Name)</td><td>Specifies how the contents of the value attribute should be interpreted in the context of this characteristic.</td></tr>
</table>

Attributes	Description

	[*Example:* The following would specify that the application supports from 0 to 10,000 columns, and that column ranges should be interpreted accordingly:
	
	`<additionalCharacteristics>`
	`  <characteristic name="numColumns" relation="le" val="10000"/>`
	`  <characteristic name="numColumns" relation="ge" val="0"/>`
	`</additionalCharacteristics>`
	
	end example]
	
	The possible values for this attribute are defined by the ST_Relation simple type (§7.7.3.1).

val (Characteristic Value)	Specifies the value of the characteristic. The possible values for this attribute are defined by the XML Schema string datatype.
vocabulary (Characteristic Grammar)	Specifies a URI defining the characteristic grammar with which the name attribute value shall be interpreted. If this attribute is omitted, then the default grammar (as defined above) shall be used. The possible values for this attribute are defined by the XML Schema anyURI datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Characteristic">
   <attribute name="name" type="xsd:string" use="required"/>
   <attribute name="relation" type="ST_Relation" use="required"/>
   <attribute name="val" type="xsd:string" use="required"/>
   <attribute name="vocabulary" type="xsd:anyURI" use="optional"/>
</complexType>
```

7.7.3 Simple Types

This is the complete list of simple types in the http://schemas.openxmlformats.org/officeDocument/2006/characteristics namespace.

7.7.3.1 ST_Relation (Characteristic Relationship Types)

This simple type specifies the possible relationships between a characteristic's name and value attributes.

This simple type's contents are a restriction of the XML Schema string datatype.

The following are possible enumeration values for this type:

Enumeration Value	Description
eq (Equal To)	Equal to.
ge (Greater Than or Equal to)	Greater than or equal to.

Enumeration Value	Description
gt (Greater Than)	Greater than.
le (Less Than or Equal To)	Less than or equal to.
lt (Less Than)	Less than.

Referenced By
characteristic@relation (§7.7.2.2)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_Relation">
   <restriction base="xsd:string">
      <enumeration value="ge"/>
      <enumeration value="le"/>
      <enumeration value="gt"/>
      <enumeration value="lt"/>
      <enumeration value="eq"/>
   </restriction>
</simpleType>
```

7.8 Office Document Relationships

Within an Office Open XML document, it is necessary to be able to explicitly reference one part within the package from another [*Example*: A PresentationML Slide needs to be able to explicitly reference each picture within it to know where each one is anchored. *end example*]

In order to ensure that all such explicit relationship references are easily identifiable within a document, all such relationships are included within attributes in this namespace. This namespace therefore only serves to define attributes used throughout Office Open XML to create explicit relationships, and a single simple type for such attributes.

7.8.1 Table of Contents

This subclause is informative.

End of informative text.

7.8.2 Simple Types

This is the complete list of simple types in the http://schemas.openxmlformats.org/officeDocument/2006/relationships namespace.

7.8.2.1 ST_RelationshipId (Explicit Relationship ID)

This simple type specifies the relationship ID in a part's relationship item which is the target of an explicit relationship from the parent XML element.

The type of relationship which shall be the target of the relationship specified shall be determined based on the context of the parent XML element.

[*Example*: Consider the following markup in an Office Open XML document:

```
<… r:id="rId5" />
```

The id attribute is of type ST_RelationshipID, and therefore the relationship with ID `rId5` shall be the target of an explicit relationship from the source part, based on the context of the parent XML element. *end example*]

This simple type's contents are a restriction of the XML Schema string datatype.

Referenced By
altChunk@id (§2.17.3.1); attachedTemplate@id (§2.15.1.6); audioFile@link (§5.1.3.2); blip@embed (§5.1.10.13); blip@link (§5.1.10.13); bold@id (§4.3.1.1); boldItalic@id (§4.3.1.2); chart@id (§5.7.2.26); control@id (§2.3.3.2); control@id (§2.3.3.3); control@id (§4.4.2.1); control@id (§3.3.1.18); custData@id (§4.2.1); customPr@id (§3.3.1.20); dataRef@id (§3.3.1.28); dataSource@id (§2.14.9); drawing@id (§3.3.1.34); embedBold@id (§2.8.2.3); embedBoldItalic@id (§2.8.2.4); embedItalic@id (§2.8.2.5); embedRegular@id (§2.8.2.6); externalBook@id (§3.14.7); externalData@id (§5.7.2.63); externalReference@id (§3.2.8); fill@id (§6.1.2.5); footerReference@id (§2.10.2); handoutMasterId@id (§4.3.1.11); header@id (§3.11.1.1); headerReference@id (§2.10.5); headerSource@id (§2.14.16); hlinkClick@id (§5.1.5.3.5); hlinkHover@id (§5.1.2.1.23); hlinkMouseOver@id (§5.1.5.3.6); htmlPubPr@id (§4.3.1.13); hyperlink@id (§2.16.24); hyperlink@id (§3.3.1.44); imagedata@href (§6.1.2.11); imagedata@id (§6.1.2.11); imagedata@pict (§6.1.2.11); italic@id (§4.3.1.14); legacyDrawing@id (§3.3.1.51); legacyDrawingHF@id (§3.3.1.52); legacyDrawingHF@id (§5.7.2.93); movie@id (§2.3.3.17); notesMasterId@id (§4.3.1.18); oleLink@id (§3.14.11); oleObj@id (§4.4.2.4); oleObject@id (§3.3.1.57); OLEObject@id (§6.2.2.19); pageSetup@id (§3.3.1.62); pageSetup@id (§3.3.1.61); picture@id (§3.3.1.65); pivotCache@id (§3.2.17); pivotCacheDefinition@id (§3.10.1.67); pivotSelection@id (§3.3.1.67); printerSettings@id (§2.6.14); quickTimeFile@link (§5.1.3.4); rangeSet@id (§3.10.1.79); recipientData@id (§2.14.27); regular@id (§4.3.1.27); relIds@cs (§5.9.2.22); relIds@dm (§5.9.2.22); relIds@lo (§5.9.2.22); relIds@qs (§5.9.2.22); saveThroughXslt@id (§2.15.1.76); shape@blip (§5.9.2.27); sheet@id (§3.2.19); sld@id (§4.2.6); sld@id (§4.3.2.14); sldId@id (§4.3.1.29); sldLayoutId@id (§4.4.1.37); sldMasterId@id (§4.3.1.32); smartTags@id (§4.3.1.35); snd@embed (§4.6.68); snd@embed (§5.1.2.1.32); sndTgt@embed (§4.6.70); sourceFileName@id (§2.15.2.38); src@id (§2.14.30); stroke@id (§6.1.2.21); subDoc@id (§2.17.2.1); tablePart@id (§3.3.1.91); tags@id (§4.2.9); userShapes@id (§5.7.2.222); videoFile@link (§5.1.3.6); wavAudioFile@embed (§5.1.3.7); worksheetSource@id (§3.10.1.95)

The following XML Schema fragment defines the contents of this simple type:

```
<simpleType name="ST_RelationshipId">
   <restriction base="xsd:string"/>
</simpleType>
```

8. Custom XML Schema References

This namespace defines the set of properties which define the location and properties associated with one or more custom XML schemas which have been stored within the contents of a Office Open XML document. Collectively, the set of schemas associated with a document's custom XML markup are referred to as that document's *schema library*. The schema library then stores the set of unique XML namespaces used within the document's custom XML markup, and allows applications to 'tag' these namespaces with appropriate metadata.

8.1 Table of Contents

This subclause is informative.

End of informative text.

8.2 Elements

The following information describes the elements in this namespace:

8.2.1 schema (Custom XML Schema Reference)

This element specifies the properties associated with a single XML namespace, for which all known XML schemas shall be loaded in order to validate the custom XML markup stored within this document. These properties may be used appropriately to locate custom XML schema(s) for use with the document.

[*Example*: Consider a WordprocessingML document which contains custom XML markup in the http://www.contoso.com namespace. The following content would be displayed in the document's schema library data:

```
<w:schemaLibrary>
  <w:schema w:uri="http://www.contoso.com" w:schemaLocation="c:\contoso.xsd" />
</w:schemaLibrary>
```

The schema element contains the properties for this one XML namespace: in this case, a namespace URI of `http://www.contoso.com` and a file location of `c:\contoso.xsd`. *end example*]

Parent Elements
schemaLibrary (§8.2.2)

Attributes	Description

manifestLocation (Resource File Location)	Specifies the location of a resource file which should be downloaded and parsed when this document is loaded. The format and contents of this resource file are application-defined.
	
	[*Example*: Consider a WordprocessingML document which contains custom XML markup in the http://www.contoso.com namespace, which is associated with a resource file located at `http://www.contoso.com/resource.xml`. The following content would be displayed in the document's schema library data:
	
	`<w:schemaLibrary>`
	`  <w:schema w:uri="http://www.contoso.com" w:manifestLocation=`
	`"http://www.contoso.com/resource.xml" />`
	`</w:schemaLibrary>`
	
	The manifestLocation attribute contains `http://www.contoso.com/manifest.xml` which is the location of a resource file that may be downloaded for use when this namespace is used. *end example*]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

schemaLocation (Custom XML Schema Location)	Specifies the location of the XML schema file which should be downloaded and parsed when this document is loaded.
	
	[*Example*: Consider a WordprocessingML document which contains custom XML markup in the http://www.contoso.com namespace, which is defined by an XML schema located at `c:\contoso.xsd`. The following content would be displayed in the document's schema library data:
	
	`<w:schemaLibrary>`
	`  <w:schema w:uri="http://www.contoso.com" w:schemaLocation=`
	`"c:\contoso.xsd" />`
	`</w:schemaLibrary>`
	
	The schemaLocation attribute contains `c:\contoso.xsd` which is the location of the XML schema file used when this namespace is used. *end example*]
	
	The possible values for this attribute are defined by the XML Schema string datatype.

uri (Custom XML Schema Namespace)	Specifies the target namespace for the XML Schema associated with this schema reference.
	
	[*Example*: Consider the following content for custom XML namespace data:
	
	…
	`<w:schema w:uri="http://www.contoso.com/schema1" />`
	`<w:schema w:uri="http://www.contoso.com/schema2" />`
	…

Attributes	Description
	The uri attribute specifies the target namespace of each XML schema reference: • http://www.contoso.com/schema1 • http://www.contoso.com/schema2 Applications may then locate and utilize a schema for these namespaces using any means available. *end example*] The possible values for this attribute are defined by the XML Schema string datatype.

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_Schema">
   <attribute name="uri" type="xsd:string" default=""/>
   <attribute name="manifestLocation" type="xsd:string"/>
   <attribute name="schemaLocation" type="xsd:string"/>
</complexType>
```

8.2.2 schemaLibrary (Embedded Custom XML Schema Supplementary Data)

This element specifies the set of XML namespaces which have been associated with the contents of the custom XML markup within the current Office Open XML document. Each unique namespace which is referenced within the document may be referenced within this element by a single schema element, regardless of the number of constituent XML schemas which comprise that namespace.

[*Example*: Consider a WordprocessingML document which contains custom XML markup in two distinct namespaces: the http://www.contoso.com namespace and the http://www.woodgroveBank.com namespace. If the first namespace is defined by a single XML schema, and the second is defined by five XML schemas (which are cross-referenced using the appropriate XML Schema syntax), the following content would be displayed in the document's schema library XML:

```
<w:schemaLibrary>
  <w:schema … />
  <w:schema … />
</w:schemaLibrary>
```

The schemaLibrary element contains only two schema elements even though there are six XML schemas in use, as there are only two distinct namespaces for which data is stored. *end example*]

Parent Elements
settings (§2.15.1.78)

Child Elements	Subclause
schema (Custom XML Schema Reference)	§8.2.1

The following XML Schema fragment defines the contents of this element:

```
<complexType name="CT_SchemaLibrary">
    <sequence>
        <element name="schema" type="CT_Schema" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
</complexType>
```

Annex A. Office Schemas – XML Schema

This Office Open XML specification includes a family of schemas defined using the XML Schema 1.0 syntax. The normative definitions of these schemas reside in an accompanying file named OfficeOpenXML-XMLSchema.zip, which is distributed in electronic form only.

If discrepancies exist between the electronic version of a schema and its corresponding representation as published in this part, Part 4, the electronic version is the definitive version.

Annex B. Schemas – RELAX NG

This clause is informative.

This Office Open XML specification includes a family of schemas defined using the RELAX NG syntax. The definitions of these schemas reside in an accompanying file named OfficeOpenXML-RELAXNG.zip, which is distributed in electronic form only.

If discrepancies exist between the RELAX NG version of a schema and its corresponding XML Schema, the XML Schema is the definitive version.

End of informative text.

Annex C. Additional Syntax Constraints

This clause is informative.

Although the set of XML Schemas included in Annex A specifies a majority of the requirements needed to ensure document conformance, there also exists a set of constraints that cannot easily be expressed in the XML Schema syntax (otherwise referred to as *additional syntax constraints*). These additional constraints can be deduced from the normative content of this Part, and are included in the requirements for document conformance.

The following is a sample of such constraints specified in this Part:

Subclause	Constraint
WordprocessingML Paragraphs	The caps element shall not be present with the smallCaps element on the same run, since they are mutually exclusive in terms of appearance.
WordprocessingML Headers and Footers	If the relationship type of the relationship specified by this element shall be present, have a TargetMode attribute value of `Internal`, and a Type attribute value of http://schemas.openxmlformats.org/officeDocument/2006/header.
WordprocessingML Annotations	Each "cross-structure" annotation shall have a start element whose id attribute value matches the id attribute value of the corresponding end element.
SpreadsheetML Formulas	The syntax of the value of a formula shall conform to the structure of a formula as defined in the Formulas subclause.
SpreadsheetML Styles	The xfId attribute on the cellStyles element shall specify an xf record which is present in the cellStyleXfs collection.
PresentationML Slides	Each sldId attribute in a sldLst shall have a unique value for its id attribute (within the scope of the collection).
PresentationML Slides	A customShow element specifies a range of slides (a start and end slide) in PresentationML to define the set of slides that define a slide show. There shall be slides in the presentation that have slide IDs referenced in these attributes and the start slide shall be before the end slide.
DrawingML Main	The id attribute on the stCxn and endCxn elements specifies the start and end shapes to be connected. Shapes with matching id attribute values shall exist elsewhere in the spTree.

End of informative text.

Annex D. Predefined SpreadsheetML Style Definitions

This Office Open XML specification includes the definitions of all predefined SpreadsheetML styles that are referenced by the following elements:

- cellStyle (§3.8.7)
- tableStyle (§3.8.40)

The normative SpreadsheetML markup defining these styles resides in an accompanying file named OfficeOpenXML-SpreadsheetMLStyles.zip, which is distributed in electronic form only.

Annex E. Example Predefined DrawingML Shape and Text Geometries

This clause is informative.

This Office Open XML specification includes an example definition for all predefined DrawingML shape geometries that are referenced by the following elements:

- prstGeom@prst (§5.1.11.18)
- prstTxWarp@prst (§5.1.11.19)

The informative sample DrawingML markup defining these shape and text geometries resides in an accompanying file named OfficeOpenXML-DrawingMLGeometries.zip, which is distributed in electronic form only.

End of informative text.

Annex F. Root Element Locations

This clause is informative.

This clause provides the location of each part's root element (as identified in §1) within the set of normative XML Schema files provided in Annex A, based on both its part name and its XML Schema:

F.1 Grouped by Part Name

Part	Schema	Element Name
DrawingML Chart	dml-chart.xsd	chartSpace
DrawingML Chart Drawing	dml-chart.xsd	userShapes
DrawingML Diagram Colors	dml-diagramColorTransform.xsd	colorsDef
DrawingML Diagram Data	dml-diagramDefinition.xsd	dataModel
DrawingML Diagram Layout Definition	dml-diagramDefinition.xsd	layoutDef
DrawingML Diagram Style	dml-diagramStyleDefinition.xsd	styleDef
DrawingML Table Styles	dml-tableStyle.xsd	tblStyleLst
DrawingML Theme	dml-stylesheet.xsd	theme
DrawingML Theme Override	dml-stylesheet.xsd	themeOverride

Part	Schema	Element Name
PresentationML Comment Authors	pml-comments.xsd	cmAuthorLst
PresentationML Comments	pml-comments.xsd	cmLst
PresentationML Handout Master	pml-slide.xsd	handoutMaster
PresentationML Notes Master	pml-slide.xsd	notesMaster
PresentationML Notes Slide	pml-slide.xsd	notes
PresentationML Presentation	pml-presentation.xsd	presentation
PresentationML Presentation Properties	pml-presentationProperties.xsd	presentationPr
PresentationML Slide	pml-slide.xsd	sld
PresentationML Slide Layout	pml-slide.xsd	sldLayout
PresentationML Slide Master	pml-slide.xsd	sldMaster
PresentationML Slide Synchronization Data	pml-slideSynchronizationData.xsd	sldSyncPr
PresentationML User-Defined Tags	pml-userDefinedTags.xsd	tagLst

Part	Schema	Element Name
PresentationML View Properties	pml-viewProperties.xsd	viewPr

Part	Schema	Element
Shared Additional Characteristics	shared-additionalCharacteristics.xsd	additionalCharacteristics
Shared Extended File Properties	shared-documentPropertiesExtended.xsd	Properties
Shared Bibliography	shared-bibliography.xsd	Sources
Shared Custom File Properties	shared-documentPropertiesCustom.xsd	Properties
Shared Custom XML Data Storage Properties	shared-customXmlDataProperties.xsd	datastoreItem

Part	Schema	Element Name
SpreadsheetML Calculation Chain	sml-calculationChain.xsd	calcChain
SpreadsheetML Chartsheet	sml-sheet.xsd	chartsheet
SpreadsheetML Comments	sml-comments.xsd	comments
SpreadsheetML Connections	sml-externalConnections.xsd	connections
SpreadsheetML Custom XML Mappings	sml-customXmlMappings.xsd	MapInfo
SpreadsheetML Dialogsheet	sml-sheet.xsd	dialogsheet
SpreadsheetML Drawing	dml-spreadsheetDrawing.xsd	wsDr
SpreadsheetML External Workbook References	sml-supplementaryWorkbooks.xsd	externalLink
SpreadsheetML Metadata	sml-sheetMetadata.xsd	metadata
SpreadsheetML Pivot Table	sml-pivotTable.xsd	pivotTableDefinition
SpreadsheetML Pivot Table Cache Definition	sml-pivotTable.xsd	pivotCacheDefinition
SpreadsheetML Pivot Table Cache Records	sml-pivotTable.xsd	pivotCacheRecords
SpreadsheetML Query Table	sml-queryTable.xsd	queryTable
SpreadsheetML Shared String Table	sml-sharedStringTable.xsd	sst
SpreadsheetML Shared Workbook Revision Headers	sml-sharedWorkbookRevisions.xsd	header
SpreadsheetML Shared Workbook Revision Log	sml-sharedWorkbookRevisions.xsd	revisions
SpreadsheetML Shared Workbook User Data	sml-sharedWorkbookUserNames.xsd	users
SpreadsheetML Single Cell Table Definitions	sml-singleCellTable.xsd	singleXmlCells
SpreadsheetML Styles	sml-styles.xsd	styleSheet

Part	Schema	Element Name
SpreadsheetML Table Definitions	sml-table.xsd	table
SpreadsheetML Volatile Dependencies	sml-volatileDependencies.xsd	volTypes
SpreadsheetML Workbook	sml-workbook.xsd	workbook
SpreadsheetML Worksheet	sml-sheet.xsd	worksheet

Part	Schema	Element Name
WordprocessingML Comments	wml.xsd	comments
WordprocessingML Document Settings	wml.xsd	settings
WordprocessingML Endnotes	wml.xsd	endnotes
WordprocessingML Font Table	wml.xsd	fonts
WordprocessingML Footer	wml.xsd	ftr
WordprocessingML Footnotes	wml.xsd	footnotes
WordprocessingML Glossary Document	wml.xsd	glossaryDocument
WordprocessingML Header	wml.xsd	hdr
WordprocessingML Mail Merge Recipient Data	wml.xsd	recipientData
WordprocessingML Main Document	wml.xsd	document
WordprocessingML Numbering Definitions	wml.xsd	numbering
WordprocessingML Style Definitions	wml.xsd	styles
WordprocessingML Web Settings	wml.xsd	webSettings

F.2 Grouped by Schema Name

Schema	Part Name	Element
dml-chart.xsd	DrawingML Chart	chartSpace
	DrawingML Chart Drawing	userShapes
dml-diagramColorTransform.xsd	DrawingML Diagram Colors	colorsDef
dml-diagramDefinition.xsd	DrawingML Diagram Data	dataModel
	DrawingML Diagram Layout Definition	layoutDef
dml-diagramStyleDefinition.xsd	DrawingML Diagram Style	styleDef
dml-spreadsheetDrawing.xsd	SpreadsheetML Drawing	wsDr
dml-stylesheet.xsd	DrawingML Theme	theme
	DrawingML Theme Override	themeOverride

Schema	Part Name	Element
dml-tableStyle.xsd	DrawingML Table Styles	tblStyleLst

Schema	Part Name	Element
pml-comments.xsd	PresentationML Comment Authors	cmAuthorLst
	PresentationML Comments	cmLst
pml-presentation.xsd	PresentationML Presentation	presentation
pml-presentationProperties.xsd	PresentationML Presentation Properties	presentationPr
pml-slide.xsd	PresentationML Handout Master	handoutMaster
	PresentationML Notes Master	notesMaster
	PresentationML Notes Slide	notes
	PresentationML Slide	sld
	PresentationML Slide Layout	sldLayout
	PresentationML Slide Master	sldMaster
pml-slideSynchronizationData.xsd	PresentationML Slide Synchronization Data	sldSyncPr
pml-userDefinedTags.xsd	PresentationML User-Defined Tags	tagLst
pml-viewProperties.xsd	PresentationML View Properties	viewPr

Schema	Part Name	Element
shared-additionalCharacteristics.xsd	Shared Additional Characteristics	additionalCharacteristics
shared-bibliography.xsd	Shared Bibliography	Sources
shared-customXmlDataProperties.xsd	Shared Custom XML Data Storage Properties	datastoreItem
shared-documentPropertiesCustom.xsd	Shared Custom File Properties	Properties
shared-documentPropertiesExtended.xsd	Shared Application-Defined File Properties	Properties

Schema	Part Name	Element
sml-calculationChain.xsd	SpreadsheetML Calculation Chain	calcChain
sml-comments.xsd	SpreadsheetML Comments	comments
sml-customXmlMappings.xsd	SpreadsheetML Custom XML Mappings	MapInfo
sml-externalConnections.xsd	SpreadsheetML Connections	connections
sml-pivotTable.xsd	SpreadsheetML Pivot Table	pivotTableDefinitio

Schema	Part Name	Element
		n
	SpreadsheetML Pivot Table Cache Definition	pivotCacheDefinition
	SpreadsheetML Pivot Table Cache Records	pivotCacheRecords
sml-queryTable.xsd	SpreadsheetML Query Table	queryTable
sml-sharedStringTable.xsd	SpreadsheetML Shared String Table	sst
sml-sharedWorkbookRevisions.xsd	SpreadsheetML Shared Workbook Revision Headers	header
	SpreadsheetML Shared Workbook Revision Log	revisions
sml-sharedWorkbookUserNames.xsd	SpreadsheetML Shared Workbook User Data	users
sml-sheet.xsd	SpreadsheetML Chartsheet	chartsheet
	SpreadsheetML Dialogsheet	dialogsheet
	SpreadsheetML Worksheet	worksheet
sml-sheetMetadata.xsd	SpreadsheetML Metadata	metadata
sml-singleCellTable.xsd	SpreadsheetML Single Cell Table Definitions	singleXmlCells
sml-styles.xsd	SpreadsheetML Styles	styleSheet
sml-supplementaryWorkbooks.xsd	SpreadsheetML External Workbook References	externalLink
sml-table.xsd	SpreadsheetML Table Definitions	table
sml-volatileDependencies.xsd	SpreadsheetML Volatile Dependencies	volTypes
sml-workbook.xsd	SpreadsheetML Workbook	workbook

Schema	Part Name	Element
wml.xsd	WordprocessingML Comments	comments
	WordprocessingML Document Settings	settings
	WordprocessingML Endnotes	endnotes
	WordprocessingML Font Table	fonts
	WordprocessingML Footer	ftr
	WordprocessingML Footnotes	footnotes
	WordprocessingML Glossary Document	glossaryDocument
	WordprocessingML Header	hdr
	WordprocessingML Mail Merge Recipient Data	recipientData

Schema	Part Name	Element
	WordprocessingML Main Document	document
	WordprocessingML Numbering Definitions	numbering
	WordprocessingML Style Definitions	styles
	WordprocessingML Web Settings	webSettings

End of informative text.

Office Open XML

Part 5: Markup Compatibility and Extensibility

December 2006

Table of Contents

Foreword

This multi-part Standard deals with Office Open XML Format-related technology, and consists of the following parts:

- Part 1: "Fundamentals"
- Part 2: "Open Packaging Conventions"
- Part 3: "Primer"
- Part 4: "Markup Language Reference"
- **Part 5: "Markup Compatibility and Extensibility" (this document)**

1. Scope

This Part (the *Markup Compatibility and Extensibility specification*) describes a set of conventions that are used by Office Open XML documents that facilitate future enhancement and extension of Office Open XML documents, while providing a baseline for interoperability.

In all subsequent uses within this document, the term "this specification" shall refer to the content of this Part.

2. Normative References

The following normative documents contain provisions, which, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this specification are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 2382.1:1993, *Information technology — Vocabulary — Part 1: Fundamental terms*.

ISO/IEC 10646:2003 (all parts), *Information technology — Universal Multiple-Octet Coded Character Set (UCS)*.

3. Definitions

For the purposes of this specification, the following definitions apply. Other terms are defined where they appear in *italics* type. Terms explicitly defined in this specification are not to be presumed to refer implicitly to similar terms defined elsewhere. [*Note*: This part uses OPC-related terms, which are defined in Part 2: "Open Packaging Conventions". *end note*]

Throughout this specification, the terms *namespace declaration*, *namespace name*, *qualified name*, *expanded name*, *prefixed name*, *unprefixed name*, and *local name* shall have the meanings as defined in the W3C Recommendation, “Namespaces in XML 1.0 (Second Edition).”

alternate content — A set of markup alternatives, of which no more than one shall be processed by a markup consumer. A markup consumer chooses from among the alternatives based upon its set of understood namespaces.

compatibility-rule attribute — An XML attribute described in this specification that expresses rules governing markup consumers’ behavior when encountering XML elements and attributes from non-understood namespaces.

ignore — To disregard the presence of an element or attribute, processing the markup as if that element or attribute did not exist.

markup consumer — A tool that can read and parse a markup document and further conforms to the requirements of a markup specification.

markup document — An XML document that conforms to the requirements of a markup specification.

markup editor — A tool that acts as a markup consumer in reading a markup document, makes changes to that markup, and acts as the producer of the modified markup.

markup preprocessor — A software module, designed for use in the implementation of markup consumers, that follows the rules of this Markup Compatibility and Extensibility specification to remove or replace all elements and attributes from the Markup Compatibility namespace, all elements and attributes from ignorable non-understood namespaces, and all elements and attributes from subsumed namespaces..

markup producer — A tool that can generate a markup document, and conforms to a markup specification.

markup specification — An XML-based markup format specification that incorporates all of the requirement of this Part.

namespace, ignorable — A namespace, identified in markup, whose elements and attributes shall be ignored by a markup consumer that does not understand that namespace.

namespace, understood — An XML namespace containing any recognized XML elements or attributes.

preserve —To retain an ignored element or attribute during the course of editing.

recognize — To have knowledge of the correct interpretation of an XML element, XML attribute, or attribute-value, as defined in a markup specification.

4. Notational Conventions

The following typographical conventions are used in this Standard:

1. The first occurrence of a new term is written in italics. [*Example*: … is considered *normative*. *end example*]
2. A term defined as a basic definition is written in bold. [*Example*: **behavior** — External … *end example*]
3. The name of an XML element is written using an Element style. [*Example*: The root element is document. *end example*]
4. The name of an XML element attribute is written using an Attribute style. [*Example*: … an id attribute. *end example*]
5. An XML element attribute value is written using a constant-width style. [*Example*: … value of `CommentReference`. *end example*]
6. An XML element type name is written using a Type style. [*Example*: … as values of the xsd:anyURI data type. *end example*]

5. Acronyms and Abbreviations

This clause is informative

The following acronyms and abbreviations are used throughout this specification

IEC — the International Electrotechnical Commission

ISO — the International Organization for Standardization

W3C — World Wide Web Consortium

End of informative text

6. General Description

This specification is intended for use by implementers, academics, and application programmers. As such, it contains a considerable amount of explanatory material that, strictly speaking, is not necessary in a formal specification.

This specification is divided into the following subdivisions:

1. Front matter (clauses 1–7);
2. Overview and introductory material (clause 8);
3. Main body (clauses 9–13);
4. Annexes

Examples are provided to illustrate possible forms of the constructions described. References are used to refer to related clauses. Notes are provided to give advice or guidance to implementers or programmers.

The following clauses form the normative pieces of this specification:

- Clauses 1–4, 6, and 8–11

The following clauses form the informative pieces of this specification:

- Introduction
- Clause 5, 7, and 12
- All annexes
- All notes and examples

Whole clauses that are informative are identified as such. Informative text that is contained within normative text is identified as either an example or note, as specified in §4.

7. Overview

This clause is informative

This Part describes a set of XML elements and attributes that collectively enable producers to explicitly guide consumers in their handling of any XML elements and attributes not understood by the consumer.

These elements and attributes enable the creation of future versions of and extensions to this Standard, while enabling desirable compatibility characteristics:

- A markup producer can produce markup documents that exploit new features defined by versions and extensions, yet remain interoperable with markup consumers thats are unaware of those versions and extensions.
- For any such markup document, a markup consumer whose implementation is aware of the exploited versions and extensions will deliver functionality that is enhanced by the markup document's use of those versions and extensions.
- For any such markup document, the markup producer can enable and precisely control graceful degradation that will occur when the markup document is processed by a markup consumer that is unaware of the exploited versions and extensions.

End of informative text

8. Markup Compatibility Fundamentals

8.1 Terminology

Any XML-based document specification can use the markup language described in this Part as the basis of its compatibility with previous and future specification revisions and to enable the creation of independent extensions of its specification. In this specification, the term *markup specification* is used to refer to a specification that relies on this Office Open XML Markup Compatibility and Extensibility Part and defines a set of XML namespaces, the elements and attributes within those namespaces, and any processing requirements for those namespaces, elements, and attributes. *Markup document* refers to an XML document that conforms to a markup specification. A *markup producer* is a software application or component that generates a markup document. A *markup consumer* is a software application or component that can process a markup document according to the processing requirements of the markup specification.

This specification is dependent on XML namespace names, expressed as URIs. A markup specification defines a set of elements and attributes within one or more namespaces. A characteristic of a markup consumer is that it can *recognize* or process the elements and attributes within *understood namespaces*, including those containing elements and attributes defined in the markup specification. Markup consumers shall process all recognized elements and attributes of any understood namespace according to the requirements of the markup specifications defining those elements or attributes. A markup specification might require that the presence of unrecognized elements or attributes in an understood namespace be treated as an error condition; however, markup consumers shall always treat the presence of an unrecognized element or attribute from the Markup Compatibility namespace as an error condition. If a markup consumer encounters an element or attribute from a non-understood namespace, the markup consumer shall treat the presence of that element or attribute as an error condition, unless the markup producer has embedded in the markup document explicit Markup Compatibility elements or attributes that override that behavior.

Within a markup document, a markup producer might use Markup Compatibility attributes to identify *ignorable namespaces*. Markup consumers shall *ignore* elements and attributes from namespaces that are both non-understood and ignorable, and shall not treat their presence as errors. A markup producer can indicate to the markup consumer whether the content of an ignored element shall be disregarded together with the ignored element, or if the content should be processed as if it were the content of the ignored element's parent.

Within a markup docume

nt, a markup producer might also use Markup Compatibility attributes to suggest to a markup editor that the editor attempt to *preserve* some ignored elements or attributes. The markup editor can attempt to persist these ignored elements and attributes when a saving markup document, despite the editor's inability to recognize the purpose of these ignored elements and attributes.

A markup producer, aware of the existence of markup consumers with overlapping but different sets of understood namespaces, might choose to include in a markup document *alternate content* regions, each holding a set of markup alternatives for use by different markup consumers. A markup consumer shall use rules embedded in the markup document by the markup producer to select no more than one of these alternatives for normal processing, and shall disregard all other alternatives.

Future versions of markup specifications shall specify new namespaces for any markup that is enhanced or modified by the new version, which a markup consumer of that version of the markup specification would include as an understood namespace. Some of the namespaces introduced in the new markup specification might each subsume one of the previous version's understood namespaces. A new understood namespace *subsumes* a previously-understood namespace if it includes all of the elements, attributes, and attribute values of the previously-understood namespace. Regardless of whether a new namespace subsumes a previously defined namespace, markup consumers based on a new version of a markup specification shall support all understood namespaces of the previous version unless the new version makes an explicit statement to the contrary.

This specification can be implemented using a pipelined, preprocessing architecture in the form of a software module called a *markup preprocessor*. A markup preprocessor can use the Markup Compatibility elements and attributes to produce output that is free of all ignorable non-understood content, all Markup Compatibility elements and attributes, and all elements and attributes in subsumed namespaces.

Markup consumers should report errors when processing non-conforming documents.

8.2 Markup Compatibility Namespace

The following is the Markup Compatibility namespace:

```
http://schemas.openxmlformats.org/markup-compatibility/2006
```

The namespace includes XML elements and attributes that markup producers can use to express to markup consumers how they shall respond to markup in non-understood namespaces. The elements and attributes described in this specification are contained in the Markup Compatibility namespace.

9. Markup Compatibility Attributes and Elements

This specification defines attributes to express compatibility rules and elements to specify alternate content.

[*Note*: Whitespace characters that can appear in attribute values, as defined in the XML specification, are described in the following table:

Table 9–1. Whitespace characters in attribute values

Character	Syntax
space	#x20
tab	#x9
line feed	#xA
carriage return	#xD

end note]

Whitespace characters that appear in values of attributes defined in this specification shall be normalized by markup consumers before processing as follows:

1. Replace each tab, line feed, and carriage return with a space.
2. Collapse contiguous sequences of spaces into a single space.
3. Remove leading and trailing spaces.

[*Note*: The following table, and Table 9–3, summarize the Markup Compatibility attributes and elements, respectively, and are further described in the sub-clauses that follow.

Table 9–2. Compatibility-rule attributes

Name	Description
Ignorable	A whitespace-delimited list of namespace prefixes that identify a set of namespaces whose elements and attributes should be silently ignored by markup consumers that do not understand the namespace of the element or attribute in question.
ProcessContent	A whitespace-delimited list of element-qualified names identifying the expanded names of elements whose content shall be processed, even if the elements themselves are ignored. In any qualified name in the list, the wildcard character “*” can replace the local name to indicate that the content of all elements in the namespace shall be processed.

Name	Description
PreserveElements	A whitespace-delimited list of element qualified names identifying the expanded names of elements that a markup producer suggests for preservation by markup editors, even if the elements themselves are ignored. In any qualified name in the list, the wildcard character "*" can replace the local name to indicate that all elements in the namespace should be preserved.
PreserveAttributes	A whitespace-delimited list of attribute qualified names identifying the expanded names of attributes that a markup producer suggests for preservation by markup editors. In any qualified name in the list, the wildcard character "*" can replace the local name to indicate that all attributes in the namespace should be preserved.
MustUnderstand	A whitespace-delimited list of namespace prefixes identifying a set of namespace names. Markup consumers that do not understand these namespaces shall not continue to process the markup document and shall generate an error.

Table 9–3. Alternate-content elements

Name	Description
AlternateContent	Associates a set of possible markup alternatives that a markup consumer might choose based on that markup consumer's understood namespaces. The markup consumer chooses the first alternative, in markup order, requiring only namespaces it understands.
Choice	This child of AlternateContent contains a single markup alternative and identifies the namespaces that the markup consumer needs to understand in order to choose and process that alternative. At least one Choice element is required.
Fallback	This child of AlternateContent specifies the fallback markup alternative a markup consumer chooses if the markup consumer cannot choose any Choice alternative. An AlternateContent element shall hold no more than one Fallback element, which if present, shall follow all Choice elements.

end note]

9.1 Compatibility-Rule Attributes

This specification describes the manner by which compatibility rules can be associated with any XML element, including Markup Compatibility elements. Compatibility rules are associated with an element by means of compatibility-rule attributes. These attributes control how markup consumers, including markup editors, shall react to elements or attributes from non-understood namespaces.

The principal compatibility-rule attribute is the Ignorable attribute. By default, markup consumers should treat the presence of any element or attribute from a non-understood namespace as an error condition. However, elements and attributes from a non-understood namespace identified in an Ignorable attribute shall be ignored without error.

Compatibility-rule attributes shall affect the element to which they are attached, including the element's other attributes and contents. The order in which compatibility-rule attributes occur on an element shall not affect the application of those rules to that element, its attributes, or its contents.

9.1.1 Ignorable Attribute

The Ignorable attribute value contains a whitespace-delimited list of namespace prefixes, where each namespace prefix identifies an ignorable namespace. During processing, if a markup consumer encounters an element or attribute in a non-understood and ignorable namespace, the markup consumer shall treat that element or attribute as if it did not exist and shall not generate an error.

Markup consumers should treat elements and attributes from non-ignorable and non-understood namespaces as errors.

[*Note*: By default, an ignored element is ignored in its entirety, including its attributes and its content. The processing of an ignored element's contents is enabled through the use of the ProcessContents attribute. The PreserveAttributes and PreserveElements attributes can be used to assist markup editors in preserving ignored elements and ignored attributes. *end note*]

If an Ignorable attribute references an understood namespace, its presence shall not affect the processing of elements and attributes from the understood namespace, regardless of whether or not those elements and attributes are recognized by the markup consumer.

The presence of an Ignorable attribute shall reset a markup consumer's content-processing and preservation behavior for all elements and attributes in the namespaces referenced by the Ignorable attribute value. Once reset, by default the markup consumer shall ignore all content contained by the ignored element and markup editors shall not preserve ignored attributes and elements. This default behavior shall be overridden by the presence of any ProcessContent, PreserveAttributes, and PreserveElements attributes on the element with the Ignorable attribute.

The value of the Ignorable attribute can be an empty or blank string. When a markup consumer encounters such a value, it shall proceed as if the Ignorable attribute was not present.

[*Example*: Example 9–1. Processing Ignorable and PreserveAttributes attributes

The example namespace with the name `http://schemas.openxmlformats.org/Circles/v1` defines a Version 1 element, Circle, in its initial version. The subsequent Version 2 of the markup specification introduces the Opacity attribute in a new Version 2 namespace. The subsequent Version 3 of the markup specification introduces the Luminance attribute in a Version 3 namespace. The markup is loadable by markup consumers conforming to any one of these markup specification versions. The PreserveAttributes attribute specifies that the v3:Luminance attribute should be preserved during editing, even when the markup editor does not understand the v3:Luminance attribute.

For a Version 1 markup consumer, Opacity and Luminance are ignored attributes.

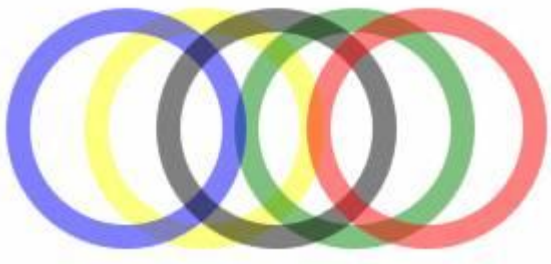
For a Version 2 markup consumer, only Luminance is an ignored attribute.

For a Version 3 markup consumer and beyond, none of the attributes are ignored.

```
<Circles xmlns="http://schemas.openxmlformats.org/Circles/v1"
  xmlns:mc="http://schemas.openxmlformats.org/markup-
  compatibility/2006"
  xmlns:v2="http://schemas.openxmlformats.org/Circles/v2"
  xmlns:v3="http://schemas.openxmlformats.org/Circles/v3"
  mc:Ignorable="v2 v3"
  mc:PreserveAttributes="v3:Luminance">
  <Circle Center="0,0" Radius="20" Color="Blue"
    v2:Opacity="0.5" v3:Luminance="13" />
  <Circle Center="25,0" Radius="20" Color="Black"
    v2:Opacity="0.5" v3:Luminance="13" />
  <Circle Center="50,0" Radius="20" Color="Red"
    v2:Opacity="0.5" v3:Luminance="13" />
  <Circle Center="13,0" Radius="20" Color="Yellow"
    v2:Opacity="0.5" v3:Luminance="13" />
  <Circle Center="38,0" Radius="20" Color="Green"
    v2:Opacity="0.5" v3:Luminance="13" />
</Circles>
```

The following figure shows an example possible rendering of the markup above.

Figure 9–1. Rings



end example]

[*Example*: Example 9–2. Processing Ignorable content using namespaces

A markup consumer that does not understand the namespace with the name `http://schemas.openxmlformats.org/MyExtension/v1` shall ignore both the a:IgnoreMe and b:IgnoreMeToo elements. Although the two elements use different namespace prefixes, they draw from the same ignorable namespace.

```
<Circles
  xmlns="http://schemas.openxmlformats.org/Circles/v1"
  xmlns:mc="http://schemas.openxmlformats.org/markup-
  compatibility/2006"
  xmlns:a="http://schemas.openxmlformats.org/MyExtension/v1"
  xmlns:b="http://schemas.openxmlformats.org/MyExtension/v1"
  mc:Ignorable="a">
  <a:IgnoreMe />
  <b:IgnoreMeToo />
</Circles>
```

end example]

9.1.2 ProcessContent Attribute

The ProcessContent attribute value contains a whitespace-delimited list of element-qualified names identifying the expanded names of elements whose content shall be processed, even if the elements themselves are ignored. In any qualified name in the list, the wildcard character "*" can replace the local name to indicate that the content all elements in the namespace shall be processed.

A markup consumer that encounters an ignored element whose expanded name matches the expanded name of an element identified in the ProcessContent attribute value, the markup consumer shall consider that element to be a processed element, regardless of whether or not the element's qualified name matches the qualified name specified in the ProcessContent attribute value.

A markup consumer that encounters a processed element shall process the contents of that element as if the contents were directly embedded within the parent element of the ignored element.

The ProcessContent attribute value shall not reference any element name that does not belong to a namespace that is identified by the Ignorable attribute of the same element.

The value of the ProcessContent attribute can be an empty or blank string. When a markup consumer encounters such a value, it shall proceed as if the ProcessContent attribute was not provided.

Markup producers shall not generate an element that has an xml:lang or xml:space attribute if that element is identified by a ProcessContent attribute value. Markup consumers that ignore an element that has an xml:lang or xml:space attribute and is also identified by a ProcessContent attribute value shall generate an error. Markup consumers that encounter a non-ignored element that has an xml:lang or xml:space attribute and is also identified by a ProcessContent attribute value might generate an error.

[*Example*: Example 9–3. Processing Ignorable and ProcessContent attributes

A Version 1 markup consumer ignores the blue, black, and red circles, but does render the yellow and green circles.

```
<Circles
  xmlns="http://schemas.openxmlformats.org/Circles/v1"
  xmlns:mc="http://schemas.openxmlformats.org/markup-
  compatibility/2006"
  xmlns:v2="http://schemas.openxmlformats.org/Circles/v2"
  mc:Ignorable="v2"
  mc:ProcessContent="v2:Blink" >
  <v2:Watermark Opacity="v0.1">
    <Circle Center="0,0" Radius="20" Color="Blue" />
    <Circle Center="25,0" Radius="20" Color="Black" />
    <Circle Center="50,0" Radius="20" Color="Red" />
  </v2:Watermark>
  <v2:Blink>
    <Circle Center="13,0" Radius="20" Color="Yellow" />
    <Circle Center="38,0" Radius="20" Color="Green" />
  </v2:Blink>
</Circles>
```

The Version 1 markup consumer, unaware of Version 2 markup, renders the above markup as if it had processed the following markup:

```
<Circles
  xmlns="http://schemas.openxmlformats.org/Circles/v1" >
  <Circle Center="13,0" Radius="20" Color="Yellow" />
  <Circle Center="38,0" Radius="20" Color="Green" />
</Circles>
```

end example]

9.1.3 PreserveElements and PreserveAttributes Attributes

The Ignorable attribute presents a challenge unique to markup editors in deciding when and how to *preserve* ignored markup in the face of modification and when to discard ignored markup. In the absence of explicit knowledge of the specification governing the unrecognized ignored markup, it is difficult for a markup editor to preserve ignored markup while simultaneously maintaining document conformance with whatever specification governs that markup. Similarly, it is difficult for the markup editor to avoid undesired change to the edited document's semantics as they would be interpreted by a markup consumer whose implementation is based on the specification governing the preserved markup.

A markup editor might use the presence of PreserveElements and PreserveAttributes attributes as hints for deciding what ignored elements and attributes to preserve when markup is modified. The markup editor's specific interpretation of those hints shall be governed by the markup specification or specifications that formed the basis for the markup editor's implementation.

Markup specifications should specify conditions under which markup editors should preserve ignored markup. Markup specifications should define the widest possible set ofconditions under which markup editors should preserve ignored markup. [*Note*: If preservation conditions are too widely defined, future versions and extensions will be over constrained in what new semantics they can introduce. *end note*]

If a markup specification lacks such guidance, markup editors for markup documents governed by that markup specification should be conservative in their preservation behavior Before preserving any ignored markup, markup editors should attempt to establish confidence that the preserved markup will be acceptable to, and interpretable with acceptable semantics by, all imaginable markup consumers that understand future versions of extensions. [*Note*: Such confidence could be established by deep understanding of the base specification. *end note*]

Even in the presence of explicit preservation guidance in a markup specification, any markup editor might choose to discard together all ignored markup without regard to the presence of any PreserveElements or PreserveAttributes attribute. [*Note:* The explicit presence of the Ignorable attribute indicates that discarding such markup before editing produces a document that conforms to relevant specifications and is self-sufficient in its semantic interpretation. *end note*]

A markup specification might define conditions calling for the preservation of some ignored elements or attributes without requiring the presence of any PreserveElements or PreserveAttributes attribute.

A markup specification might restrict preservation of elements identified by the PreserveElements attribute to those elements that are descendants of particular elements. Likewise, a markup specification might restrict the set of elements whose ignored attributes can be preserved, as identified by the PreserveAttributes element, to those that are descendants of particular elements. Regardless of any such restrictions, markup consumers shall always accept, but possibly disregard, PreserveElements and PreserveAttributes attributes on any element.

9.1.3.1 PreserveElements Attribute

The PreserveElements attribute value contains a whitespace-delimited list of element qualified names identifying the expanded names of elements that a markup producer suggests for preservation by markup editors, even if the elements themselves are ignored. In any qualified name in the list, the wildcard character “*” can replace the local name to suggest that all elements in the namespace can be preserved.

If a markup consumer encounters an ignored element whose expanded name matches the expanded name of an element identified in the PreserveElements attribute value, the markup consumer shall consider that element to be a candidate for preservation, regardless of whether or not the element’s qualified name matches the qualified name specified in the PreserveElements attribute value.

When an element is ignored and preserved, all of its unprefixed attributes shall also be preserved along with any preserved attributes identified in a PreserveAttributes attribute value.

The PreserveElements attribute value shall not reference any element name that does not belong to a namespace that is identified by the Ignorable attribute of the same element.

The value of the PreserveElements attribute can be an empty or blank string. When a markup consumer encounters such a value, it shall proceed as if the PreserveElements attribute was not provided.

9.1.3.2 PreserveAttributes Attributes

The PreserveAttributes attribute value contains a whitespace-delimited list of attribute qualified names identifying the expanded names of attributes that a markup producer suggests for preservation by markup editors. [*Note*: An attribute cannot be preserved if it appears on a non-preserved ignored element. *end note*] In any qualified name in the list, the wildcard character "*" can replace the local name to indicate that all attributes in the namespace should be preserved. [*Note*: An unprefixed local attribute cannot be identified by a PreserveAttributes attribute value. *end note*]

If a markup consumer encounters an ignored attribute whose expanded name matches the expanded name of an attribute identified in the PreserveAttributes attribute value, the markup consumer shall consider that attribute to be a candidate for preservation, regardless of whether or not the attribute's qualified name matches the qualified name specified in the PreserveAttributes attribute value.

The ProcessAttributes attribute value shall not reference any attribute name that does not belong to a namespace that is identified by the Ignorable attribute of the same element.

The value of the PreserveAttributes attribute can be an empty or blank string. When a markup consumer encounters such a value, it shall proceed as if the PreserveAttributes attribute was not provided.

9.1.4 MustUnderstand Attribute

The MustUnderstand attribute value containsa whitespace-delimited list of namespace prefixes identifying a set of namespace names. A markup consumer that does not understand these identified namespaces shall not continue to process the markup document, regardless of whether the non-understood namespace was identified as an ignorable namespace on an ancestor element. Markup consumers shall generate an error condition if one or more of these identified namespaces is not understood.

The value of the MustUnderstand attribute can be an empty or blank string. When a markup consumer encounters such a value, it shall proceed as if the MustUnderstand attribute was not declared.

[*Note*: §9.2 clarifies the rules for processing the MustUnderstand attribute when it is applied to a Choice or Fallback element, or when it is applied to a descendant element of one of those elements. *end note*]

[*Example*: Example 9–4. Processing an attribute's prefixed qualified name

The declaration of a Version 2 attribute causes a Version 1 markup consumer to trigger an error when processing the last Circle element.

```
<Circles
  xmlns="http://schemas.openxmlformats.org/Circles/v1"
  xmlns:mc="http://schemas.openxmlformats.org/markup-
  compatibility/2006"
  xmlns:v2="http://schemas.openxmlformats.org/Circles/v2">
```

```
    <Circle Center="0,0" Radius="20" Color="Blue" />
    <Circle Center="25,0" Radius="20" Color="Black" />
    <Circle Center="50,0" Radius="20" Color="Red" />
    <Circle Center="13,0" Radius="20" Color="Yellow" />
    <Circle Center="38,0" Radius="20" Color="Green"
      v2:Opacity="0.5" />
  </Circles>
```

Example 10–5. Processing a MustUnderstand attribute

The value of the MustUnderstand attribute causes a Version 1 markup consumer to trigger an error when processing the root Circles element.

```
  <Circles
    xmlns="http://schemas.openxmlformats.org/Circles/v1"
    xmlns:mc=http://schemas.openxmlformats.org/markup-
    compatibility/2006
    xmlns:v2="http://schemas.openxmlformats.org/Circles/v2"
    mc:MustUnderstand="v2">
    <Circle Center="0,0" Radius="20" Color="Blue" />
    <Circle Center="25,0" Radius="20" Color="Black" />
    <Circle Center="50,0" Radius="20" Color="Red" />
    <Circle Center="13,0" Radius="20" Color="Yellow" />
    <Circle Center="38,0" Radius="20" Color="Green"
      v2:Opacity="0.5" />
  </Circles>
```

end example]

9.2 Alternate-Content Elements

[*Note*: Markup producers can generate a markup document that includes multiple markup alternatives, each labeled with the namespaces that needs to be understood by any markup consumer choosing that alternative. A markup consumer shall choose only a single alternative. A particular markup alternative can exploit features introduced in subsequent revisions of the markup specification or in extensions to the markup specification.

In some cases, the Ignorable attribute could provide flexibility sufficient for a markup producer to create an acceptable experience to users of a markup consumer that is unaware of any revisions or extensions. In other cases, it could be desirable or necessary for markup producers to provide different markup alternatives, with one alternative processed by markup consumers implemented according to particular revisions or extensions of the markup specification, and others processed by markup consumers implemented according to different revisions or extensions of the markup specification. *end note*]

9.2.1 AlternateContent Element

The AlternateContent element contains the full set of all possible markup alternatives. Each possible alternative is contained within either a Choice or Fallback child element of the AlternateContent element.

An AlternateContent element shall contain one or more Choice child elements, optionally followed by a Fallback child element. If present, there shall be only one Fallback element, and it shall follow all Choice elements. An AlternateContent element shall not be the child of an AlternateContent element.

More than one Choice child element might be specified, each identifying the namespaces that a markup consumer needs to understand in order to choose the markup alternative contained within the Choice element. Markup consumers shall rely solely on the namespaces identified by the Choice element rather than the alternate content markup itself in order to decide which content to use.

AlternateContent elements might include the attributes Ignorable, MustUnderstand, ProcessContent, PreserveElements, and PreserveAttributes described in this specification. These attributes' qualified names shall be prefixed when associated with an AlternateContent element. A markup consumer shall generate an error if it encounters an unprefixed attribute name associated with an AlternateContent element. [*Note*: A namespace declaration is not considered to be an unprefixed attribute name. *end note*]

AlternateContent elements might have ignored attributes or contain ignored child elements. Markup consumers shall not generate an error when encountering such attributes or child elements. However, markup consumers shall generate an error when encountering an attribute or child element of the AlternateContent element that belongs to a namespace that is neither understood nor ignorable. [*Note*: In addition to Choice and Fallback elements, an ignored element can occur as a child of AlternateContent. *end note*]

When a markup consumer processes an AlternateContent element, each successive Choice or Fallback element is considered in markup order for selection based on its attributes. If a Choice or Fallback element is not selected for processing, all of the child and descendant elements of that Choice or Fallback element shall be treated as if they did not exist. A markup consumer shall not generate an error based on any MustUnderstand attribute applied to an element contained within the content of the unselected Choice or Fallback element. A markup consumer shall not generate an error based on any markup that is contained within an unselected Choice or Fallback element, even if that markup is not conformant to this specification.

In processing an AlternateContent element, the attributes of every child Choice or Fallback element shall be processed and checked for conformance to this specification, regardless of whether the Choice or Fallback element precedes or follows the selected alternative in markup order. If the AlternateContent element's child Choice or Fallback elements include an attribute from a namespace that is not understood and is not ignorable, the markup consumer shall generate an error. Likewise, a markup consumer shall generate an error if it encounters a MustUnderstand attribute included on a Choice or Fallback element that identifies a namespace that it does not understand.

The content of the selected Choice or Fallback element is processed as though it replaces the entire AlternateContent element. All namespace declarations and compatibility-rule attributes present on the AlternateContent element or selected Choice or Fallback element shall be processed as though they appeared

on every child element of the selected Choice or Fallback element. This logical removal of the parent AlternateContent, Choice, and Fallback elements shall not change the expanded name of any element or attribute contained within the selected Choice of Fallback element. Additionally, this logical removal shall not change the set of ignorable namespaces, or their corresponding preservation and content-processing behavior, when processing the contents of the selected Choice or Fallback element.

[*Note*: The AlternateContent element can appear as the root element of a markup document. *end note*]

Markup producers shall not generate AlternateContent elements that have the xml:lang or xml:space attributes. Markup consumers shall generate an error if they encounter the xml:lang or xml:space attributes on an AlternateContent element.

9.2.2 Choice Element

All Choice elements shall have a Requires attribute whose value contains a whitespace-delimited list of namespace prefixes that identify the namespaces that a markup consumer needs to understand to select that Choice and process the content. If the markup consumer does not understand all of the namespaces identified, it shall not select that Choice element. Markup consumers shall select the first Choice element, in markup order, in which all namespaces identified by the Requires attribute are understood.

Choice elements can include the attributes Ignorable, MustUnderstand, ProcessContent, PreserveElements, and PreserveAttributes described in this specification. These attributes shall be prefixed when on a Choice element. A markup consumer shall generate an error if it encounters a Choice element having any unprefixed attribute name, with the single exception of the Requires attribute, which shall be unprefixed. [*Note*: A namespace declaration is not considered to be an unprefixed attribute name. *end note*] A markup consumer that encounters a prefixed Requires attribute, when the prefix is associated with the Markup Compatibility namespace, shall generate an error.

Choice elements might have ignored attributes. Markup consumers shall not generate an error when encountering such attributes. However, markup consumers shall generate an error when encountering an attribute of the Choice element that belongs to a namespace that is neither understood nor ignorable.

Markup producers shall not generate Choice elements that have the xml:lang or xml:space attributes. Markup consumers shall generate an error if they encounter the xml:lang or xml:space attributes on a Choice element, regardless of whether the element is preceded by a selected Choice element.

9.2.3 Fallback Element

If no Choice element can be selected, markup consumers shall use the content provided in a Fallback element, if present. If no Choice element can be selected and no Fallback element is provided, the document content is processed as if the AlternateContent element were absent.

Fallback elements can include the attributes Ignorable, MustUnderstand, ProcessContent, PreserveElements, and PreserveAttributes described in this specification. These attributes shall be prefixed when on a Fallback element. A markup consumer shall generate an error if it encounters a Fallback element

having any unprefixed attribute name. [*Note*: A namespace declaration is not considered to be an unprefixed attribute name. *end note*]

Fallback elements might have ignored attributes. Markup consumers shall not generate an error when encountering such attributes. However, markup consumers shall generate an error when encountering an attribute of the Fallback element that belongs to a namespace that is neither understood nor ignorable.

Markup producers shall not generate Fallback element s that have the xml:lang or xml:space attributes. Markup consumers shall generate an error if they encounter the xml:lang or xml:space attributes on a Fallback element, regardless of whether the element is preceded by a selected Choice element.

9.2.4 Alternate-Content Examples

The following examples illustrate the usage of alternate-content elements.

[*Example*: Example 9–6. Processing AlternateContent markup

In this example, luminance is expressed as an attribute of a Circle element for markup consumers supporting the Version 3 namespace, identified by the v3 namespace prefix, and as an attribute of a LuminanceFilter element for other markup consumers.

```
<Circles
  xmlns="http://schemas.openxmlformats.org/Circles/v1"
  xmlns:mc=http://schemas.openxmlformats.org/markup-
  compatibility/2006
  xmlns:v2=http://schemas.openxmlformats.org/Circles/v2
  xmlns:v3=http://schemas.openxmlformats.org/Circles/v3
  mc:Ignorable="v2 v3">
  <mc:AlternateContent>
    <mc:Choice Requires="v3">
      <v3:Circle Center="0,0" Radius="20" Color="Blue"
        Opacity="0.5" Luminance="13" />
      <v3:Circle Center="25,0" Radius="20" Color="Black"
        Opacity="0.5" Luminance="13" />
      <v3:Circle Center="50,0" Radius="20" Color="Red"
        Opacity="0.5" Luminance="13" />
      <v3:Circle Center="13,0" Radius="20" Color="Yellow"
        Opacity="0.5" Luminance="13" />
      <v3:Circle Center="38,0" Radius="20" Color="Green"
         Opacity="0.5" Luminance="13" />
    </mc:Choice>
    <mc:Fallback>
      <LuminanceFilter Luminance="13">
         <Circle Center="0,0" Radius="20" Color="Blue"
           v2:Opacity="0.5" />
```

```
            <Circle Center="25,0" Radius="20" Color="Black"
              v2:Opacity="0.5" />
            <Circle Center="50,0" Radius="20" Color="Red"
              v2:Opacity="0.5" />
            <Circle Center="13,0" Radius="20" Color="Yellow"
              v2:Opacity="0.5" />
            <Circle Center="38,0" Radius="20" Color="Green"
              v2:Opacity="0.5" />
          </LuminanceFilter>
        </mc:Fallback>
      </mc:AlternateContent>
    </Circles>
```

end example]

[*Example*: Example 9–7. Processing AlternateContent markup using namespaces

In this example, if the markup consumer understands the metallic-finishes namespace, the contents of the mc:Choice block are used. If it does not, the contents of mc:Fallback are used instead.

```
    <Circles
      xmlns="http://schemas.openxmlformats.org/Circles/v1"
      xmlns:mc="http://schemas.openxmlformats.org/markup-
      compatibility/2006" >
      <mc:AlternateContent
        xmlns:m="http://schemas.openxmlformats.org/metallic-
        finishes/v1">
        <mc:Choice Requires="m">
          <Circle m:Finish="GoldLeaf" Center="100,100" Radius="50"
            />
        </mc:Choice>
        <mc:Fallback>
          <Circle Fill="Gold" Center="100,100" Radius="50" />
        </mc:Fallback>
      </mc:AlternateContent>
    </Circles>
```

end example]

10. Namespace Subsumption

10.1 The Subsumption Process

A namespace *subsumes* another namespace if it includes all of the elements, attributes, and attribute values of the subsumed namespace.

A markup specification that defines a subsuming namespace shall require that any instance of the following constructs that would be recognized in the subsumed namespace shall also be recognized and interpreted identically in the subsuming namespace.

- Element local names
- Unprefixed attribute names of elements
- Prefixed attribute names of elements
- Attribute values
- Element contents

Subsumption is the process by which a markup consumer recognizes and interprets elements, attributes, and attribute values in a subsumed namespace as if they occurred in the subsuming namespace.

When performing subsumption, markup consumers might confirm that the markup using the old namespace is compatible with the specification of the old namespace

Any markup specification that relies on the Markup Compatibility namespace should define a namespace naming convention for use by future versions of that specification when those future versions introduce new namespaces that subsume older namespaces.

When markup consumers encounter a non-understood namespace name, they might examine it for a naming convention that suggests that an understood namespace is being subsumed. The suggestion of a subsumption relationship shall not suppress error processing triggered by the non-understood namespace name. If error processing is triggered, markup consumers might use the suggested subsumption relationship to choose the most appropriate error message or error code. [*Example*: One error message might encourage upgrading to a newer application version, while another might merely highlight the non-understood namespace name. *end example*]

10.2 Special Considerations for Attributes

§6.3 of the W3C Recommendation “Namespaces in XML 1.0 (Second Edition)” makes a distinction between prefixed attributes and unprefixed attributes sharing the same local name. Given an element whose expanded name refers to namespace *N*, an unprefixed attribute with local name *L* is distinct from a prefixed attribute with local name *L* and namespace *N*. The existence of this distinction is important in defining correct subsumption behavior for markup consumers.

A subsuming namespace might extend a pre-existing element local name with a new unprefixed attribute. Similarly, a subsuming namespace might create a new attribute designed for prefixed use.

[*Note*: The statements above introduce a potential ambiguity in defining the correct behavior of a markup consumer performing subsumption.

Assume that the namespace associated with prefix v2 subsumes the namespace associated with v1. Suppose a markup consumer that understands the v2 namespace encounters markup of the following form:

```
<v1:OldElement mc:Ignorable=”v2” v2:NewAttribute=”value” />
```

How should that markup consumer interpret that markup? Should it be considered equivalent to the following markup?

```
<v2:OldElement NewAttribute=”value” />
```

Or should it be considered equivalent to the following markup?

```
<v2:OldElement v2:NewAttribute=”value” />
```

According to “Namespaces in XML 1.0 (Second Edition)”, these two potential pieces of markup are not equivalent. Additionally, the XML Schema specification allows for the construction of different XSD schemas that validate one, the other, or both of these constructs. *end note*]

When processing an element from an older namespace that carries a prefixed attribute from a newer, subsuming namespace, a markup consumer shall decide whether to treat the new attribute as if its expanded name refers to the new namespace or as if its expanded name refers to no namespace. If a subsuming namespace adds a new attribute or permissible attribute value to an element that was present in the subsumed namespace, the markup specification that defines the subsuming namespace shall state which of the two subsumption behaviors shall be used by markup consumers and assumed by markup producers.

[*Note*: Example 13-1 illustrates how a markup preprocessor handles each of the two possible behaviors. *end note*]

In order to support a preprocessing model for Markup Compatibility elements and attributes, specifications should restrict the use of any combination of prefixed and unprefixed attributes with the same local name.

A namespace should not be subsumed by a newer namespace if the older namespace includes both a prefixed attribute and an unprefixed attribute sharing its local name but having a different type or different semantics.

A subsuming namespace should not include both a prefixed attribute and an unprefixed attribute sharing its local name but having a different type or different semantics.

11. Application-Defined Extension Elements

A markup specification using Markup Compatibility elements and attributes might define one or more specific extension elements in the namespaces it defines. *Extension elements* suspend Markup Compatibility processing within their content. Except as noted below, within the content of an extension element, markup consumers shall not treat elements and attributes from non-understood namespaces as Markup Compatibility errors. Similarly, under the same conditions, markup consumers shall disregard elements and attributes from the Markup Compatibility namespace.

A specification for an element nested somewhere within an extension-element might require a markup consumer to re-establish Markup Compatibility processing. Within the scope of such a nested element and its content, a markup consumer shall disregard all Markup Compatibility attributes that were encountered on elements outside of the element that re-establishes Markup Compatibility processing. Within the scope of such a nested element, a markup consumer might understand a set of namespaces that is different from the set of namespaces understood at the point in the markup where the extension element was encountered.

The following examples illustrate two uses of application-defined extension elements:

[*Example*: Example 11–1. An application-defined XML island

An extension element can be used to introduce an “island” of unprocessed XML whose markup is otherwise unconstrained by the application's specification. The specification of the island element can further require preservation of the contents of the island by markup processors, without requiring the use of the PreserveElements and PreserveAttributes Markup Compatibility attributes. *end example*]

[*Example*: Example 11–2. An application-defined add-in element

Some markup specifications and markup consumers can use an extension element to implement an add-in model. In an add-in model, the specification for the contents of the extension element is separate from the specification for the extension element itself.

The specification for some particular nested content can include support for Markup Compatibility elements and attributes, while the specification for other nested content could omit such support. If the specification for the nested content does include support for Markup Compatibility elements and attributes, the Markup Compatibility processing state is reset temporarily for processing of the nested content. Any Ignorable attribute-value associated with an extension element or any of its ancestor elements is “forgotten” during the processing of content nested within that extension element. In an add-in model the set of namespaces assumed to be understood when processing descendant elements of an extension element is completely unrelated to the set of understood namespaces when that extension element itself is processed.

In this example, the "Circles" specification includes an extension AddIn element, allowing for nested markup to be handled by a markup consumer that does not process Circle markup. The specification for the nested "TextFlow" markup does not provide for the processing of Markup Compatibility elements and attributes.

```
<Circles
  xmlns="http://schemas.openxmlformats.org/Circles/v1"
  xmlns:mc=http://schemas.openxmlformats.org/markup-
  compatibility/2006
  xmlns:v2="http://schemas.openxmlformats.org/Circles/v2"
  mc:Ignorable="v2 ">
  <Circle Center="0,0" Radius="20" Color="Blue"
    v2:Opacity="0.5" />
  <Circle Center="25,0" Radius="20" Color="Black"
    v2:Opacity="0.5" />
  <Circle Center="50,0" Radius="20" Color="Red"
    v2:Opacity="0.5" />
  <Circle Center="13,0" Radius="20" Color="Yellow"
    v2:Opacity="0.5" />
  <Circle Center="38,0" Radius="20" Color="Green"
    v2:Opacity="0.5" />
  <AddIn Center="25,10" Radius="10" CodeBase=
    "http://www.openxmlformats.org/code/TextFlowAddin.jar">
    <TextFlow
      xmlns="http://schemas.openxmlformats.org/TextFlow/v1">
      <!--
        Because the TextFlow specification does not make use
        of Markup Compatibility elements and attributes,
        the TextFlow processor would consider the presence
        of an mc:Ignorable attribute to be an error condition.
        Because the TextFlow specification is completely
        unaware of all versions of the Circles specification,
        the TextFlow processor would also consider the
        presence of a Circle or v2:Ellipse element to be an
        error condition.
      -->
      <Paragraph>How are <Bold>you</Bold>?</Paragraph>
    </TextFlow>
  </AddIn>
</Circles>
```

end example]

12. Preprocessing Model for Markup Consumption

This clause is informative.

Markup consumers might rely on a preprocessing model to support Markup Compatibility. Such a model can simplify the markup consumer's implementation and allow it to rely on XML schema validation for the preprocessed markup document. Furthermore, a single preprocessing implementation can be shared by markup consumers that target various markup specifications.

A markup preprocessor accepts as input XML markup containing a variety of elements and attributes from the following namespace categories:

- The Markup Compatibility namespace.
- *Current namespaces*: Understood namespaces that have not been subsumed by newer namespaces.
- *Obsolete namespaces*: Understood namespaces known to have been subsumed by newer namespaces).
- Non-understood namespaces.

The markup preprocessor transforms its input markup into output markup that contains elements and attributes drawn only from current and non-understood namespaces. This transformation is disabled for input markup nested within an application-defined extension element. The markup preprocessor accomplishes its transformation using the following rules:

The markup preprocessor checks for the correct usage of all elements and attributes in the Markup Compatibility namespace, and triggers error processing if an element or attribute in that namespace violates the requirements of this Markup Compatibility Part.

The markup preprocessor interprets occurrences of the MustUnderstand attribute, and triggers error processing when appropriate.

The markup preprocessor removes all elements and attributes in the Markup Compatibility namespace, removing the contents of unselected Choice and Fallback elements.

The markup preprocessor removes all elements and attributes in obsolete namespaces, and replaces them with identically named elements and attributes in current namespaces.

Guided by the Ignorable attribute, the markup preprocessor removes some elements and attributes in non-understood namespaces, leaving others undisturbed.

Guided by the ProcessContent attribute, the markup preprocessor removes all nested content within some removed elements from non-understood namespaces.

In addition to producing such transformed output, the markup preprocessor might also implement mechanisms to optionally provide to a markup editor the additional information necessary to preserve some ignored content.

[*Note*: The output of the markup preprocessor can be validated against a schema written entirely in terms of current namespaces. Where a schema does not allow for an open attribute or content model, occurrences of elements and attributes from non-understood namespaces in the markup preprocessor's output will trigger validation failures. Where a schema does allow for an open attribute or content model, occurrences of elements and attributes from non-understood namespaces will validate successfully.

If a markup preprocessor is used in conjunction with XSD schema validation, the schema should set the value of the elementFormDefault attribute of the root schema element to the value `qualified`. *end note*]

[*Example*: Example 12–1. Preprocessing using Markup Compatibility elements and attributes

Given the following input document:

```
<Circles
  xmlns="http://schemas.openxmlformats.org/Circles/v1"
  xmlns:mc="http://schemas.openxmlformats.org/markup-
  compatibility/2006"
  xmlns:v2="http://schemas.openxmlformats.org/Circles/v2"
  xmlns:v3="http://schemas.openxmlformats.org/Circles/v3"
  mc:Ignorable="v3"
  mc:ProcessContent="v3:Blink">
  <mc:AlternateContent>
    <mc:Choice Requires="v3">
      <v3:Circle Center="0,0" Radius="20" Color="Blue"
        Opacity="0.5" Luminance="13" />
      <v3:Circle Center="25,0" Radius="20" Color="Black"
        Opacity="0.5" Luminance="13" />
      <v3:Circle Center="50,0" Radius="20" Color="Red"
        Opacity="0.5" Luminance="13" />
      <v3:Circle Center="13,0" Radius="20" Color="Yellow"
        Opacity="0.5" Luminance="13" />
      <v3:Circle Center="38,0" Radius="20" Color="Green"
        Opacity="0.5" Luminance="13" />
    </mc:Choice>
    <mc:Fallback>
      <LuminanceFilter Luminance="13">
        <Circle Center="0,0" Radius="20" Color="Blue"
          v2:Opacity="0.5" />
```

```
            <Circle Center="25,0" Radius="20" Color="Black"
              v2:Opacity="0.5" />
            <Circle Center="50,0" Radius="20" Color="Red"
              v2:Opacity="0.5" />
            <Circle Center="13,0" Radius="20" Color="Yellow"
              v2:Opacity="0.5" />
            <Circle Center="38,0" Radius="20" Color="Green"
              v2:Opacity="0.5" />
          </LuminanceFilter>
        </mc:Fallback>
      </mc:AlternateContent>
      <v3:Blink>
        <Circle Center="13,0" Radius="20" Color="Yellow" />
        <Circle Center="38,0" Radius="20" Color="Green" />
      </v3:Blink>
      <v3:Watermark>
        <Circle Center="50,0" Radius="20" Color="Red" />
      </v3:Watermark>
    </Circles>
```

According to §10.2, this markup document's markup specification, when it defines the namespace name `http://schemas.openxmlformats.org/Circles/v2`, shall state the intended subsumption behavior when processing a Version 2 Opacity attribute that is applied to a Circle element from an earlier version. Depending on that statement, a Version 2 markup consumer renders the above markup as if it had processed one of the following pieces of preprocessed markup:

```
    <v2:Circles
      v2:xmlns="http://schemas.openxmlformats.org/Circles/v2">
      <v2:LuminanceFilter Luminance="13">
        <v2:Circle Center="0,0" Radius="20" Color="Blue"
          Opacity="0.5" />
        <v2:Circle Center="25,0" Radius="20" Color="Black"
          Opacity="0.5" />
        <v2:Circle Center="50,0" Radius="20" Color="Red"
          Opacity="0.5" />
        <v2:Circle Center="13,0" Radius="20" Color="Yellow"
          Opacity="0.5" />
        <v2:Circle Center="38,0" Radius="20" Color="Green"
          Opacity="0.5" />
      </v2:LuminanceFilter>
      <v2:Circle Center="13,0" Radius="20" Color="Yellow" />
      <v2:Circle Center="38,0" Radius="20" Color="Green" />
    </v2:Circles>
```

Or:

```
  <v2:Circles
    v2:xmlns="http://schemas.openxmlformats.org/Circles/v2">
    <v2:LuminanceFilter Luminance="13">
      <v2:Circle Center="0,0" Radius="20" Color="Blue"
        v2:Opacity="0.5" />
      <v2:Circle Center="25,0" Radius="20" Color="Black"
        v2:Opacity="0.5" />
      <v2:Circle Center="50,0" Radius="20" Color="Red"
        v2:Opacity="0.5" />
      <v2:Circle Center="13,0" Radius="20" Color="Yellow"
        v2:Opacity="0.5" />
      <v2:Circle Center="38,0" Radius="20" Color="Green"
        v2:Opacity="0.5" />
    </v2:LuminanceFilter>
    <v2:Circle Center="13,0" Radius="20" Color="Yellow" />
    <v2:Circle Center="38,0" Radius="20" Color="Green" />
  </v2:Circles>
```

end example]

End of informative text.

Annex A. Validation Using NVDL

This annex is informative.

Namespace-based Validation Dispatching Language (NVDL) allows documents to be decomposed into validation candidates, each of which may be validated independently.

NVDL may be used for validation against the normative requirements of this Part. It may also be used for validation against the combination of Office Open XML documents (including the elements and attributes defined in this Part) and any extensions.

A.1 Validation Against Requirements of this Part

A markup document may satisfy requirements of this Annex without being an Office Open XML document. The following NVDL script examines whether a given document correctly uses the attributes and elements as defined by this Part.

This NVDL script first extracts elements and attributes in the Markup Compatibility namespace, and then validates them against the appropriate RELAX NG schemas.

Note that AlternateContent, Choice and Fallback elements are allowed to have foreign elements and attributes.

```
<?xml version="1.0" encoding="UTF-8"?>
<rules xmlns="http://purl.oclc.org/dsdl/nvdl/ns/structure/1.0">
  <namespace match="attributes" ns="http://schemas.openxmlformats.org/markup-
    compatibility/2006">
    <validate schemaType="application/relax-ng-compact-syntax">
      <schema>
        namespace mc="http://schemas.openxmlformats.org/markup-
         compatibility/2006"
        nsList = list { xsd:NCName* }
        qnameList = list { (xsd:QName | xsd:string {pattern = "\i\c*:\*" })*}
        start = element * {
          attribute mc:Ignorable { nsList }?,
          attribute mc:ProcessContent { qnameList }?,
          attribute mc:PreserveElements { qnameList }?,
          attribute mc:PreserveAttributes { qnameList }?,
          attribute mc:MustUnderstand { nsList }?
        }
      </schema>
    </validate>
```

```
    </namespace>
    <namespace match="elements" ns="http://schemas.openxmlformats.org/markup-
    compatibility/2006">
      <validate schemaType="application/relax-ng-compact-syntax">
        <schema>
          default namespace ="http://schemas.openxmlformats.org/markup-
            compatibility/2006"
          nsList = list { xsd:NCName* }
          qnameList = list { (xsd:QName | xsd:string {pattern = "\i\c*:\*" })*}
          start = element AlternateContent {choice+,fallback?}
          choice = element Choice {attribute Requires { nsList }, text}
          fallback = element Fallback {text}
        </schema>
      </validate>
    </namespace>
    <namespace ns="" match="attributes">
      <attach/>
    </namespace>
    <anyNamespace match="elements attributes">
      <allow/>
    </anyNamespace>
  </rules>
```

The two RELAX NG schemas embedded in the above NVDL script may be rewritten in the analogous XML Schema form.

A.2 Validation Against the Combination of Office Open XML and Extensions

An extension of Office Open XML specified using the mechanisms defined in this Part may be captured by an NVDL script that invokes the Office Open XML schema and schemas for the extension.

The following schema allows two extensions. They have the namespaces http://www.example.com/myExtensionWithFallback and http://www.example.com/myExtensionWithoutFallback. The first extension is accompanied with a parent AlternateContent element and a sibling Fallback element, while the second one may appear anywhere in the document without AlternateContent or Fallback elements.

```
  <?xml version="1.0" encoding="UTF-8"?>
  <rules xmlns="http://purl.oclc.org/dsdl/nvdl/ns/structure/1.0" startMode="top">
      <mode name="top">
          <namespace
            ns="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
              <validate schema="wml.xsd" useMode="nested"/>
          </namespace>
      </mode>
```

```
        <mode name="nested">
            <namespace match="attributes elements"
                ns="http://schemas.openxmlformats.org/drawingml/2006/*">
                <attach/>
            </namespace>
            <namespace match="attributes elements"
                ns="http://schemas.openxmlformats.org/officeDocument/2006/*">
                <attach/>
            </namespace>
            <namespace match="attributes elements"
              ns="http://schemas.openxmlformats.org/package/2006/*">
                <attach/>
            </namespace>
            <namespace match="attributes elements"
                ns="http://schemas.openxmlformats.org/presentationml/2006/main">
                <attach/>
            </namespace>
            <namespace match="attributes elements"
                ns="http://schemas.openxmlformats.org/schemaLibrary/2006/main">
                <attach/>
            </namespace>
            <namespace match="attributes elements"
                ns="http://schemas.openxmlformats.org/spreadsheetml/2006/7/main">
                <attach/>
            </namespace>
            <namespace match="attributes elements"
              ns="urn:schemas-microsoft-com:*">
                <attach/>
            </namespace>
            <namespace match="attributes"
                ns="http://schemas.openxmlformats.org/markup-compatibility/2006">
                <validate schemaType="application/relax-ng-compact-syntax">
                    <schema>
                        namespace mc =
                        "http://schemas.openxmlformats.org/markup-
                          compatibility/2006"
                        nsList = list { xsd:NCName* }
                        qnameList = list { (xsd:QName | xsd:string {pattern =
                          "\i\c*:\*" })*}
                        start = element * {
                          attribute mc:Ignorable { nsList }?,
                          attribute mc:ProcessContent { qnameList }?,
                          attribute mc:PreserveElements { qnameList }?,
```

```
                    attribute mc:PreserveAttributes { qnameList }?,
                    attribute mc:MustUnderstand { nsList }?
                  }
              </schema>
          </validate>
      </namespace>
      <namespace match="elements"
        ns="http://schemas.openxmlformats.org/markup-compatibility/2006">
          <validate schemaType="application/relax-ng-compact-syntax">
              <schema>
                  default namespace =
                  "http://schemas.openxmlformats.org/markup-
                    compatibility/2006"
                  nsList = list { xsd:NCName* }
                  qnameList = list { (xsd:QName | xsd:string {pattern =
                    "\i\c*:\*" })*}
                  start = element AlternateContent {choice,fallback}
                  choice = element Choice {attribute Requires { nsList },
                    text}
                  fallback = element Fallback {text}
              </schema>
              <mode>
                  <anyNamespace>
                      <allow/>
                  </anyNamespace>
              </mode>
              <context path="Choice">
                  <mode>
                      <namespace
                        ns="http://www.example.com/myExtenstionWithFallback">
                          <validate schema="myExtensionWithFallback.rng">
                              <mode>
                                  <anyNamespace>
                                      <attach/>
                                  </anyNamespace>
                              </mode>
                          </validate>
                      </namespace>
                  </mode>
              </context>
          </validate>
          <unwrap>
              <mode>
```

```
                        <anyNamespace>
                            <allow/>
                        </anyNamespace>
                    </mode>
                    <context path="Fallback">
                        <mode>
                            <anyNamespace>
                                <attach/>
                            </anyNamespace>
                        </mode>
                    </context>
                </unwrap>
            </namespace>
            <namespace ns="http://www.example.com/myExtensionWithoutFallback">
                <validate schema="myExtensionWithoutFallback.rng">
                    <mode>
                        <anyNamespace>
                            <attach/>
                        </anyNamespace>
                    </mode>
                </validate>
            </namespace>
        </mode>
    </rules>
```

End of informative text.

Annex B. Bibliography

This annex is informative.

The following documents are useful references for implementers and users of this specification, in addition to the normative references:

ISO/IEC 197575-4, Information technology - Document Schema Definition Languages (DSDL) - Part 4: Namespace-based Validation Dispatching Language (NVDL).

ISO/IEC Directives Part 2, Rules for the structure and drafting of International Standards, Fourth edition, 2001, ISBN 92-67-01070-0.

Extensible Markup Language (XML) 1.0 (Fourth Edition), W3C Recommendation, 16 August 2006.

Namespaces in XML 1.0 (Second Edition), W3C Recommendation, 16 August 2006.

RFC 3986 *Uniform Resource Identifier (URI): Generic Syntax, The Internet Society*, Berners-Lee, T., R. Fielding, and L. Masinter, 2005, http://www.rfc-editor.org.

RFC 4234 *Augmented BNF for Syntax Specifications: ABNF, The Internet Society*, Crocker, D., P. Overell, 2005

XML Base, W3C Recommendation, 27 June 2001.

XML Schema Part 1: Structures Second Edition, W3C Recommendation, 28 October 2004.

XML Schema Part 2: Datatypes Second Edition, W3C Recommendation, 28 October 2004.

End of informative text.

Annex C. Index

This annex is informative.

End of informative text.